

# COURSE MANUAL



**WELLCOME  
GENOME  
CAMPUS**  
ADVANCED  
COURSES AND  
SCIENTIFIC  
CONFERENCES



## **WORKING WITH PATHOGEN GENOMES**

**17-22 JUNE 2018**

**University of Cape Town, South Africa**





**Wellcome Genome Campus Advanced Course**

**Working with Pathogen Genomes**

**17-22 June 2018**

**Held at**

University of Cape Town

South Africa

# STAFF

## Advanced Courses Manager

**Rebecca Twells**

Wellcome Genome Campus, Hinxton, Cambridge, UK

## Course Instructors and Assistants

**Adam Reid**

Wellcome Sanger Institute, Hinxton, UK

**Benjamin Kumwenda**

College of Medicine, University of Malawi

**Christine J. Boinett**

Oxford University Clinical Research Unit, Vietnam

**Stephen Doyle**

Wellcome Sanger Institute, Hinxton, UK

**Felix Dube**

University of Cape Town, South Africa

**Daryl Domman**

Wellcome Sanger Institute, Hinxton, UK

**Gerrit Botha**

University of Cape Town, South Africa

**Kamohelo Direko**

University of Western Cape - SANBI, South Africa

## University of Cape Town

**Nicola Mulder**

Head Computational Biology Division & H3ABionet

**Leigh Marinus**

Administrative Assistant

**Suresh Maslamoney**

Systems Administrator

## Advanced Courses Team

**Mrs Yvonne Thornton**

Advanced Courses Administrator Manager

**Mrs Julie Ormond**

Advanced Courses Laboratory Manager

**Dr Darren Hughes**

Advanced Courses Scientific Officer

**Mrs Nicola Stevens**

Advanced Courses Assistant Administrator

**Mrs Karon Chappell**

Advanced Courses Assistant Administrator

**Miss Kate Waite**

Advanced Courses Assistant Laboratory Manager

**Mr Martin Aslett**

Advanced Courses IT Manager

**Mrs Pamela Black**

Advanced Courses Education Officer

**Dr Alice Matimba**

Advanced Courses Overseas Development Officer

**Adam Crewdson**

Advanced Courses Laboratory Assistant

Wellcome Genome Campus  
Advanced Courses

URL: <http://www.wellcome.ac.uk/advancedcourses>  
email: [advancedcourses@wellcomegenomecampus.org](mailto:advancedcourses@wellcomegenomecampus.org)

## **ACKNOWLEDGEMENTS**

Wellcome Genome Campus Advanced Courses and Conferences Programme gratefully acknowledges the support and assistance of staff at the Computational Biology Division, Institute of Infectious Disease & Molecular Medicine, Faculty of Health Sciences, University of Cape Town.

We would like to thank the following for giving the Course opening day seminar:

**Professor Alan Christofels**

Director - South African National Bioinformatics Institute

University of Western Cape, South Africa

Working with Pathogen Genomes Cape Town 17-22 June 2018							
	Sunday 17/06/2018	Monday 18/06/2018 Bus	Tuesday 19/06/2018 Bus	Wednesday 20/06/2018 Bus	Thursday 21/06/2018 Bus	Friday 22/06/2018 Bus	
08:00							08:00
08:30		Virtual Machine Introduction	Module 3 Linux & scripting	Module 5 Phylogenetics	Module 7 Genome annotation & differential expression	Task 2: Bacterial exercise	08:30
09:00		Module 1 Artemis	ADAM	CHRISTINE	ADAM	CHRISTINE	09:00
09:30		BENJAMIN					09:30
10:00		Tea/Coffee	Tea/Coffee	Tea/Coffee	Tea/Coffee	Tea/Coffee	10:00
10:30		Module 1 continued		Module 5 continued	Module 7 continued	Bacterial exercise continued	10:30
11:00							11:00
11:30		Seminar	Seminar	Seminar			11:30
12:00							12:00
12:30	Registration & Lunch (Hotel) Train the Trainer	Lunch	Lunch	Lunch	Lunch	Lunch	12:30
13:00							13:00
13:30	Introduction (Alice)	Module 2 Comparative genomics	Module 4 Short read mapping	Module 6 Genome assembly	Task 1: Eukaryotic exercise	Bacterial exercise continued	13:30
14:00	Introductory Seminar	FELIX	DARYL	STEVE	STEVE	Participant presentations	14:00
14:30							14:30
15:00	Participant Talks  Hotel					Departure	15:00
15:30		Tea/Coffee	Tea/Coffee	Tea/Coffee	Tea/Coffee		15:30
16:00		Module 2 continued	Module 4 continued	Module 6 continued	Eukaryotic exercise continued		16:00
16:30							16:30
17:00							17:00
17:30						17:30	
18:00	Welcome Drinks Bar	Bus to Hotel	Bus to Hotel	Bus to Hotel	Train the Trainer		18:00
18:30					Bus to Hotel		18:30
19:00	Dinner Hotel						19:00
19:30							19:30
20:00							20:00
20:30							20:30
21:00							21:00
21:30							21:30
22:00	Sunday 17/06/2018	Monday 18/06/2018	Tuesday 19/06/2018	Wednesday 20/06/2018	Thursday 21/06/2018	Friday 22/06/2018	22:00

**Wellcome Genome Campus Advanced Courses**  
**Working with Pathogen Genomes**  
**17-22 June 2018**  
University of Cape Town, South Africa  
**Instructors and Assistants**

**Adam James Reid    WGC Sanger Institute**

Adam James Reid received a BSc in Genetics from the University of Sheffield in 2002. He then completed a master's course in Bioinformatics at the University of York and spent two years at AstraZeneca. He studied for his PhD in computational biology with Professor Christine Orengo at University College London. His thesis concerned protein domain classification, remote homology detection, functional prediction and protein complex evolution. Adam then moved to the Wellcome Sanger Institute where he is now a senior staff scientist working on host-parasite interactions in malaria and nematode worms. He is particularly interested in using genomics and transcriptomics to understand the function of parasite gene families.



**Christine J. Boinett    Wellcome Trust Major Overseas Programme,  
Oxford University Clinical Research Unit**

I began my academic career in Imperial College London attaining a BSc in Biochemistry in 2007, and subsequently completed my Msc at the London School of Hygiene and Tropical Medicine in 2008. Later that year I began my PhD at the Animal Health and Veterinary Laboratories Agency (Surrey, UK), registered at Royal Holloway University of London (Surrey, UK), where I studied how resistance to antibiotics in *E. coli* was acquired and transferred in cattle in the UK. I attained my PhD in 2012, and in the same year began my Post-Doctoral Fellowship at the Wellcome Trust Sanger Institute in the Pathogen genomics group with Prof. Julian Parkhill. My research focussed on understanding how resistance to antibiotics emerges in bacterial pathogens using a high throughput mutagenesis approach to inactivate genes in the bacterial genome known as Transposon Directed Insertion site Sequencing (TraDIS). I moved to Vietnam in April of 2016 to work with Prof. Stephen Baker in the Enterics division. I use next generation sequencing techniques to understand the genetic mechanisms of antimicrobial resistance acquisition and transfer in Gram negative pathogens.



**Gerrit Botha    University of Cape Town. South Africa**

Gerrit Botha is a Bioinformatics engineer at the Computational Biology division of the University of Cape Town. His background is electronic engineering but gained experience in bioinformatics whilst working as a software developer at the division. His area of expertise is in human variant analysis, microbial analysis and bioinformatics compute environment and pipeline design.



**Wellcome Genome Campus Advanced Courses**  
**Working with Pathogen Genomes**  
**17-22 June 2018**  
University of Cape Town, South Africa  
**Instructors and Assistants**

**Kamohelo Direko    UWC (SANBI), South Africa**

Mmakamohelo (Kamo) Direko is a PhD student at the UWC (SANBI). She completed an MSc in Bioinformatics, describing the genome of *M. oryx*. The project led to SANBI's ongoing support and training in NGS data pipeline design. Kamo registered towards a PhD as part of a collaborative project, COMBAT-TB. Her project included prediction of sRNAs, their target mRNAs within drug-resistant *M.tb* isolates. COMBAT-TB demonstrates; whole genome sequencing can be used to decipher tuberculosis epidemic in high tuberculosis prevalence settings. The project also characterizes *M.tb* isolates based on SNPs, RDs and sRNAs. Combat-TB trains in microbial pipelines through Galaxy platform.



**Stephen Doyle    WGC Sanger Institute**

My research is focused on understanding the evolution of drug resistance in human and veterinary helminths. After completing a PhD in Genetics at La Trobe University (Melbourne, Australia), I worked on *Onchocerca volvulus*, the human filarial pathogen responsible for the disease river blindness. This work involved population genomic and genome-wide scans for loci associated with ivermectin susceptibility in parasites from Ghana and Cameroon. Since joining the Parasite Genomics group at Sanger in 2015 as a Postdoctoral Fellow, I have switched focus to *Haemonchus contortus* and *Teladorsagia circumcincta*, two important pathogens of sheep and other ruminants. My work involves the assembly and annotation of the genomes of these species, as well as comparative and population genomic analyses to characterise genome-wide variation associated with drug resistance.



**Daryl Domman    WGC Sanger Institute**

As a Postdoctoral Fellow at the Wellcome Sanger Institute, I have focused my efforts towards understanding global infectious disease. I have a keen interest in understanding diarrheal disease, which affects a tremendous number of individuals, but primarily children, worldwide. My work focuses around using large scale genomic data to address the global spread of cholera. Due to the visibility of cholera epidemics, we are using cholera as a model for understanding more large-scale patterns of diarrheal diseases. More broadly, my background is in microbial ecology. I have experience with metagenomics which allows for the genomic characterization of entire microbial communities.





**Wellcome Genome Campus Advanced Courses**  
**Working with Pathogen Genomes**  
**17-22 June 2018**  
University of Cape Town, South Africa  
**Instructors and Assistants**

**Felix Dube: University of Cape Town**

I am a research scientist and lecturer at the Department of Molecular and Cell Biology, University of Cape Town with a PhD in Medical Microbiology. My research focuses on the characterisation of the respiratory microbiome with an immediate interest into the population biology of *Streptococcus pneumoniae*. Using high-throughput whole genome sequencing technology we are able to provide further insights into the *S. pneumoniae* pathobiome, including changes in *S. pneumoniae* sero-epidemiology, pathogenesis, and transmission dynamics. We apply phylogenetic, statistical and computational approaches to unravel genotype-phenotype associations.



**Benjamin Kumwenda College of Medicine, Malawi**

Benjamin Kumwenda is Malawian. He obtained his BSc degree in Computer Sciences from University of Malawi, Chancellor College in 2001 and an Honors degree in Computer Sciences from the University of the Witwatersrand in Johannesburg, South Africa in 2006. He completed his MSc degree in Computer Sciences in 2008 at the University of the Witwatersrand in Johannesburg, South Africa. In 2010, he enrolled for a PhD in Bioinformatics at University of Pretoria, which he graduated in April 2014. He has been a Postdoctoral Research Fellow at the Malawi-Liverpool Wellcome Trust from 2014 to 2015 funded by the H3Africa Bioinformatics Network (H3BioNet) and in 2016 at the University of Liverpool at the Institute of Integrative Biology under the MRC African Research Excellence Fund (AREF). He is currently the Head of Biomedical Sciences Department and Bioinformatics Research Theme leader at the University of Malawi, College of Medicine in Blantyre Malawi. His research interests are pathogenesis of Salmonella and understanding drug resistance in human pathogens



**Wellcome Genome Campus Advanced Course  
Working with Pathogen Genomes  
University of Cape Town, South Africa  
17-22 June 2018  
Participants**

**Olawale Adelowo    University of Ibadan, NIGERIA**

Molecular characterization of antibiotic resistance in bacteria from human, animal and environmental sources is my area of interest. I am particularly interested in using whole genome sequencing to trace the epidemiological relationship between antibiotic resistant bacteria from these three sources in Nigeria. I am also interested in the ecology of mobile genetic elements (plasmids and integrons) involved in dissemination of antibiotic resistance in human, animal and environmental sources in Nigeria.



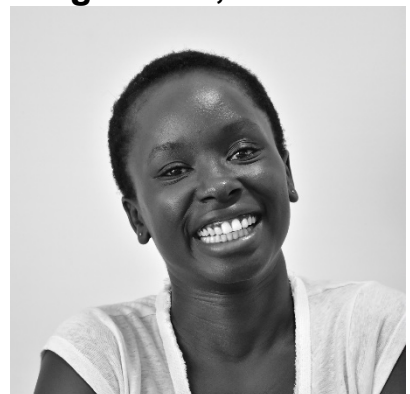
**Daniel Gyamfi Amoako    University of KwaZulu-Natal, SOUTH AFRICA**

My current research interest includes the genomic insights into multidrug resistance (MDR) pathogens. This involves the use of whole genome sequencing (WGS) and bioinformatic analysis as well as other complementary approaches to ascertain the microbial structure, current burden, molecular epidemiology, resistance mechanisms, novel mutations, gene expression levels, mobile genetic support, virulence factors and evolutionary relationship of MDR pathogens in KZN, South Africa. Hence this course will introduce me to the current bioinformatic techniques and tools that will help build my interest in pathogenomics.



**Anne Amulele    KEMRI Wellcome Trust Research Programme, KENYA**

I am a PhD fellow with Open University and my current research interest is antimicrobial resistance. My study aims to understand the relative contributions of resistance mechanisms towards the fitness and pathogenicity of Gram negative bacteria isolated in Kenya, which will increase our understanding of the effect of antibiotic selective pressure on resistance. I intend to use whole genome sequencing to identify resistance and pathogenicity genes, and to explore the molecular epidemiology of the strains isolated as part of my project. This course will equip me with the necessary skills to analyse bacterial genomes, a key component of my research.





**Wellcome Genome Campus Advanced Course  
Working with Pathogen Genomes  
University of Cape Town, South Africa  
17-22 June 2018  
Participants**

**Jonathan Asante University of KwaZulu Natal, SOUTH AFRICA**

Prevalence and mechanisms of multidrug-resistance among *Klebsiella pneumoniae*, *Acinetobacter baumannii* and *Pseudomonas aeruginosa* clinical and environmental isolates. Antibiotic-resistant bacteria and resistance genes are increasingly being detected in the clinical settings as well as in the environment. The proposed study seeks to ascertain the prevalence and mechanisms of multi-drug resistance in clinical and environmental isoates of *Klebsiella pneumoniae*, *Acinetobacter baumannii* and *Pseudomonas aeruginosa*. Whole genome sequencing will be used to determine resistance determinants and relatedness of isolates. This conference is expected to equip me with relevant skills regarding working with the genomes of pathogens, including the analysis of genome sequences, which would help facilitate the interpretation of my results.



**Prince Asare Noguchi Memorial Institute for Medical Research, GHANA**

In Ghana, the transmission pattern of the different circulating tuberculosis (TB) strains has not been critically assessed. Molecular and genomic understanding (using comparative whole genome sequence analysis) of the circulating strains is key for the epidemiological tracking and control of TB disease. My current research project aims to understand transmission dynamics and identify risk factors for recent TB transmission in Ghana. The content of this course is ideal and will broaden my understanding, and provide adequate knowledge and skills needed to help analyze the generated genomic data which inevitably will provide vital information to aid the national tuberculosis control program.



**Larson Boundenga CIRMF, GABON**

Dr Larson Boundenga is a researcher at the International Center of Medical Research of Franceville (CIRMF), where he works within the Evolution and Inter-species Transmission of Pathogens group. He received a Ph.D. degree in ecology and evolution of pathogens from Cheikh Anta Diop University (Senegal) in 2015, and his research relate to the characterization of diversity, ecology and pathogens evolution of wild fauna, especially malaria parasites. Actually, his research interests focus in the determination and analyzing of the full genomes of some parasitic lineages of wild animals. Therefore, the WGCAC course is a relevant opportunity to get knowledge of genomic analyses for his current and ongoing works on "pathogens evolution".



**Wellcome Genome Campus Advanced Course  
Working with Pathogen Genomes  
University of Cape Town, South Africa  
17-22 June 2018  
Participants**

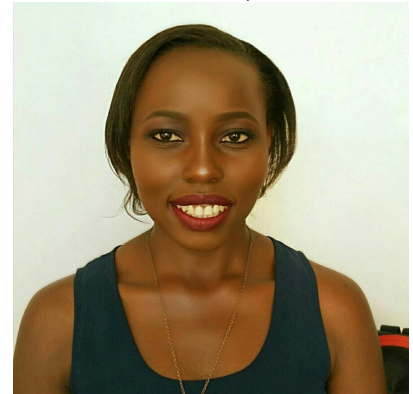
**Melissa Chengalroyen    MMRU, SOUTH AFRICA**

The project I am involved in proposes to investigate the deletion of a mismatch repair gene in *M. tuberculosis*, proposed to play an imperative role in the maintenance of genomic stability in this pathogen. This will be achieved by assessing the effect of its disruption on genome mutations, which we hypothesize to be enhanced. To gain a deeper understanding, whole genome sequencing will be performed on select isolates and SNPs identified to establish which genes and consequently pathways are susceptible to mutations and infer whether augmented genomic diversity could be an adaptive strategy in clinical isolates. This course will allow us to perform the genome assembly, analyze the raw sequencing data and identify SNPs in the resulting hypermutable strains and compare these to clinical isolates.



**Marine Chepkwony    International Livestock Research Institute, KENYA**

I am a PhD Graduate Fellow in Molecular Epidemiology undertaking my research within the CTLGH program. I am investigating the binding and entry of *Theileria parva* sporozoites into bovine lymphocytes focusing on identifying which genes are responsible for tolerant phenotypes observed in a specific cattle lineage. This work will involve querying the function of several lymphocyte and *T. parva* sporozoite surface proteins for their role in the infectivity process. Both parasite and host cells will be sequenced at different stages of infectivity and analyzed. This workshop is the ideal opportunity for me to learn the ropes of working with pathogen genomes.



**Brigitta Derendinger    Stellenbosch University, SOUTH AFRICA**

Currently I am registered as a Masters student at Stellenbosch University in the Division of Molecular Biology and Human Genetics and am planning to upgrade to a PhD in 2019. My project involves studying tuberculosis patients receiving bedaquiline (BDQ) and/or delamanid (DLM) in their drug regimens and the emergence of resistance to these relatively new anti tuberculosis drugs. I am investigating the frequency of known and novel genetic variants associated with different levels of phenotypic susceptibility to BDQ and DLM. This "Working with Pathogen Genomes" course will be valuable for my current research and postgraduate studies.



**Wellcome Genome Campus Advanced Course  
Working with Pathogen Genomes  
University of Cape Town, South Africa  
17-22 June 2018  
Participants**

**Abla Djebbar    University Hassiba Benbouali of Chlef, ALGERIA**

I am currently a PhD student at the University of Algiers, working on a project that focuses on the prevalence and molecular characterisation of isolates of the nosocomial pathogen, *Clostridium difficile*, in Algeria.

I am also a teaching assistant of a bioinformatic practical course for first and second year Masters students.

By attending this course, I will strength further my basic theoretical and practical skills in computational biology; which are necessary to conduct more efficiently my current and future research projects, and to contribute to the efforts in building a much needed bioinformatics capacity in Algeria.



**Mohammed Ahmed Elnour    National University Research Institute, SUDAN**

Prior to my current position at NURI, I worked for 10 years at the Tropical Medicine Research Institute (TMRI), as a researcher for different Neglected Tropical Diseases. My research focuses on Molecular Parasitology and Bioinformatics mainly parasites/pathogens and vectors of tropical diseases, population genetics, Parasites genomics, genetics competence of disease vectors to arboviral diseases. Currently I am working in the Genome Sequence of Methicillin-Resistant *Staphylococcus aureus* Strain SO-1977, Sudan. I am actively involved in comparative genomics, Proteomics and functional genomics. I believe that the course will be very useful because it will provide me with genomic analysis software (s), handling and processing of genomic DNA sequences, genome assembly...etc.



**Temitope Ekundayo    University of Fort Hare, SOUTH AFRICA**

I am Ekundayo Temitope Cyrus, a doctoral student of the University of Fort Hare. My ongoing research deals with Prevalence and Virulence Profiling of *Plesiomonas shigelloides* in selected freshwater resources in the Eastern Cape Province, South Africa. *P. shigelloides* is an emerging waterborne pathogen. The study aims at determining strain-specific molecular signatures of *P. shigelloides* with intention to understand diagnostic labels

necessary for differentiating pathogenic strains from non-pathogenic ones. In the future, the study hopes to development diagnostic protocols to enhance its infection managements. WGCAC Working with Pathogen Genomes is a promising platform to acquire more hand-ons skills.





**Wellcome Genome Campus Advanced Course  
Working with Pathogen Genomes  
University of Cape Town, South Africa  
17-22 June 2018  
Participants**

**Emmanuel Eze    University of KwaZulu-Natal, SOUTH AFRICA**

New generation sequencing has proved to be a high through-put technique for the study of genome-wide variation and differentially expressed genes between bacteria species. RNA-seq has been successfully used to compare the transcriptome and resistome profiles of resistant bacterial species. Acinetobacter is an important nosocomial pathogen exhibiting multidrug resistance and persistence. In my current project, I intend to compare the genomes and transcriptomes of Acinetobacter baumannii strains using WGS and RNA-seq in order to identify differentially expressed genes associated with the biofilm phenotype and their correlation with antimicrobial resistance phenotypes.



**Arash Iranzadeh    University of Cape Town, SOUTH AFRICA**

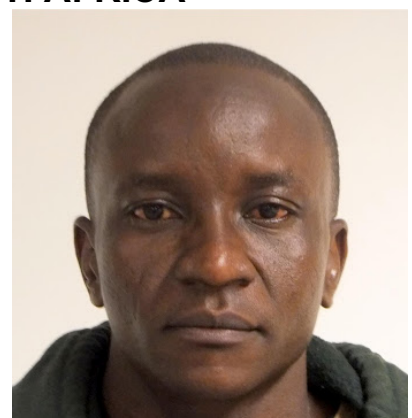
My current research is about a pan-genome wide association study to identify genes associated with invasive Pneumococcus. Streptococcus pneumoniae or Pneumococcus is a commensal human pathogen that is harmlessly carried in upper respiratory tract without causing any pathogenic symptoms. However, it sometimes invades other internal organs such as lung, nervous system and blood and cause dangerous diseases. Therefore, this bacterium can exist in two forms: invasive and non-invasive. The hypothesis is that there should be some genetic differences between invasive isolates and non-invasive isolates. The main goal of my project is to find these genetic differences specially genes responsible for the pathogenic behaviour of the Pneumococcus.



**Chacha Issarow    University of Cape Town, SOUTH AFRICA**

I am a Postdoctoral Research Fellow in Bioinformatics at the University of Cape Town (UCT), South Africa. Currently, I am working on Next Generation Sequencing project based on Whole Genome Sequencing technology to determine mutations and evolution of Mycobacterium tuberculosis (MTB) strains associated with drug-resistant TB transmission.

I am interested in Next Generation Sequencing based on Whole Genome Sequencing, Statistical Analysis, Infectious Diseases Epidemiology, Big Data Analysis and Visualisation.



**Wellcome Genome Campus Advanced Course  
Working with Pathogen Genomes  
University of Cape Town, South Africa  
17-22 June 2018  
Participants**

**Mary Kivata US Army Medical Research Directorate, KENYA**

My current research is on molecular characterization of AMR genes in *Neisseria gonorrhoeae* (GC) isolates from Kenya through whole genome sequencing. Among my specific objectives is; to determine the origin, evolution, spread and genetic relatedness of Kenyan GC; to identify, characterize, and compare expressed AMR determinants with known ones and determine molecular mechanisms involved in AMR development in the Kenyan GC. This course will equip me with knowledge on; Linux OS, genome assembly, mapping Illumina read against a reference genome, SNP calling and Genome-wide phylogenetic analysis, Genome annotation and Artemis Comparison Tool which will enable me to analyze my data.



**Meriem Laamarti Mohamed V university, MOROCCO**

In Morocco, *Mycobacterium tuberculosis* is a major public health problem with a relatively high incidence. thus a better understanding of TB profile could help to identify risk settings. we aim to apply the WGS to analyze *Mycobacterium tuberculosis* (MDR - Pre XDR) isolates obtained from different part of Morocco to predict unexpected resistance associated mutations, study of the dissemination, virulence and plasticity mechanism, and finally identification of the sublineages circulating in Morocco from the WGS data. Which fits perfectly the objective of the course that will allow me to better analyze the genomes and get insight from experts.



**Tapfumanei Mashe National Microbiology Reference Laboratory, ZIMBABWE**

I am a young researcher who prides himself as an emerging scientist and catalyst with untamed interest in antimicrobial resistance, infectious diseases and molecular epidemiology, using one health approach. I have done studies on antimicrobial resistance of *Salmonella* species, *Shigella* species and *Cholera* species in Zimbabwe. Currently I am working on a PhD entitled, "Genomic characterisation of *Salmonella enterica* from environmental, animal, food and human sources in Zimbabwe using whole genome sequencing. I will sequence 600 *Salmonella enterica* isolates and this course will be of great importance in impacting technical skills for me to analyse the sequences.



**Wellcome Genome Campus Advanced Course  
Working with Pathogen Genomes  
University of Cape Town, South Africa  
17-22 June 2018  
Participants**

**Haddijatou Mbye    University of Ghana, GHANA**

Haddijatou Mbye is one of the pioneer MRC Unit The Gambia at LSHTM PhD fellows funded by the Wellcome Trust- DELTAS (Developing Excellence in Leadership Training and Science) project at the West African Centre for Cell Biology of Infectious Pathogens (WACCBIP), University of Ghana. Her PhD research is focused on mechanisms of antimalarial drug resistance in African Plasmodium falciparum populations. As a part of her PhD, she is involved in developing a new deployable phenotypic assay suitable for drug sensitivity testing in remote resource poor settings. Her goal is to continue to build a scientific career and contribute towards the improvement of health and wellbeing in developing countries.



**Estelle Mewamba Mezajou    University of Dschang, CAMEROON**

I am a PhD research student and my current research study is focused on the epidemiology of drug resistance in African Animal Trypanosomiasis in Cameroon. For this we shall be using molecular, genomic and transcriptomic approaches in order to map trypanocidal resistance and to understand genetic and epigenetic bases involved in the development of drug resistance. To achieve this, comparative genomics and transcriptomics investigations between trypanosomes isolates (resistant/sensitive to trypanocides) are needed. Therefore, this training will be an opportunity for me to build my capacity in comparative genomics and in transcriptome analysis that are necessary for my study.



**Camus Nimmo    Africa Health Research Institute, SOUTH AFRICA**

I am a second year PhD student working at the Africa Health Research Institute in Durban and UCL in London. I am investigating the role of Mycobacterium tuberculosis genetic diversity within individual patients and its role in the development of further drug resistance and clinical outcomes. For this study I am enrolling a cohort of patients with drug-resistant tuberculosis and whole genome sequencing mycobacteria cultured from sputum samples. I would like to learn how to identify, measure and compare within-host bacterial diversity from next gen WGS data.





**Wellcome Genome Campus Advanced Course  
Working with Pathogen Genomes  
University of Cape Town, South Africa  
17-22 June 2018  
Participants**

**Stefan Opperman NHLS, SOUTH AFRICA**

This is a descriptive and molecular epidemiology study to investigate a community-associated outbreak of wild-type (pan-drug-susceptible) *Pseudomonas aeruginosa* in the Cape Metropolitan area during a drought period. We explore a possible link between an increase in wild-type *P. aeruginosa* infection and the water infrastructure system, using ArcGis geographical mapping software and environmental water sampling done by the Department of Water and Sanitation. Whole genome sequencing is implemented to compare stored isolates from blood cultures before, during and after the outbreak period in an effort to identify hyper-virulent strains and/or clonal variability in a time of water stress.



**Eniyou Oriero Medical Research Council Unit, The Gambia**

Infectious diseases like malaria still constitute a global health challenge. Successful control and elimination of malaria in sub-Saharan Africa would have a significant impact towards achieving the sustainable development goals. To achieve my long-term career goal of becoming an independent scientist capable of delivering world-class research in Africa, I am currently carrying out a postdoctoral project focussed on generating genomic data on mixed plasmodium species infections. I am therefore interested in exploring available training opportunities to develop and improve my research and bioinformatics skills, in order to maximize my creative and intellectual abilities to develop innovative and competitive research proposals.



**Benjamin Pokam Thumamo University of Buea, CAMEROON**

My research interest has focused on the molecular epidemiology of tuberculosis (TB) in Africa. Currently, my study on the genomic and transcriptomic analysis of *Mycobacterium africanum* (MAF) lineages compared to *Mycobacterium tuberculosis* (MTB) in West/Central Africa aims to determine the genomic diversity and the variation in the transcriptome expression between the two members. The difference in gene expression and adaptation of MAF lineages compared to MTB might help to understand the pathogenicity/virulence processes of the different lineages/sublineages of the MTB complex. This course thus offers the opportunity to learn and acquire the skills necessary for data analysis and interpretation.



**Wellcome Genome Campus Advanced Course  
Working with Pathogen Genomes  
University of Cape Town, South Africa  
17-22 June 2018  
Participants**

**Yolandi Snyman Stellenbosch University, SOUTH AFRICA**

My project characterises the colistin resistance in Gram-negative bacterial pathogens in the Western Cape of South Africa. The plasmid-mediated colistin resistance *mcr-1* gene was identified in the majority of *E. coli* and *K. pneumoniae* isolates in our setting. In addition, colistin resistant isolates that are *mcr-1* negative suggests novel resistance mechanisms. WGS of both colistin resistant and susceptible isolates would form a major component of elucidating resistance mechanisms, would allow us to compare strain lineages among susceptible and resistant isolates, and to evaluate the presence of virulence genes.





# Module 0: Virtual Machine

## Working with Pathogen Genomes

### What is an Operating System?

- Software that supports the computer's basic functions
- Manages computer hardware (screen, mouse, keyboard)
- Provides tools for managing files, running software
- Provides a way via software applications to interact with the computer

Examples include:

- **Windows**
- **OSX**
- **Linux**



### What is a virtual machine?

A virtual machine (VM) is a software computer that, like a physical computer, runs an operating system and a set of software applications. It allows you to run one operating system (e.g. Linux) within another operating system (e.g. Windows).

The VM produced for this course already includes the data and software you'll need for the exercises.

### Why use a VM?

The virtual machines can be built once and then run in a range of different environments without problems. This makes it a lot easier for you to reproduce what you've learnt on this course after you've left as well as making it easier for us to jump straight into teaching you new things without lots of setup.

A lot of bioinformatics software is built to run on Linux, some of this runs on OSX and a bit runs on Windows. This is because many high performance clusters run variants of Linux. We therefore want to teach in a similar environment.

### Adding the virtual machine to Oracle VirtualBox

On this course we will be running a VM based on [BioLinux](#) which is based on a distribution of Linux called Ubuntu. We periodically provide [updates for the VM](#) on our website but you should use the supplied copy for now.

We will be running the VM on a Windows machine using the VirtualBox software. Other software is available to run the VM but this is a good free alternative. The following instructions walk you through how to setup the VM.

**IMPORTANT NOTE: There are small differences between the images below, the instructions and what you'll see on the computer. If there's any doubt, follow the instructions in the text rather than the images.**

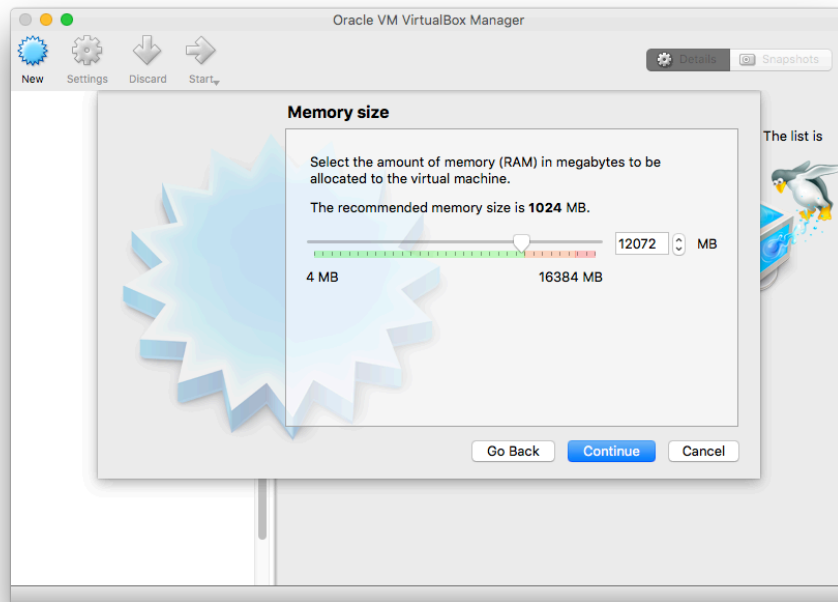
Insert the USB stick into your machine. Our sticks will use USB 3 and run faster if you **plug the stick into a blue port**. Then open **Oracle VirtualBox** on your machine and select **'New'**. On these machines you'll find VirtualBox under the Windows Start Menu.



Enter a name for your VM, **'Vietnam\_2017'**. Select **'Linux'** as the type of operating system that will run on your VM. Select the version as **'Ubuntu (64bit)'** and click **Continue**.



Select the amount of memory (RAM) to allocate to this VM, the values available to you will depend on the amount of memory (RAM) available on the host machine. On both our machines and your machine at home move the slider to the top of the green section.



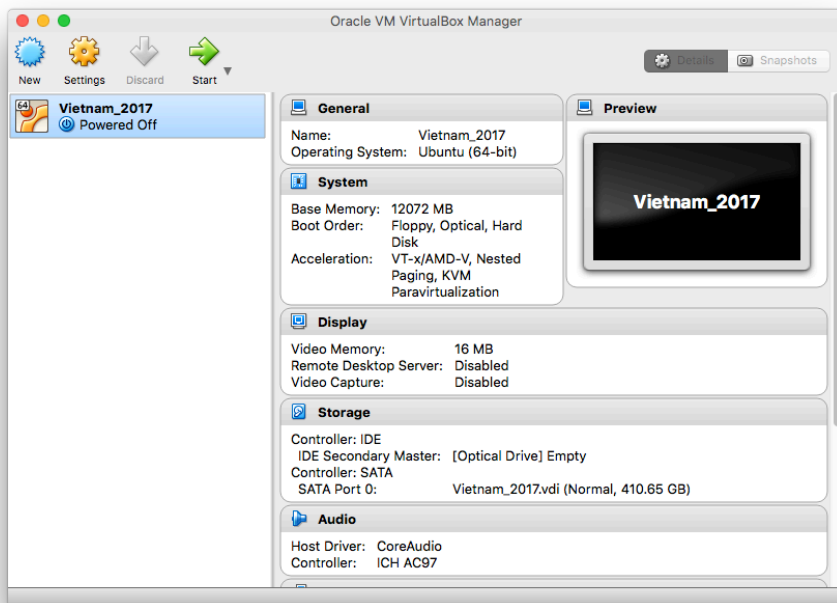
As we have already created the virtual hard disk for you, select ***'Use an existing virtual hard disk file'*** and click on the ***folder with a green arrow*** on the right hand side.



Navigate to the USB drive and select the file **'Vietnam\_2017.vdi'** and click **'Create'**.

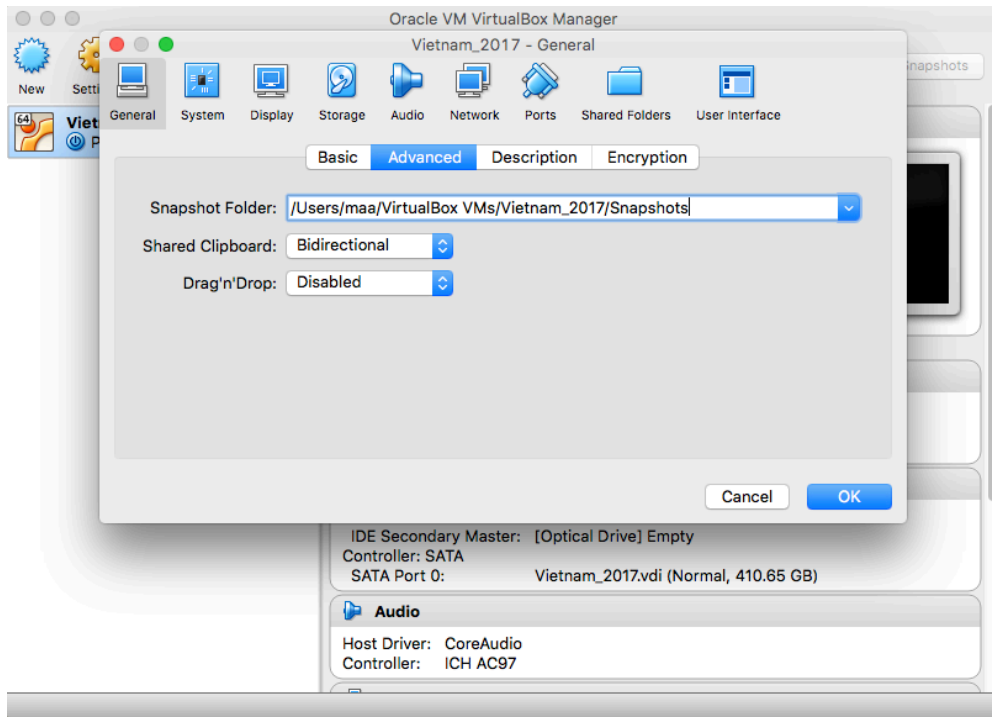


You should then see the screen below containing your VM **'Vietnam\_2017'** in the Powered Off state.

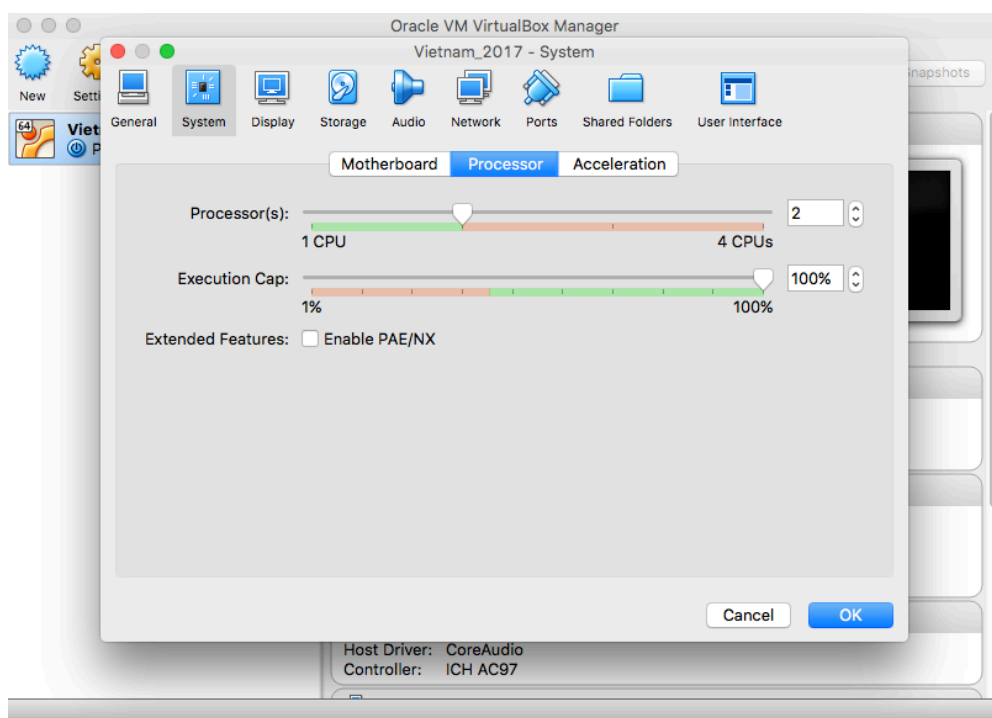


# Update Settings

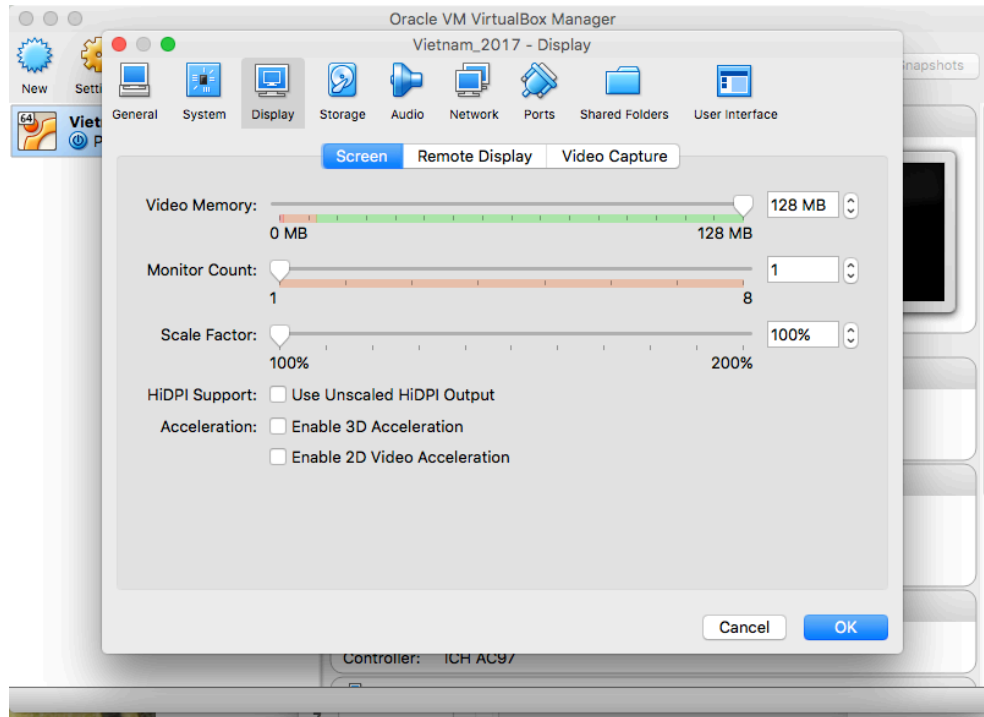
Left click on **'Vietnam\_2017'**, then click on Settings at the top of the window. Click General >> Advanced and then change "Shared Clipboard" to Bidirectional. This will make it possible to copy and paste things from the Windows "host" into and out of the Linux "guest".



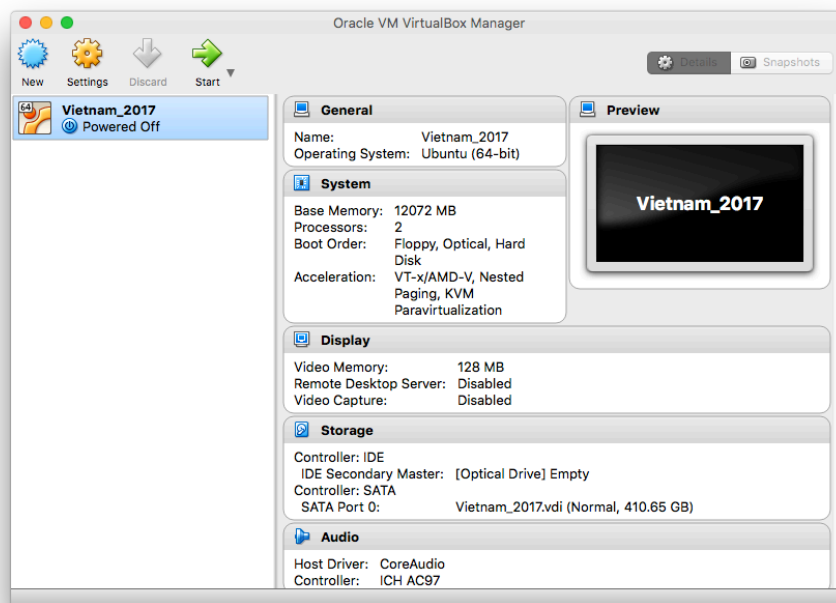
Now click **System >> Processor** and slide the slider to the top of the green section (i.e. use half the CPUs).



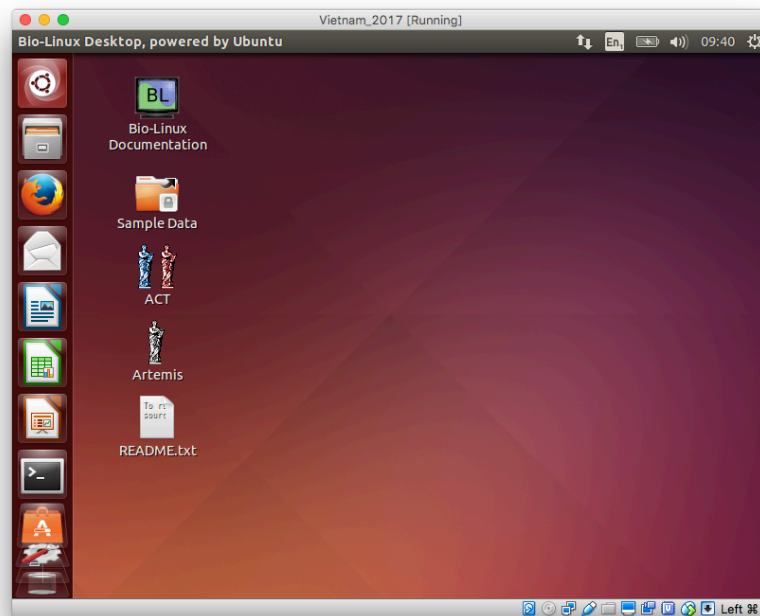
Click **Display >> Screen** and change Video Memory to the maximum amount.



Click **OK** to save these changes. You should now be able to start the VM: **click on Vietnam\_2017** and then **press the Start button**.



It should then look a little like this:



You can make the VM use the entire screen by pressing the right hand ‘Ctrl’ key and ‘f’ at the same time. This should work on Windows and (probably) Linux hosts; on OSX you should press the right ‘Cmd’ key (the funny box with circles on the corners) and the ‘f’ key. You can escape by using the same key combination.

## Username and password

If you need them, the username is “manager” and the password is also “manager”.

## Using the VM at home

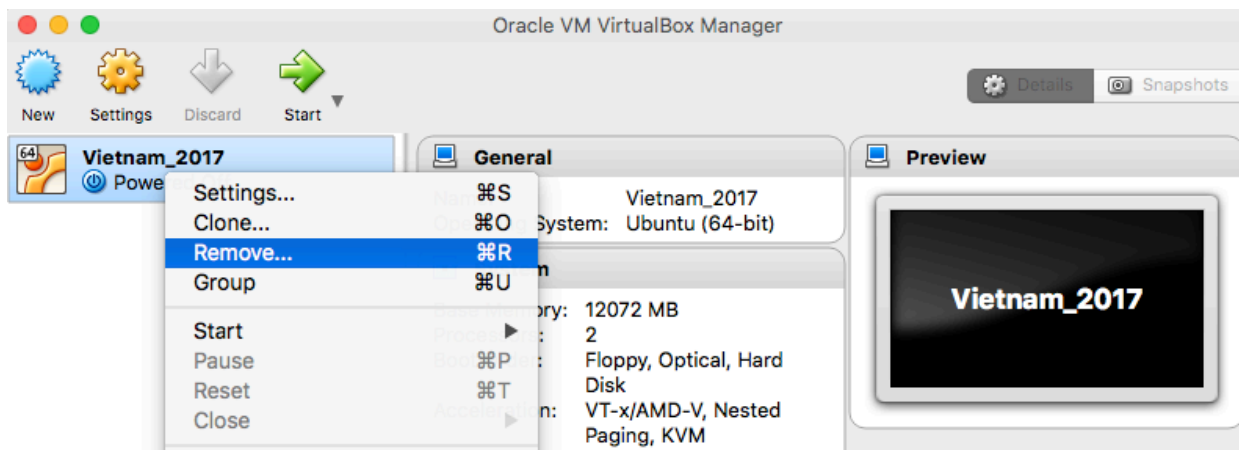
You get to keep the USB stick after the course and can use it to continue working on the exercises.

- On Mac OSX you will probably need to copy the virtual machine onto your hard disk. Copy Vietnam\_2017.vdi from the USB stick onto your hard drive and then follow these instructions again. Note, when selecting the “existing hard disk” in one of the steps, you need to point it at the copy on your hard disk.
- On Linux, you may or may not be able to run the VM from the USB stick.
- On Windows, you should be able to run the VM from the USB stick by following these instructions. You may see a slight performance benefit from moving it onto your hard disk though.

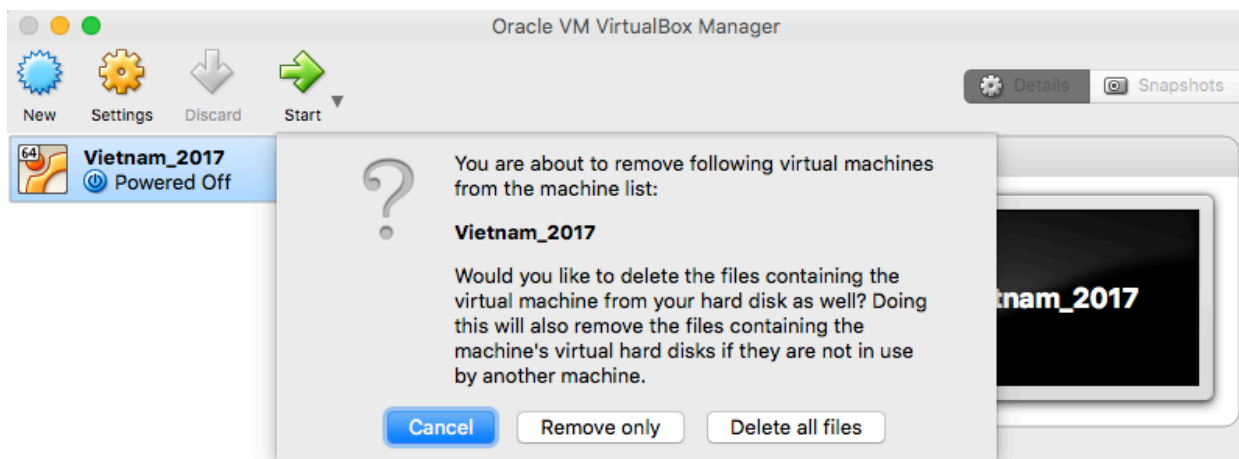
# Removing the virtual machine from VirtualBox

**DO NOT DO THIS UNTIL THE END OF THE COURSE OR UNLESS TOLD OTHERWISE**

In the main **VirtualBox** window, **right click** on your virtual machine (seen in the left hand column). Select **'Remove'** from the menu.



Then select **'Remove only'**, do not select the 'Delete all files' option as this will delete the the vdi file that contains the virtual machine.



## Shared folders

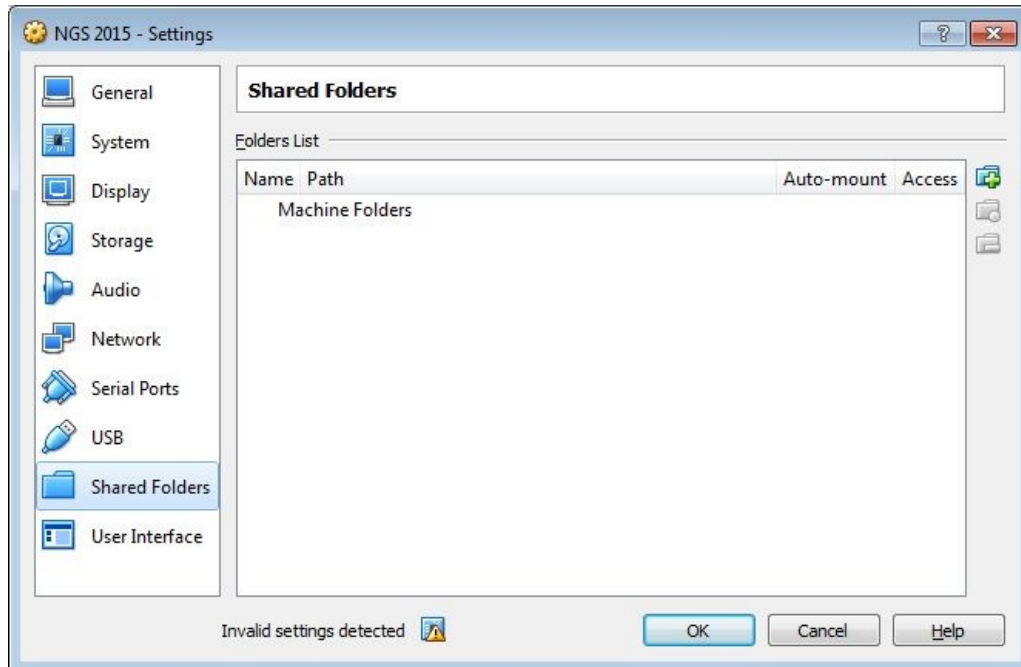
**This is an optional step which you can do at home. It is not needed as part of the course.**

By default, none of the files on the host machine are visible from inside the virtual machine; this describes how to share data between the host and the virtual machine. This might be

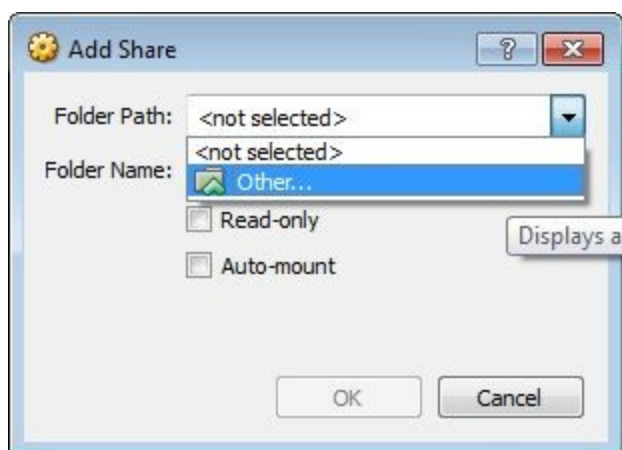


useful if you want to rerun some analysis with your own data or you want to share your results with a colleague.

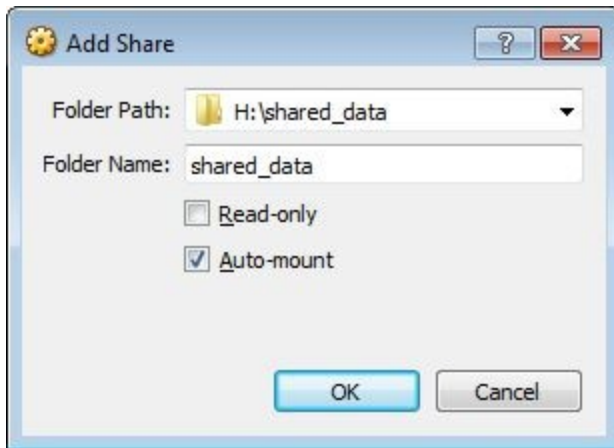
First, **shut down the virtual machine**. In the main **VirtualBox** window, **select your virtual machine** (on the left). Then go to **Settings >> Shared Folders**.



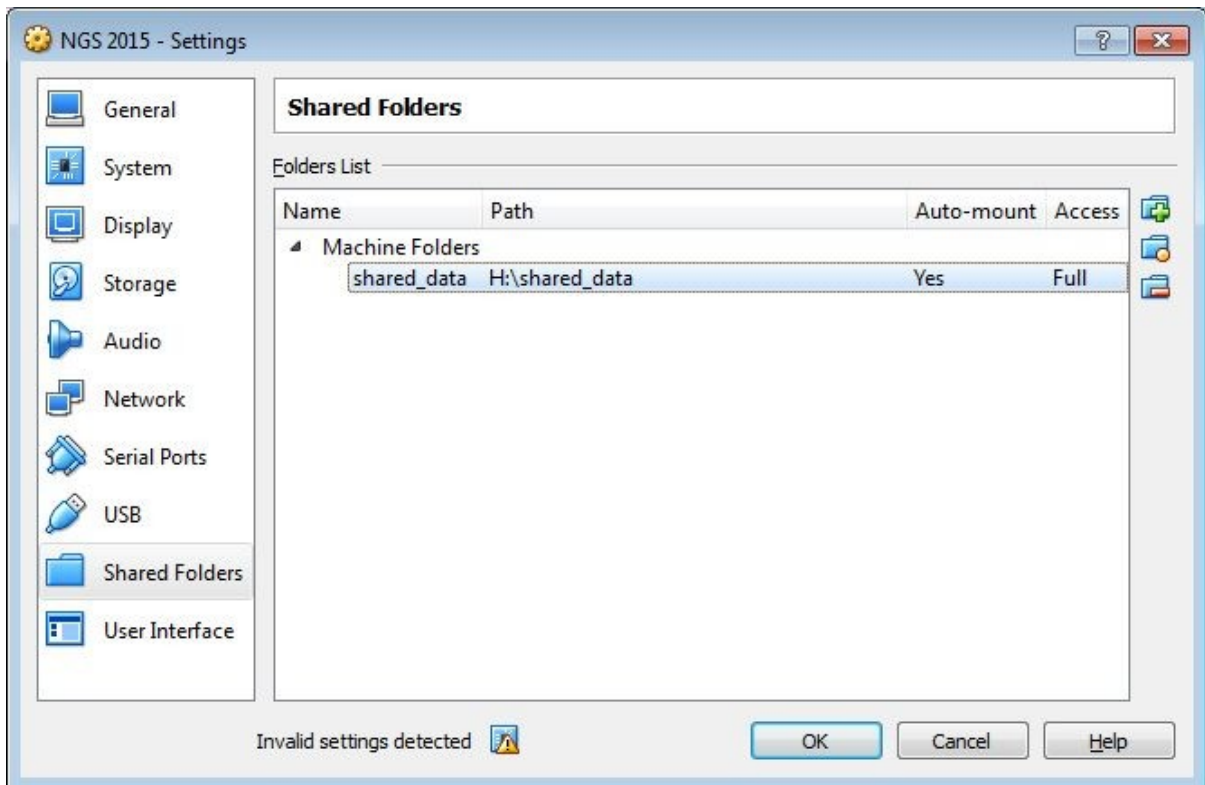
**Click on the blue folder** with a green plus on the right hand side. In the pop up box, **select "Other"** from the Folder Path option.



**Find the folder** that you would like to share with the Virtual Machine. It should now be visible in the Folder Path box. **Tick the Automount box**.



Now **click on OK**. The new shared folder should be in the Folders List:



**Select OK** and you will return to the main VirtualBox window. Now **start the virtual machine**.

### Finding the shared folder in the virtual machine

Inside the virtual machine, the shared folder will be inside the directory `/media`. It will have the same name as on the folder on the host machine, but with `sf_` added. For example, if the folder is called `shared_data` on the host, then it will be called `/media/sf_shared_data/` inside the virtual machine.

# Module 1

# Artemis

## Introduction

Artemis is a DNA viewer and annotation tool, free to download and use, written by Kim Rutherford from the Sanger Institute (Rutherford *et al.*, 2000). The program allows the user to view a range of files, from simple sequence files (e.g. fasta format) to EMBL/Genbank entries, as well as the results of sequence analyses, in a highly interactive and intuitive graphical format. Artemis is routinely used by the Infection Genomics group for annotation and analysis of both prokaryotic and eukaryotic genomes, and can also be used to visualise mapped data from next generation sequencing. Several types/sets of information can be viewed simultaneously within different contexts. For example, Artemis gives you the two views of the same genome region, so you can zoom in to inspect detailed DNA sequence motifs, and also zoom out to view local gene architecture (e.g. operons), or even an entire chromosome or genome, all within one screen. It is also possible to perform analyses within Artemis and save the output for future reference.

## Aims

The aim of this Module is for you to become familiar with the basic functions of Artemis using a series of worked examples. These examples are designed to take you through the most immediately useful functions. However, there will be time, and encouragement, for you to explore other menus; features of Artemis that are not described in the exercises in this manual, but which may be of particular interest to some users. Like all the Modules in this workshop, please remember:

IF YOU DON' T UNDERSTAND, PLEASE ASK!

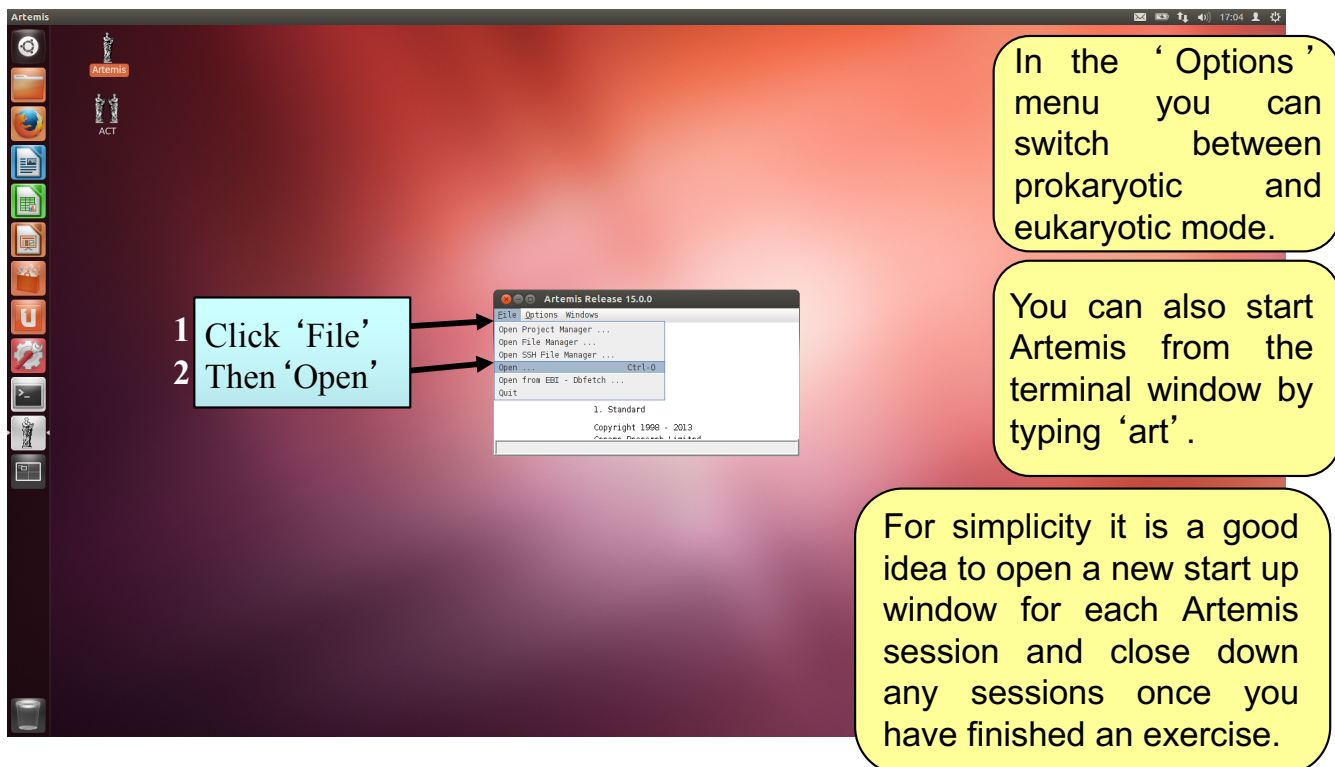
## Artemis Exercise 1

### 1. Starting up the Artemis software

Double click the Artemis icon on the desktop.

A small start-up window will appear (see below). The directory **Module\_1\_Artemis** contains all files you will need for this module.

Now follow the sequence of numbers to load up the *Salmonella* Typhi chromosome sequence. Ask a demonstrator for help if you have any problems.

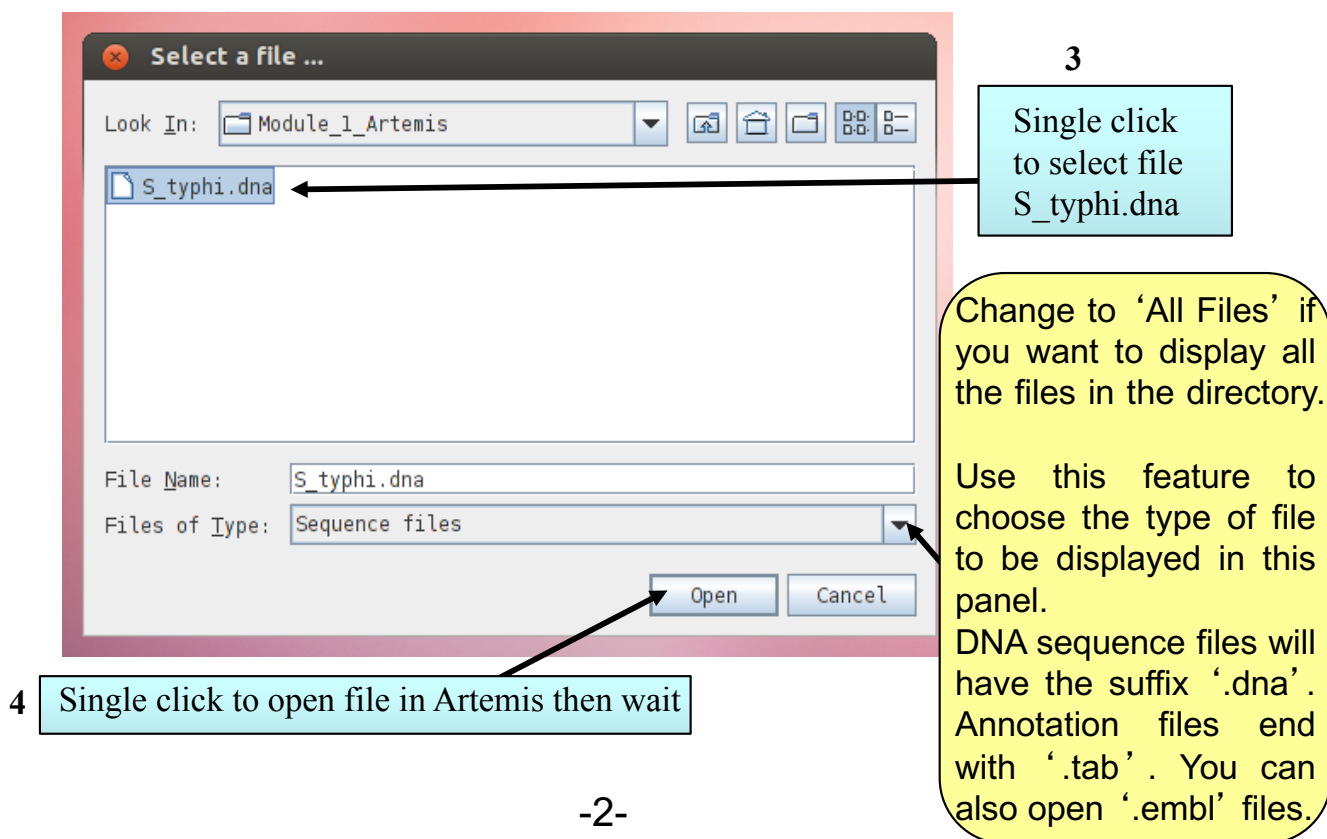


The screenshot shows the Artemis desktop environment. A window titled 'Artemis Release 15.0.0' is open, displaying a menu with options: 'File', 'Options', 'Windows', 'Open Project Manager ...', 'Open File Manager ...', 'Open SSH File Manager ...', 'Open ...', 'Open from EBI - Dofetch ...', and 'Quit'. A callout box with the number '1' points to the 'File' menu, and a callout box with the number '2' points to the 'Open' option.

In the 'Options' menu you can switch between prokaryotic and eukaryotic mode.

You can also start Artemis from the terminal window by typing 'art'.

For simplicity it is a good idea to open a new start up window for each Artemis session and close down any sessions once you have finished an exercise.



The screenshot shows the 'Select a file ...' dialog box. The 'Look In' field is set to 'Module\_1\_Artemis'. The file list contains 'S\_typhi.dna'. The 'File Name' field is 'S\_typhi.dna' and the 'Files of Type' is 'Sequence files'. The 'Open' button is highlighted.

3 Single click to select file S\_typhi.dna

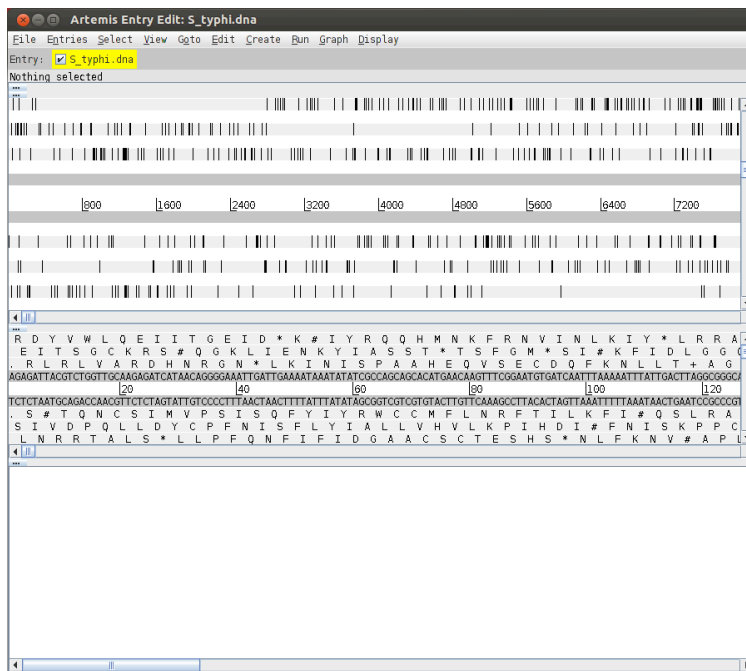
Change to 'All Files' if you want to display all the files in the directory.

Use this feature to choose the type of file to be displayed in this panel. DNA sequence files will have the suffix '.dna'. Annotation files end with '.tab'. You can also open '.embl' files.

4 Single click to open file in Artemis then wait

## 2. Loading an annotation file (entry) into Artemis

Hopefully you will now have an Artemis window like this! If not, ask a demonstrator for assistance.



Now follow the numbers to load the annotation file for the *Salmonella* Typhi chromosome.

1

Click 'File' then 'Read an Entry'

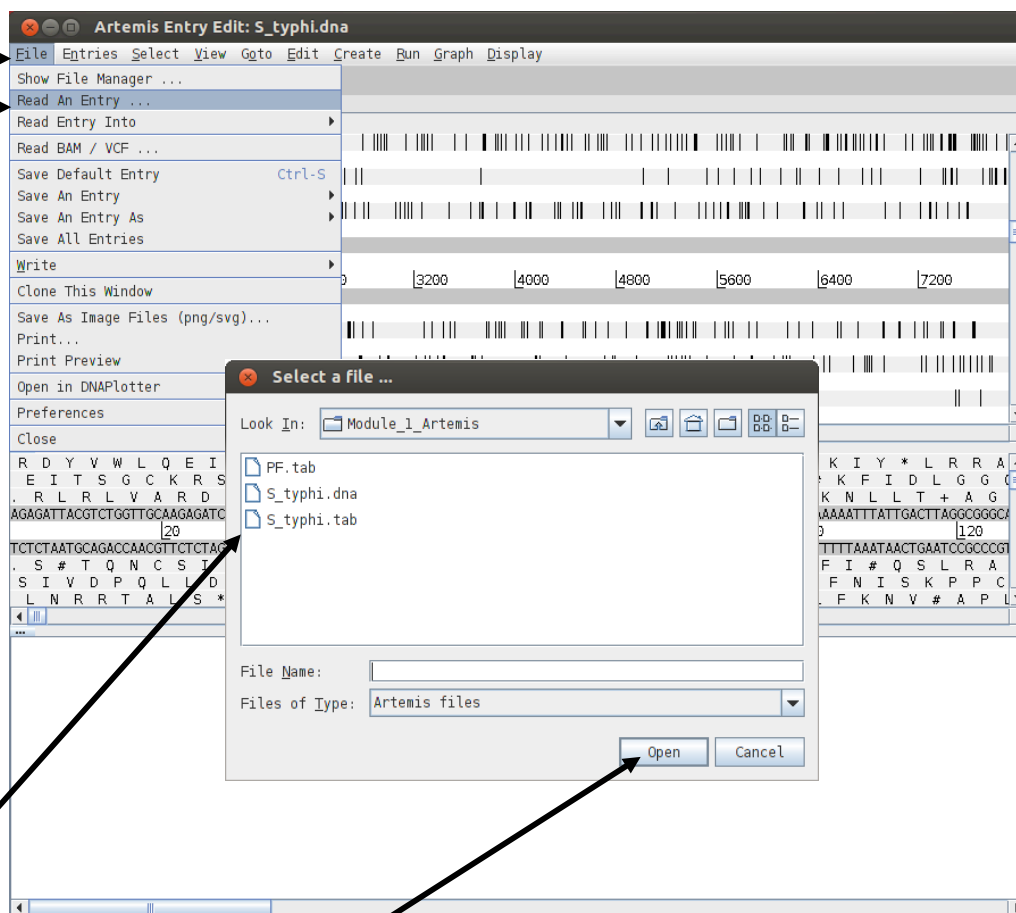
What's an "Entry"? It's a file of DNA and/or features which can be overlaid onto the sequence information displayed in the main Artemis view panel.

2

Single click to select file S\_typhi.tab

3

Single click to open file in Artemis then wait (click 'no' if an error window pops up)



### 3. The basics of Artemis

Now you have an Artemis window open let's look at what's in there.

The screenshot shows the Artemis genome browser interface. At the top, there is a menu bar and an entry field showing 'S\_typhi.dna' and 'S\_typhi.tab'. Below this, the 'Selected feature' section displays details for 'STY0004', including its coordinates, class, color, gene name, and product. The main panel shows a sequence view with forward and reverse DNA strands, reading frames, and various features represented by colored boxes. A zoomed-in view of a CDS is shown below, displaying nucleotides and amino acids. A feature list is visible at the bottom, listing various features and their coordinates.

Numbered callouts (1-8) point to specific elements in the interface:

1. Drop-down menus
2. Entry (top line)
3. Main sequence view panel
4. Zoomed-in view of a CDS
5. Feature panel
6. Sliders for zooming view panels
7. Sliders for scrolling along the DNA
8. Slider for scrolling feature list

1. **Drop-down menus:** There are lots in there so don't worry about all the details right now.
2. **Entry (top line):** shows which entries are currently loaded with the default entry highlighted in yellow (this is the entry into which newly created features are created). Selected feature: the details of a selected feature are shown here; in this case gene STY0004 (yellow box surrounded by thick black line).
3. This is the main **sequence view panel**. The central 2 grey lines represent the forward (top) and reverse (bottom) DNA strands. Above and below those are the 3 forward and 3 reverse reading frames. Stop codons are marked on the reading frames as black vertical bars. Genes and other annotated features (eg. Pfam and Prosite matches) are displayed as coloured boxes. We often refer to predicted genes as coding sequences or CDSs.
4. This panel has a similar layout to the main panel but is zoomed in to show nucleotides and amino acids. Double click on a CDS in the main view to see the zoomed view of the start of that CDS. Note that both this and the main panel can be scrolled left and right (7, below) zoomed in and out (6, below).
5. **Feature panel:** This panel contains details of the various features, listed in the order that they occur on the DNA. Any selected features are highlighted. The list can be scrolled (8, below).
6. **Sliders** for zooming view panels.
7. **Sliders** for scrolling along the DNA.
8. **Slider** for scrolling feature list.

## 4. Getting around in Artemis

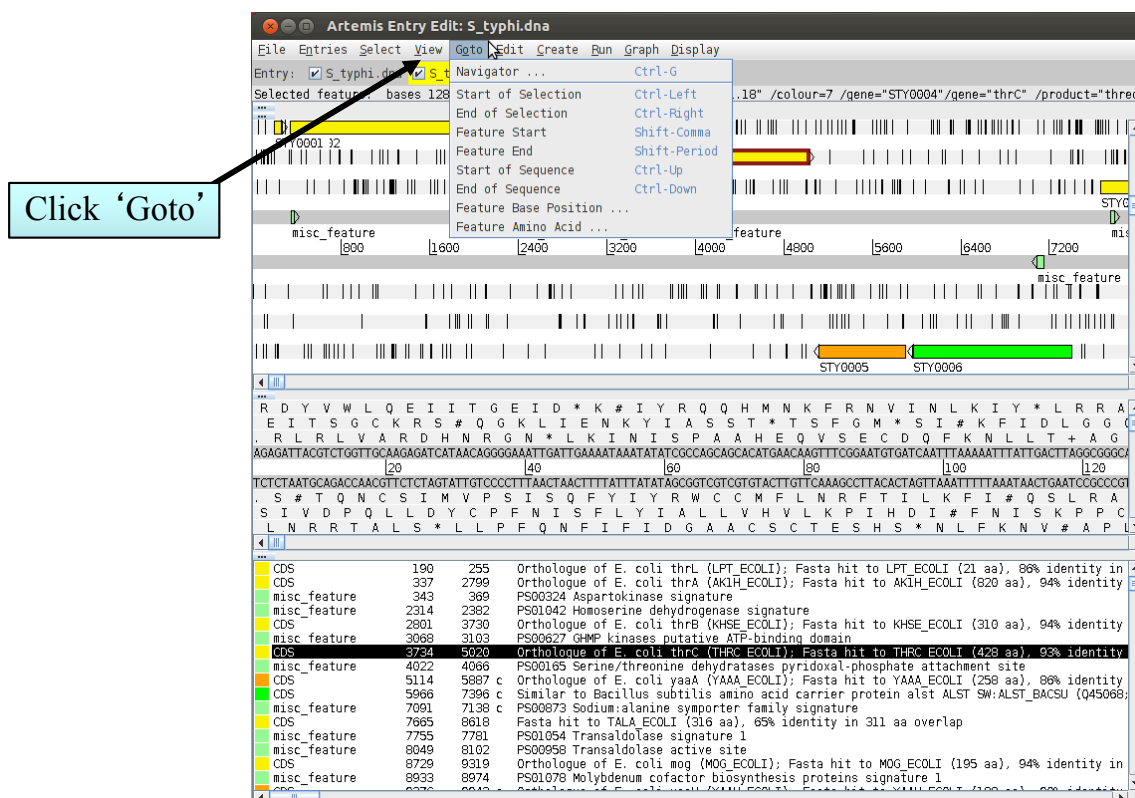
There are three main ways of getting to a particular DNA region in Artemis:

- the Goto drop-down menu
- the Navigator and
- the Feature Selector (which we will use in Part IV)

The best method depends on what you're trying to do. Knowing which one to use comes with practice.

### 4.1 The 'Goto' menu

The functions on this menu (below the Navigator option) are shortcuts for getting to locations within a selected feature or for jumping to the start or end of the DNA sequence. This is really intuitive so give it a try!



It may seem that 'Goto' 'Start of Selection' and 'Goto' 'Feature Start' do the same thing. Well they do if you have a feature selected but 'Goto' 'Start of Selection' will also work for a region which you have selected by click-dragging in the main window.

So yes, give it a try!

Suggested tasks:

1. Zoom out, select / highlight a large region of sequence by clicking the left hand button and dragging the cursor then go to the start and end of this selected region.
2. Select a CDS then go to the start and end.
3. Go to the start and end of the genome sequence.
4. Select a CDS. Within it, go to a base (nucleotide) and/or amino acid of your choice.
5. Highlight a region then, from the right click menu, select 'Zoom to Selection'.



## 4.2 Navigator

The Navigator panel is fairly intuitive so open it up and give it a try.

Click 'Goto'  
then Navigator

Check that the  
appropriate search  
button is on

The screenshot shows the Artemis genome browser interface. The top window is 'Entry Edit: S\_typhi.dna' with a 'Goto' menu open. The menu options include: Goto Base (Ctrl-G), Start of Selection (Ctrl-Left), End of Selection (Ctrl-Right), Feature Start (Shift-Comma), Feature End (Shift-Period), Start of Sequence (Ctrl-Up), End of Sequence (Ctrl-Down), Feature Base Position..., and Feature Amino Acid... The bottom window is 'Artemis Navigator', which has several search options: Goto Base (selected), Goto Feature With Gene Name, Goto Feature With This Qualifier Value, Goto Feature With This Key, Find Base Pattern, and Find Amino Acid String. There are also checkboxes for 'Start search at: beginning (or end) selection', 'Overlaps With Selection', 'Forward Strand', 'Reverse Strand', 'Search Backward', 'Ignore Case', and 'Allow Substring Matches'. At the bottom are 'Goto', 'Clear', and 'Close' buttons. The Navigator window also displays a list of genomic features with columns for coordinates and feature types.

Suggestions about where to go:

1. Think of a number between 1 and 4809037 and go to that base (notice how the cursors on the horizontal sliders move with you).
2. Your favourite gene name (it may not be there so you could try 'fts').
3. Use '**Goto Feature With This Qualifier value**' to search the contents of all qualifiers for a particular term. For example using the word 'pseudogene' will take you to the next feature with the word 'pseudogene' in any of its qualifiers. Note how repeated clicking of the 'Goto' button takes you to the following pseudogene in the order that they occur on the chromosome.
4. Look at **Appendix VIII** which is a functional classification scheme used for the annotation of *S. Typhi*. Each CDS has a class qualifier best describing its function. Use the '**Goto Feature With This Qualifier value**' search to look for CDSs belonging to a class of interest by searching with the appropriate class values.
5. tRNA genes. Type 'tRNA' in the '**Goto Feature With This Key**'.
6. Regulator-binding DNA consensus sequence (real or made up!). Note that degenerate base values can be used (**Appendix Xa**).
7. Amino acid consensus sequences (real or made up!). You can use 'X' s. Note that it searches all six reading frames regardless of whether the amino acids are encoded or not.

What are Keys and Qualifiers? See **Appendix IV**



Clearly there are many more features of Artemis which we will not have time to explain in detail. Before getting on with this next section it might be worth browsing the menus. Hopefully you will find most of them easy to understand.

## Artemis Exercise 2

This part of the exercise uses the files and data you already have loaded into Artemis from Part I. By a method of your choice go to the region from bases 2188349 to 2199512 on the DNA sequence. This region is bordered by the *fbaB* gene which codes for fructose-bisphosphate aldolase. You can use the Navigator function discussed previously to get there. The region you arrive at should look similar to that shown below (maybe you have to use the zoom sliders).

The screenshot shows the Artemis genome browser interface. The main display area shows a DNA sequence with various features represented by colored bars and labels. Two yellow callout boxes point to 'CDS features' (green bars) and 'Misc features' (orange and blue bars). Below the sequence, a table lists features with their coordinates and descriptions.

Feature Type	Start	End	Description
CDS	190	255	Orthologue of E. coli thrL (LPT_ECOLI); Fasta hit to LPT_ECOLI (21 aa), 86% identity in 21 aa overlap
CDS	337	2799	Orthologue of E. coli thrA (AKIH_ECOLI); Fasta hit to AKIH_ECOLI (820 aa), 94% identity in 820 aa overlap
misc_feature	343	369	PS00324 Aspartokinase signature
misc_feature	2314	2382	PS01042 Homoserine dehydrogenase signature
CDS	2801	3730	Orthologue of E. coli thrB (KHSE_ECOLI); Fasta hit to KHSE_ECOLI (310 aa), 94% identity in 308 aa overlap
misc_feature	3068	3103	PS00627 GHMP kinases putative ATP-binding domain
CDS	3734	5020	Orthologue of E. coli thrC (THRC_ECOLI); Fasta hit to THRC_ECOLI (428 aa), 93% identity in 428 aa overlap
misc_feature	4022	4066	PS00165 Serine/threonine dehydratases pyridoxal-phosphate attachment site
CDS	5114	5887	Orthologue of E. coli yaaA (YAAA_ECOLI); Fasta hit to YAAA_ECOLI (258 aa), 86% identity in 257 aa overlap
CDS	5966	7396	Similar to Bacillus subtilis amino acid carrier protein alst ALST 9W:ALST_BACSU (Q45068; P40743) fast
misc_feature	7091	7138	PS00873 Sodium:alanine symporter family signature
CDS	7665	8618	Fasta hit to TALA_ECOLI (316 aa), 65% identity in 311 aa overlap
misc_feature	7755	7781	PS01054 Transaldolase signature 1
misc_feature	8049	8102	PS00958 Transaldolase active site
CDS	8729	9319	Orthologue of E. coli mog (MOG_ECOLI); Fasta hit to MOG_ECOLI (195 aa), 94% identity in 192 aa overlap
misc_feature	8933	8974	PS01078 MolYbdenum cofactor biosynthesis proteins signature 1

Once you have found this region have a look at some of the information available:

### **Information to view:**

#### **Annotation**

If you click on a particular feature you can view the annotation associated with it: select a CDS feature (or any other feature) and click on the 'Edit' menu and select 'Selected Feature in Editor'. A window will appear containing all the annotation that is associated with that CDS. The format for this information is constrained by that which can be submitted to the EMBL database.

#### **Viewing amino acid or protein sequence**

Click on the 'View' menu and you will see various options for viewing the bases or amino acids of the feature you have selected, in two formats i.e. EMBL (view -> selection) or fasta (view -> bases or view -> amino acids). This can be very useful when using other programs that are not integrated into Artemis e.g. those available on the Web that require you to cut and paste sequence into them.

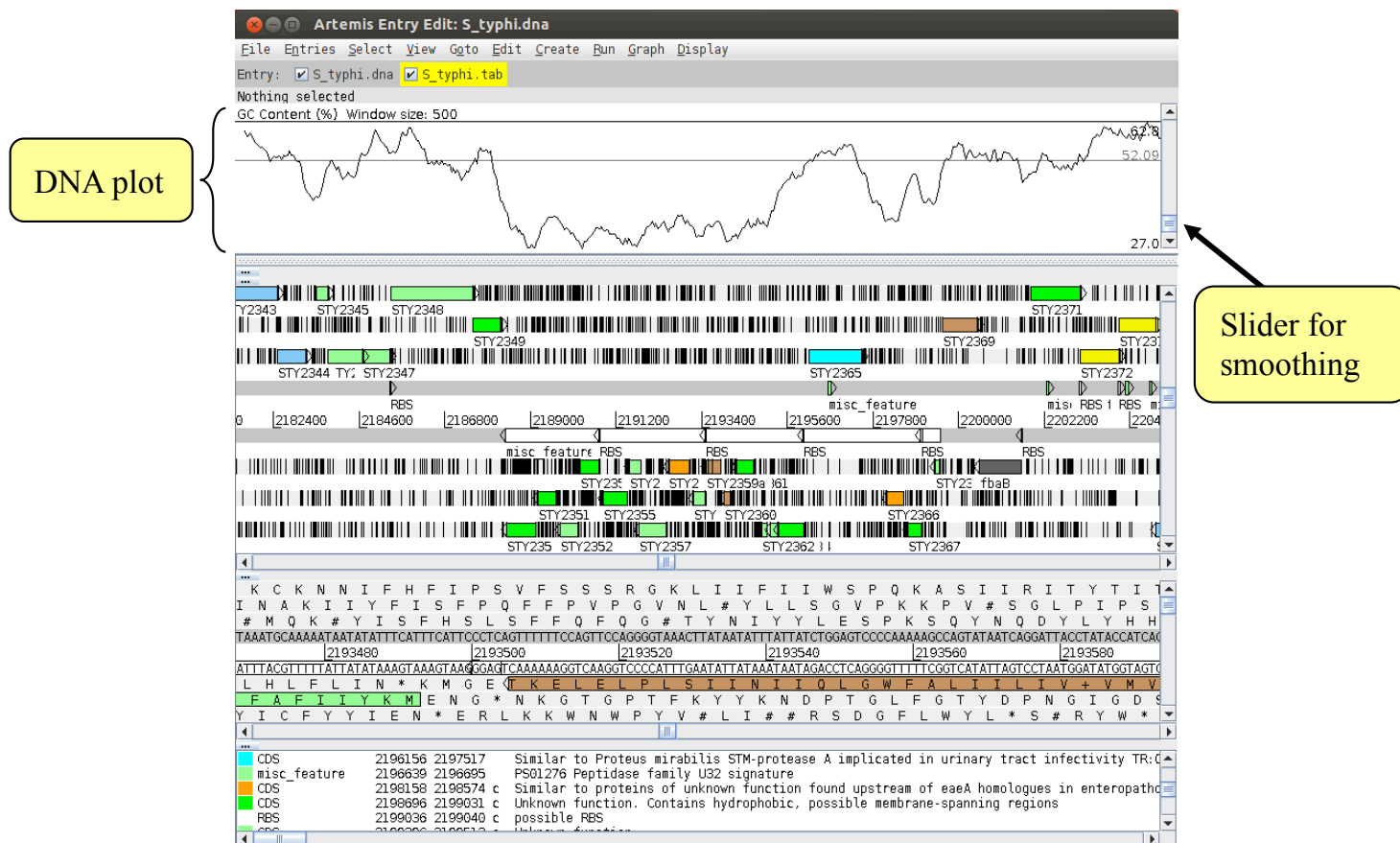
#### **Plots/Graphs**

Feature plots can be displayed by selecting a CDS feature then clicking 'View' and 'Feature Plots'. The window which appears shows plots predicting hydrophobicity, hydrophilicity and coiled-coil regions for the protein product of the selected CDS.

In addition to looking at the fine detail of the annotated features it is also possible to look at the characteristics of the DNA covering the region displayed. This can be done by adding various plots to the display, showing different characteristics of the DNA. Some of the plots can be used to look at the protein coding potential of translation frames within the DNA, and others can be used to search for horizontally acquired DNA (such as GC frame plot).

**To view the graphs:**

Click on the 'Graph' menu to see all those available and then tick the box for 'GC Content (%)'. To adjust the smoothing of the graph you change the window size over which the points on the graph are calculated, using the slider shown below.



Notice how the plot shows a marked deviation around the region you are currently looking at. To fully appreciate how anomalous this region is, move the genome view by scrolling to the left and right of this region. The apparent unusual nucleotide content of this region is indicative of laterally acquired DNA that has inserted into the genome.

As well as looking at the characteristics of small regions of the genome, it is possible to zoom out and look at the characteristics of the genome as a whole. To view the entire genome, you can use the sliders indicated above. However, be careful zooming out quickly with all the features being displayed, as this may temporarily lock up the computer. Read further to see how to zoom out.

1. To make this process faster and clearer, **switch off stop codons** by clicking with the right mouse button in the main view panel. A menu will appear with an option to de-select 'Stop Codons' (see below).
2. You will also need to temporarily **remove all of the annotated features** from the Artemis display window. In fact if you leave them on, which you can, they would be too small to see when you zoomed out to display the entire genome. To remove the annotation click on the S\_typhi.tab entry button on the grey entry line of the Artemis window shown above.

2

To de-select the annotation click here.

The screenshot shows the Artemis genome browser interface. At the top, the title bar reads "Artemis Entry Edit: S\_typhi.dna". Below the title bar, there are menu options: "File", "Entries", "Select", "View", "Goto", "Edit", "Create", "Run", "Graph", "Display". The "Entry:" section shows "S\_typhi.dna" and "S\_typhi.tab" (highlighted in yellow). Below this, it says "Nothing selected" and "GC Content (%) Window size: 500". A line graph shows GC content across the genome. Below the graph, there are several tracks of annotated features, including CDS (Coding DNA Sequences) and misc\_feature. A context menu is open over the features, with the "Stop Codons" option highlighted. A yellow callout box points to the "Stop Codons" option in the menu, stating "Menu item for de-selecting stop codons". Another yellow callout box points to the feature tracks, stating "No stop codons shown on frame lines".

Artemis Entry Edit: S\_typhi.dna  
File Entries Select View Goto Edit Create Run Graph Display  
Entry: S\_typhi.dna S\_typhi.tab  
Nothing selected  
GC Content (%) Window size: 500

Smallest Features In Front  
Set Score Cutoffs ...  
Raise Selected Features  
Lower Selected Features  
Zoom to Selection  
Select Visible Range  
Select Visible Features  
Frame Line Features ...

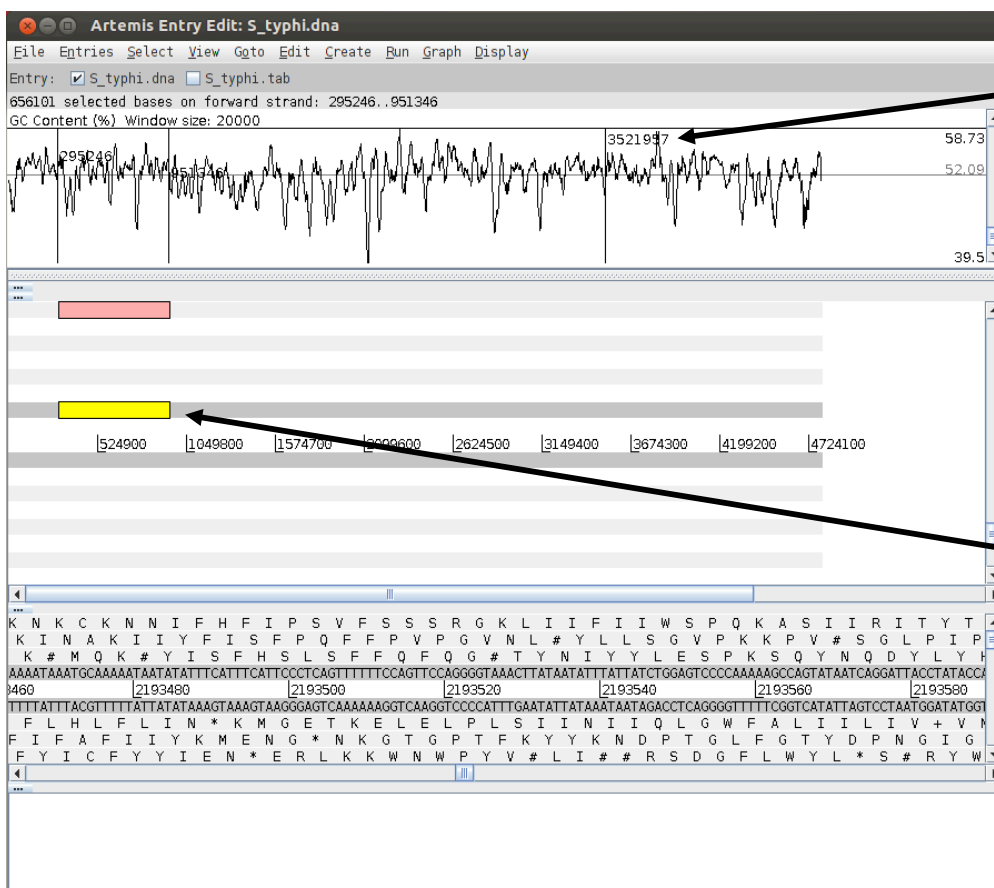
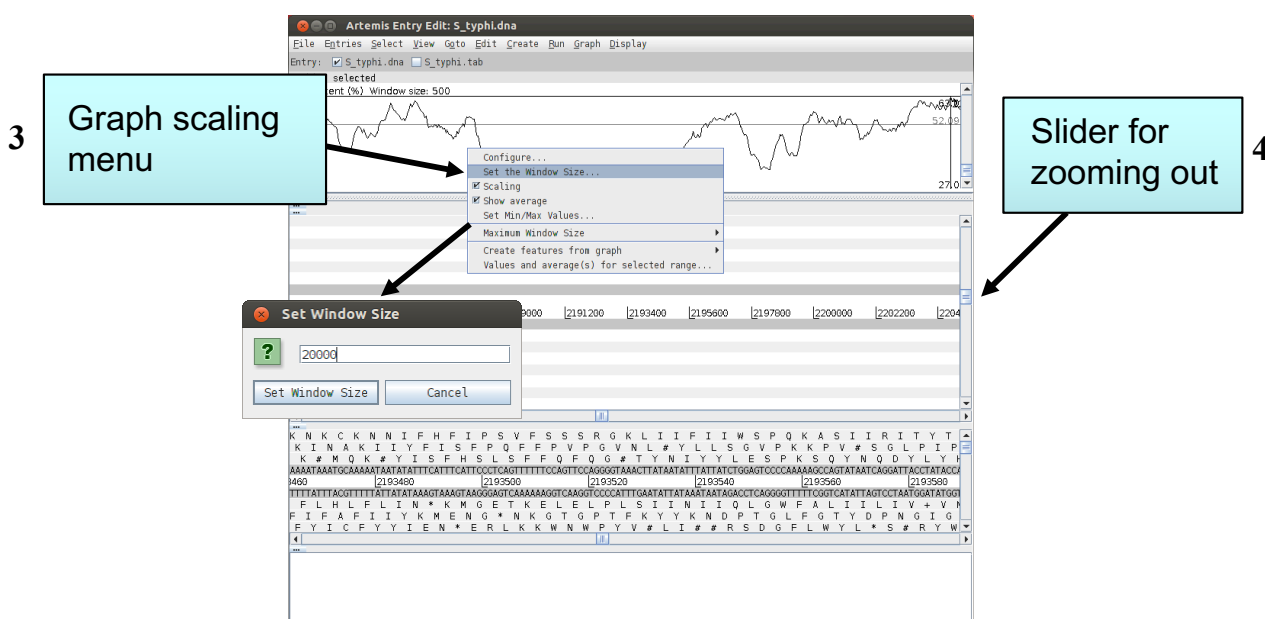
Entries  
Select  
View  
Goto  
Edit  
Create  
Write  
Run

Start Codons  
 Stop Codons  
 Feature Arrows  
 Feature Borders  
 Feature Labels  
 One Line Per Entry  
 Feature Stack View  
 Forward Frame Lines  
 Reverse Frame Lines  
 ALL Features On Frame Lines  
 Show Source Features  
 Flip Display  
 Colourise Bases

No stop codons shown on frame lines

Menu item for de-selecting stop codons

- One final tip is to **adjust the scaling** for each graph displayed before zooming out. This increases the maximum window size over which a single point for each plot is calculated. To adjust the scaling click with the right mouse button over a particular graph window. A menu will appear with an option "Set the Window size" (see above), set the window size to '20000'. You should do this for each graph displayed (if you get an error message press continue).
- You are now ready to zoom out by dragging or clicking the slider indicated below. Once you have zoomed out fully to see the entire genome you will need to adjust the smoothing of the graphs using the vertical graph sliders as before, to have a similar view to that shown below.



Click with the left mouse button in a graph window. A line and a number will appear. The number is the relative position within the genome (bps).

Click and drag to highlight a region on the main DNA line. Notice that the boundaries of this region are now marked in the graph windows that you previously clicked in.

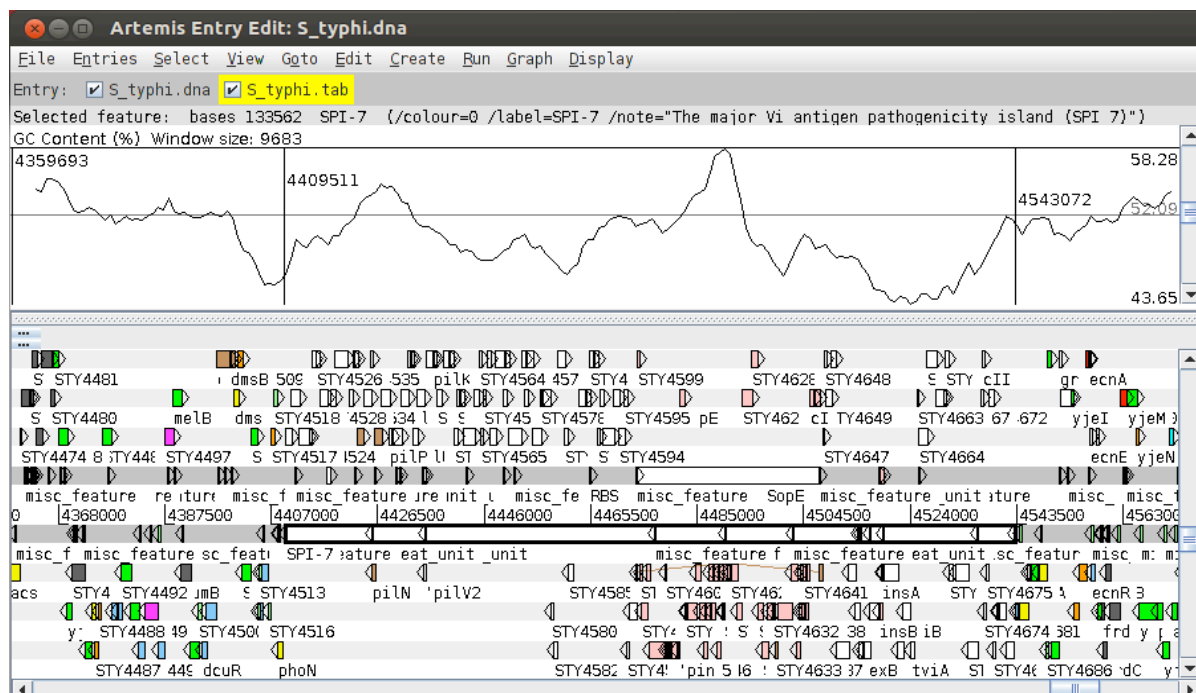
## Artemis Exercise 3

Now select the 'S.typhi.tab' entry box to switch on the annotation and go to position 4409511. The next region we are looking at is defined as a *Salmonella* pathogenicity island (SPI). SPI-7, or the major Vi pathogenicity island, is ~134 kb in length and contains ~30 kb of integrated bacteriophage.

The region you should be looking at is shown below and is a classical example of a *Salmonella* pathogenicity island (SPI). The definitions of what constitutes a pathogenicity island are quite diverse. However, below is a list of characteristics which are commonly seen within these regions, as described by Hacker *et al.*, 1997.

1. Often inserted alongside stable RNAs
2. Atypical G+C contents.
3. Carry virulence-related functions
4. Often carry genes encoding transposase or integrase-like proteins
5. Unstable and self-mobilisable
6. Of limited phylogenetic distribution

Have a look in and around this region and look for some of these features.



We are going to extract this region from the whole genome sequence and perform some more detailed analysis on it. We will aim to write and save new EMBL format files which will include just the annotations and DNA for this region. Follow the numbers on the next page to complete the task.

Artemis Entry Edit: S\_typhi.dna

File Entries Select View Goto Edit Create Run Graph Display

Entry:  S\_typhi.dna  S\_typhi.dna

136810 selected bases on forward strand

GC Content (%) Window size: 9583

4359693

Click 'Edit'

Click 'Edit subsequence (and features)'

1 Select region by clicking with the left mouse button & dragging

2

3

S STY4481 dmsB 509 S

S STY4480 melB dms STY451

STY4474 8 STY448 STY4497 S STY451

misc\_feature reiture misc\_f misc

0 4368000 4387500 4407000

misc\_f misc feature sc feat SPI-7 nature eat unit unit

STY4513 pilN 'pilV2

STY4516

phoN

STY4626 STY4648 E STY cII gr ecnA

STY462 cI TY4649 STY4663 67 672 yjeI yjeM

STY4647 STY4664 ecnE yjeN

ture SopE misc\_feature\_unit iture misc\_misc\_f

85000 4504500 4524000 4543500 4563000

STY4585 STY460 STY461 insA STY4675 a ecnR 3

STY4580 STY462 STY463 38 insB iB STY4674 381 frd yjeA

STY4582 STY463 37 exB tviA STY4686 dC yjeM

A new Artemis window will appear displaying only the region that you highlighted

Artemis Entry Edit

File Entries Select View Goto Edit Create Run Graph Display

Entry:  no name  no name

Nothing selected

Nothing selected

STY4525 STY4526

STY4522 STY4523

STY4521 STY4524

misc\_feature

900 1600 2400 3200 4000 4800 5600 6400 7200

trnA

phoN

L L R I F W L Y F P R L Y L H P Q Q L V S I V G N P L S H S P # N K P P Y H S G K

Y W R Y S G Y I F H A Y I C I L S S L F P S W G T L Y R I A L K I S L L I I A E S

T G E N L A I F S T L I S A S S A C F H R G E P F I A + P L K # A S L S + R K

CTACTGGCGAGAATATCTGGCTATATTTCCACGCTTATATCGCATCCTCAGCAGCTTGTTCCATCGTGGGGAACCCCTTATGCGCATAGCCCTTAAAATAAGCCCTTATCATAGCGGAAAC

GATGACCCCTTATAGACCGGATATAAAAAGTGGCAATATAGAGCGTAGGAGTGGTGGCAACAAGGTAGCACCCCTTGGGAAATAGCGTATCGGGAATTTATTTCGGAGGAATAGTATGCGCTTT

S A L I N Q S Y K G R K Y R C G \* C S T E M T P F G K D C L G # F L G G # \* L P F

+ Q R S Y E P + I K W A # I Q M R L L K N G D H P V R # R M A R L L I L L R R I M A S L

V P S F I R A I N E V S I D A D E A A Q K W R P S G K I A Y G K F Y A E K D Y R F

CDS	<1	271	c	Similar to Salmonella typhimurium nonspecific acid phosphatase precursor phoN SW:PHON_SALT
trnA	975	1038	c	possible truncated trnA Phe.
misc_feature	975	134536	c	The major Vi antigen pathogenicity island (SPI 7)
CDS	1116	2150	c	Weakly similar to the C-terminus of several polysaccharide biosynthesis proteins e.g. Str
CDS	2147	3511	c	Similar to Bacteriophage P1 Ban helicase TR:080281 (EMBL:AJ011592) (453 aa) fasta scores:
misc_feature	2777	2800	c	P500017 ATP/GTP-binding site motif A (P-loop)
CDS	3504	5303	c	no significant database hits
CDS	5472	5777	c	Doubtful CDS
CDS	5905	6486	c	no significant database hits.
CDS	6571	7128	c	Weakly similar to Yersinia pestis orf 77 TR:Q92381 (EMBL:AL031866) (193 aa) fasta scores:
CDS	7373	6716	c	no significant database hits
misc_feature	8718	9154	c	Low G+C region containing repeat region with 10xTGGT(A/-)(T/C)AAAAA(A/G)T.
CDS	9302	10981	c	no significant database hits. Contains a hydrophilic region in the N-terminus between resi
CDS	10192	12186	c	Previously sequenced Salmonella typhi topoisomerase B TopB TR:Q9PHF5 (EMBL:AF000901) (664
CDS	12854	13303	c	no significant database hits
CDS	13384	13602	c	doubtful CDS
CDS	13616	14151	c	Previously sequenced Salmonella typhi topoisomerase B TopB TR:Q9PHF5 (EMBL:AF000901) (664

Note the entry names have changed

Note the bases have been renumbered from the first base you selected.



Note that the two entries on the grey 'Entry' line are now denoted 'no name'. They represent the same information in the same order as the original Artemis window but simply have no assigned 'Entry' names. As the sub-sequence is now viewed in a new Artemis session, this prevents the original files (S\_typhi.dna and S\_typhi.tab) from being over-written.

We will save the new files with relevant names to avoid confusion. So click on the 'File' menu then 'Save An Entry As' and then 'New File'. Another menu will ask you to choose one of the entries listed. At this point they will both be called 'no name'. Left click on the top entry in the list. A window will appear asking you to give this file a name. Save this file as spi7.dna

Do the same again for the second unnamed entry and save it as spi7.tab

The screenshot shows the Artemis Entry Edit window. The 'File' menu is open, and 'Save An Entry As' is selected, leading to a sub-menu where 'New File' is chosen. The main window displays a DNA sequence with a feature table below it. The feature table includes entries for CDS, tRNA, and misc\_feature, with their respective coordinates and descriptions.

Feature Type	Start	End	Description
CDS	<1	271	c Similar to Salmonella typhimurium nonspecific acid phosphatase precursor phoN SW:PHON_SALT
tRNA	975	1038	c possible truncated tRNA Phe
misc_feature	975	134536	c The major Vi antigen pathogenicity island (SPI 7)
CDS	1116	2150	Weakly similar to the C-terminus of several polysaccharide biosynthesis proteins e.g. Str
CDS	2147	3511	Similar to Bacteriophage P1 Ban helicase TR:Q80281 (EMBL:AJ011592) (453 aa) fasta scores:
misc_feature	2777	2800	PS00017 ATP/GTP-binding site motif A (P-loop)
CDS	3504	5303	no significant database hits
CDS	5472	5777	Doubtful CDS
CDS	5905	6486	no significant database hits.
CDS	6571	7128	Weakly similar to Yersinia pestis orf 77 TR:Q9Z381 (EMBL:AL031866) (193 aa) fasta scores:
CDS	7373	8716	no significant database hits
misc_feature	8718	9154	Low G+C region containing repeat region with 10xTGGT(A/-)(T/C)AAAAA(A/G)T.
CDS	9302	10081	no significant database hits. Contains a hydrophobic region in the N-terminus between resi
CDS	10192	12186	Previously sequenced Salmonella typhi topoisomerase B TopB TR:Q9RHF5 (EMBL:AF000001) (664
CDS	12854	13303	no significant database hits
CDS	13384	13602	doubtful CDS

We are going to look at this region in more detail and to attempt to define the limits of the bacteriophage that lies within this region. Luckily for us all the phage-related genes within this region have been given a colour code number 12 (pink; for a list of the other numerical values that Artemis will display as colours for features see **Appendix IX**). We are going to use this information to select all the relevant phage genes using the Feature selector as shown below and then define the limits of the bacteriophage.

First we need to create a new entry (click 'Create' then 'New Entry'). Another entry will appear on the entry line called, you guessed it, 'no name'. We will eventually copy all our phage-related genes into here.

The image shows a sequence of six numbered steps for using the Artemis Feature Selector:

- Click 'Select' then 'Feature Selector'**: The 'Select' menu is open, and 'Feature Selector...' is highlighted.
- Make sure the buttons are selected**: A yellow callout box points to the 'Feature Selector' button.
- Set Key to 'CDS' and Qualifier to 'colour'**: The 'Artemis Feature Selector' dialog box is open. 'CDS' is selected in the 'Key' dropdown, and 'colour' is selected in the 'Qualifier' dropdown.
- Type search term**: The search term '12' is entered in the 'Containing this text' field.
- Click to select features containing search term**: The 'Select' button is clicked, resulting in a list of features.
- Click to view selected features in a list**: The 'View' button is clicked, opening a window titled 'All features with key "CDS" with qualifier "colour" containing text "12"'. This window displays a list of features with columns for ID, description, and similarity.

The feature list in window (6) includes the following entries:

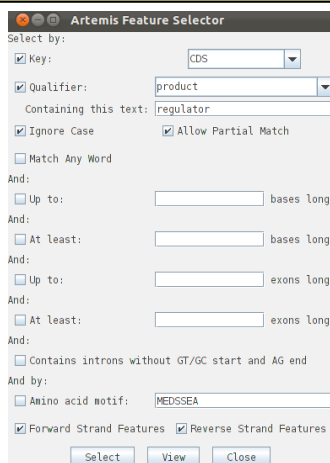
ID	Description	Similarity
CDS 65714	66991	no significant database hits
CDS 66178	66396	c Similar to Escherichia coli prophage P2 Ogr protein SW:0GRK_E
CDS 66464	67564	c Similar to Bacteriophage P2 late gene control protein D SW:VPI
CDS 67561	68046	c Similar to Bacteriophage P2 complete genome U essential tail p
CDS 68046	76826	c Similar to Bacteriophage 186 protein G TR:Q37848 (EMBL:U52222)
CDS 70819	70938	c Similar to Bacteriophage 186 Otr52 H TR:Q60315 (EMBL:U52222)
CDS 70953	71255	c Similar to Bacteriophage P2 complete genome E_essential tail
CDS 71310	71825	c Similar to Bacteriophage P2 major tail tube protein fII SW:VPI
CDS 71835	73007	c Similar to Bacteriophage P2 major tail sheath protein fI SW:VPI
CDS 73542	74284	c Similar to Salmonella typhimurium invasion-associated secreted
CDS 74462	74869	c Similar to Bacteriophage P2 probable tail fiber assembly prot
CDS 74876	76495	c Similar to Bacteriophage P2 probable tail fiber protein SW:VPI
CDS 76492	77097	c Similar to Bacteriophage P2 tail protein I SW:VPI_BFP2 (P2670)
CDS 77090	77998	c Similar to Bacteriophage P2 baseplate assembly protein J SW:VPI
CDS 77985	78344	c Similar to Bacteriophage P2 baseplate assembly protein W SW:VPI
CDS 78341	78919	c Similar to Bacteriophage P2 baseplate assembly protein V SW:VPI
CDS 78988	79434	c Similar to Bacteriophage P2 tail completion protein S SW:VPI
CDS 79427	79858	c Similar to Bacteriophage P2 tail completion protein R SW:VPI
CDS 79954	80379	c Similar to Bacteriophage P2 protein LysB protein involved in
CDS 80379	80756	c no significant database hits. Contains possible membrane spann
CDS 80761	81231	c Similar to Serratia marcescens putative phage lysozyme NucD TF

The genes listed in (6) are only those fitting your selection criteria. They can be copied or cut / moved in to a new entry so we can view them in isolation from the rest of the information within spi7.tab.

Firstly in window (6) select all of the CDSs shown by clicking on the 'Select' menu and then selecting 'All'. All the features listed in window (6) should now be highlighted. To copy them to another entry (file) click 'Edit' then 'Copy Selected Features To' then 'no name'. Close the two smaller feature selector windows and return to the SPI-7 Artemis window. You could rename the 'no name' entry as phage.tab, as you did before (if you can't remember how to do it have a look at page 14). Temporarily remove the features contained in 'spi7.tab' file by left clicking on the entry button on the grey entry line. Only the phage genes should remain.

## Additional methods for selecting/extracting features using the Feature Selector

It is worth noting that the Feature Selector can be used in many other ways to select and extract subsets of features from the genome, using eg text or amino acid searches.



Space for a search term or amino acid motif

## Defining the extent of the prophage

Even from this preliminary analysis it is clear that the prophage occupies a fairly discrete region within SPI-7 (see below). It is often useful to create a new DNA feature to define the limits of this type of genome landmark. To do this switch off stop codons, then use the left mouse button to click and drag over the region that you think defines the prophage.

Key	Start	End	Description
CDS	65437	65614	no significant database hits
CDS	65901	66119	Similar to Escherichia coli prophage P2 Dgp protein SW:09K_ECOLI (1) (72 aa) fasta scores: E(1)
CDS	66187	67287	Similar to Bacteriophage P2 late gene control protein D SW:VPO_BPP2 (P10312) (387 aa) fast
CDS	67284	67769	Similar to Bacteriophage P2 complete genome J essential tail protein TR:064315 (EMBL:AF0603
CDS	67769	70549	Similar to Bacteriophage 186 protein G TR:037649 (EMBL:U32222) (8812 aa) fasta scores: E(1)
CDS	70542	70661	Similar to Bacteriophage 186 0r152_H TR:080316 (EMBL:U32222) (58 aa) fasta scores: E(1); 3;
CDS	70676	70978	Similar to Bacteriophage P2 complete genome E, essential tail protein TR:064312 (EMBL:AF0603
CDS	71233	71346	Similar to Bacteriophage P2 esjor tail tube protein FI SW:VFP2_BPP2 (P22502) (171 aa) fast
CDS	71556	72730	Similar to Bacteriophage P2 major tail sheath protein FI SW:VFP1_BPP2 (P22501) (395 aa) fa
CDS	73205	73697	Similar to Salmonella typhimurium invasion-associated secreted protein SW:IE TR:026323 (EMB
CDS	74155	74592	Similar to Bacteriophage P2 probable tail fiber assembly protein G SW:VFA_BPP2 (P26669) (1)
CDS	74599	76216	Similar to Bacteriophage P2 probable tail fiber protein SW:VPH_BPP2 (1) (669 aa) fasta scor
CDS	76215	76620	Similar to Bacteriophage P2 tail protein I SW:VPI_BPP2 (P26701) (176 aa) fasta scores: E(1)
CDS	76813	77721	Similar to Bacteriophage P2 baseplate assembly protein J SW:VJP_BPP2 (P51767) (302 aa) fas
CDS	77738	78687	Similar to Bacteriophage P2 baseplate assembly protein V SW:VVP_BPP2 (P51768) (1115 aa) fas
CDS	78984	78942	Similar to Bacteriophage P2 baseplate assembly protein V SW:VVP_BPP2 (P51768) (1115 aa) fas

While the region is highlighted, click on the 'Create' menu and select 'Create feature from base range'. A feature edit window will appear. The default 'Key' value given by Artemis when creating a new feature is 'CDS'. With this 'Key' the newly created feature would automatically be put on the translation line. However, if we change this to 'misc\_feature' (an option in the 'Key' drop down menu in the top left hand corner of the Edit window), Artemis will place this feature on the DNA line. This is perhaps more appropriate and is easier to visualise. You can also add a qualifier, such as '/label': select 'label' from the 'Add Qualifier' list and click 'Add Qualifier', '/label=' will appear in the text window; add text of your choice, then click 'OK'. That text will be used as a feature label to be displayed in the main sequence view panel.

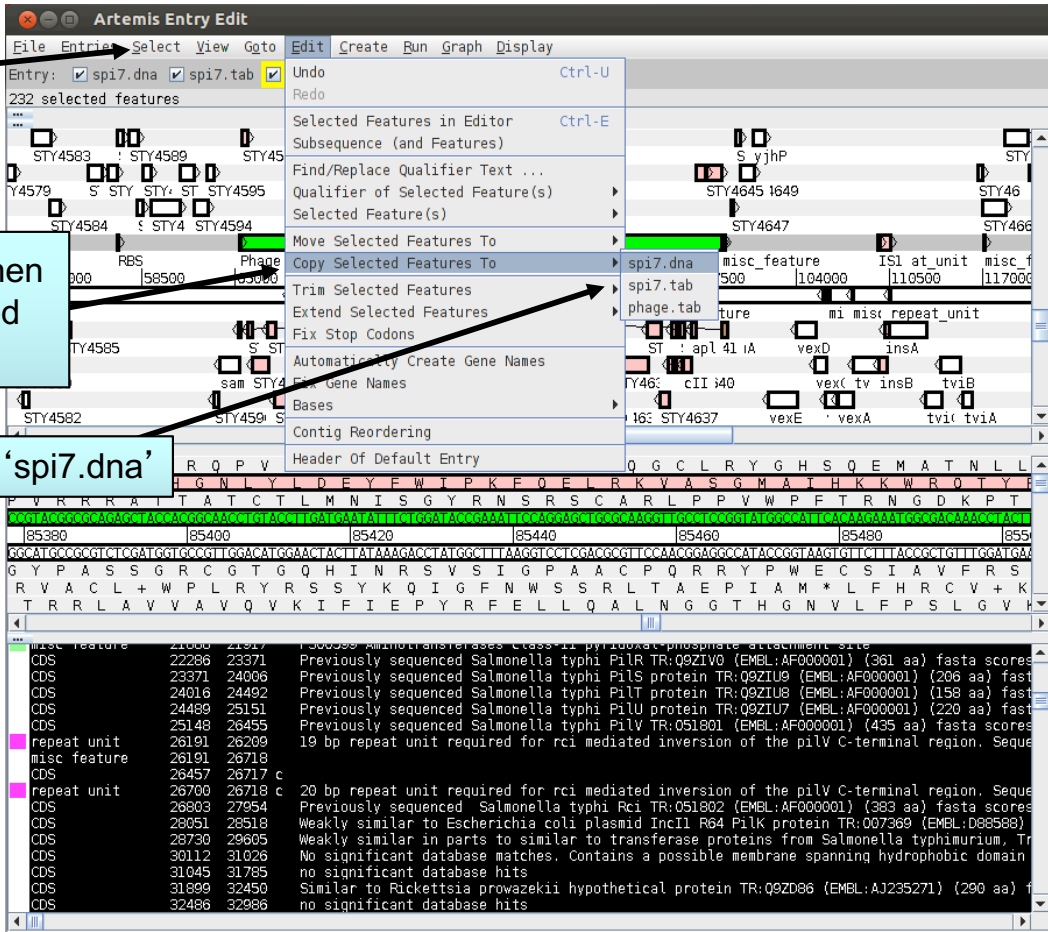
To see how well you have done, tick the little box to turn on spi7.tab.

Your final task is to write out the spi7 files in EMBL submission format, and create a merged annotation and sequence file in EMBL submission format. In Artemis you are going to copy the annotation features from the '.tab' file into the '.dna' file, and then save this entry in EMBL format. Don't worry about error messages popping up. This is because not all entries are accepted by the EMBL database.

1 Click 'Select' then 'All'

2 Click 'Edit', then 'Copy Selected Features To'

3 Select 'spi7.dna'

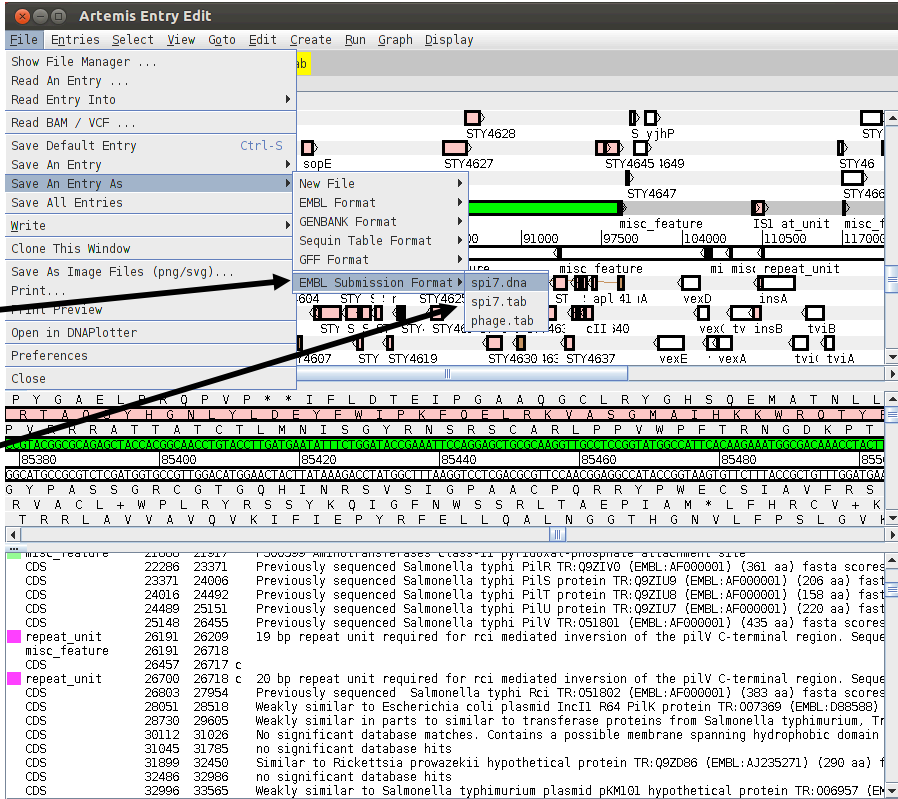


4 Click 'File' then 'Save An Entry As'

5 'EMBL Submission Format'

6 Select 'spi7.dna'

7 Save file as spi7.embl



Now open the EMBL format file that you have just created in Artemis.

Artemis Entry Edit: spi7.embl

File Entries Select View Goto Edit Create Run Graph Display

Entry:  spi7.embl

Nothing selected

\*\*\*

STY4521 STY4523 STY4524 STY4525 STY4526 STY4522 STY4528

misc\_feature

tRNA feature

\*\*\*

E # V V P G L G I E P R T R G F S I P L S K S + T P L \* L F E L V A G R R T H S N K  
 N K W C P D S E S N H G H G D F Q S P C Q K V R H R F D F L N W + Q A G E H I R I  
 . I S G A R T R N R T T D T G I F N P L V K K L D T A L T F \* I G S R Q A N T F E #  
 GAATAAGTGGTGCCCGGACTCGGAATCGAACCACGGACACGGGGATTTTCAATCCCCTGTCAAAAAGTTAGACACCGCTTTGACTTTTGAATTGGTAGCAGGCAGGCGAACACATTGGAATAAA  
 20 40 60 80 100 120  
 CTTATTCAACCACGGGCTGAGCCTTAGCITGGTGCCTGTGCCCTAAAAGTAGGGGAACAGTITTCATCTGTGGCGAACTGAAAAACTTAACCATGGTCCGTCGGCTGGTAAAGCCTAT  
 . Y T T G P S P I S G R V R P N E I G K D F L # V G S Q S K S N T A P L R V C E F L  
 F L H H G S E S D F W P C P S K \* D G Q \* F T L C R K S K K F Q Y C A P S C M R I F  
 I L P A R V R F R V V S V P I K L G R T L F N S V A K V K Q I P L L C A F V N S Y I

\*\*\*

tRNA	1	64	c	possible truncated tRNA Phe.
misc_feature	1	133562	c	The major Vi antigen pathogenicity island (SPI 7)
CDS	142	1176		Weakly similar to the C-terminus of several polysaccharide biosynthesis proteins e.g. Str
CDS	1173	2537		Similar to Bacteriophage P1 Ban helicase TR:080281 (EMBL:AJ011592) (453 aa) fasta scores:
misc_feature	1803	1826		PS00017 ATP/GTP-binding site motif A (P-loop)
CDS	2530	4329		no significant database hits
CDS	4498	4803		Doubtful CDS
CDS	4931	5512		no significant database hits.
CDS	5597	6154		Weakly similar to Yersinia pestis orf 77 TR:Q9Z381 (EMBL:AL031866) (193 aa) fasta scores:
CDS	6399	7742		no significant database hits
misc_feature	7744	8180		Low G+C region containing repeat region with 10xTGGT(A/-)(T/C)AAAAA(A/G)T.
CDS	8328	9107		no significant database hits. Contains a hydrophobic region in the N-terminus between resi
CDS	9218	11212		Previously sequenced Salmonella typhi topoisomerase B TopB TR:Q9RHF5 (EMBL:AF000001) (664
CDS	11880	12329		no significant database hits
CDS	12410	12628		doubtful CDS
CDS	12641	13177		Previously sequenced Salmonella typhi single strand binding protein ssB TR:Q9RHE4 (EMBL:AF

You will see that the colours of the features have now changed. This is because not all the qualifiers in the previous entry are accepted by the EMBL database, so some have not been saved in this format. This includes the '/colour' qualifier, so Artemis displays the features with default colours.

When you download sequence files from EMBL and visualize them in Artemis you will notice that they are displayed using default colours. You can customize your own annotation files with the '/colour' qualifier and chosen number (**Appendix IX**), to differentiate features. To do this you can use the Feature Selector to select certain features and annotate them all using the 'Edit', 'Change Qualifiers of Selected' function.

## Artemis Exercise 4

This exercise will introduce you to database searches and will give you a first insight in the annotation of genes.

Return to the *S. Typhi* window. The gene you will work on is *hpcC* (STY1136). Go to this gene by using one of the different methods you have learned so far. You will now analyse the reading frame of *hpcC* gene. To do this switch on the stop codons (also, switch off the GC content graph if still displayed).

As you can see the gene is full with stop codons indicating that we are looking at a pseudogene. To correct the annotation we are going to use database search. Follow the numbers in the figure below to start a database search. The search may take a couple of minutes to run; a banner will pop up to tell you when its complete (3).

### 1 Select CDS

The screenshot shows the Artemis software interface. On the left, the 'Entry Edit: S\_typhi.dna' window displays the genomic map of the *hpcC* gene. A red box highlights the 'hpcC' feature, and an arrow points to it from the '1 Select CDS' label. Below the map, the 'misc\_feature' table is visible, listing various features including CDS and misc\_feature entries with their coordinates and descriptions.

On the right, the 'Run' window shows a search query: `/colour=11 /gene="hpcC"/gene="STY1136" /product="5-carbo`. A red box highlights the 'Run blastp on selected features against' option, and an arrow points to it from the '2 Start blastp' label. Below the query, the 'blastp process completed' dialog box is shown, indicating the search is finished.

To view the search results click 'View', then 'Search Results', then 'blastp results'. The results will appear in a scrollable window. Scroll down to the first sequence comparison and you should see the results as shown in the next figure.





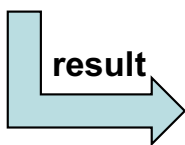
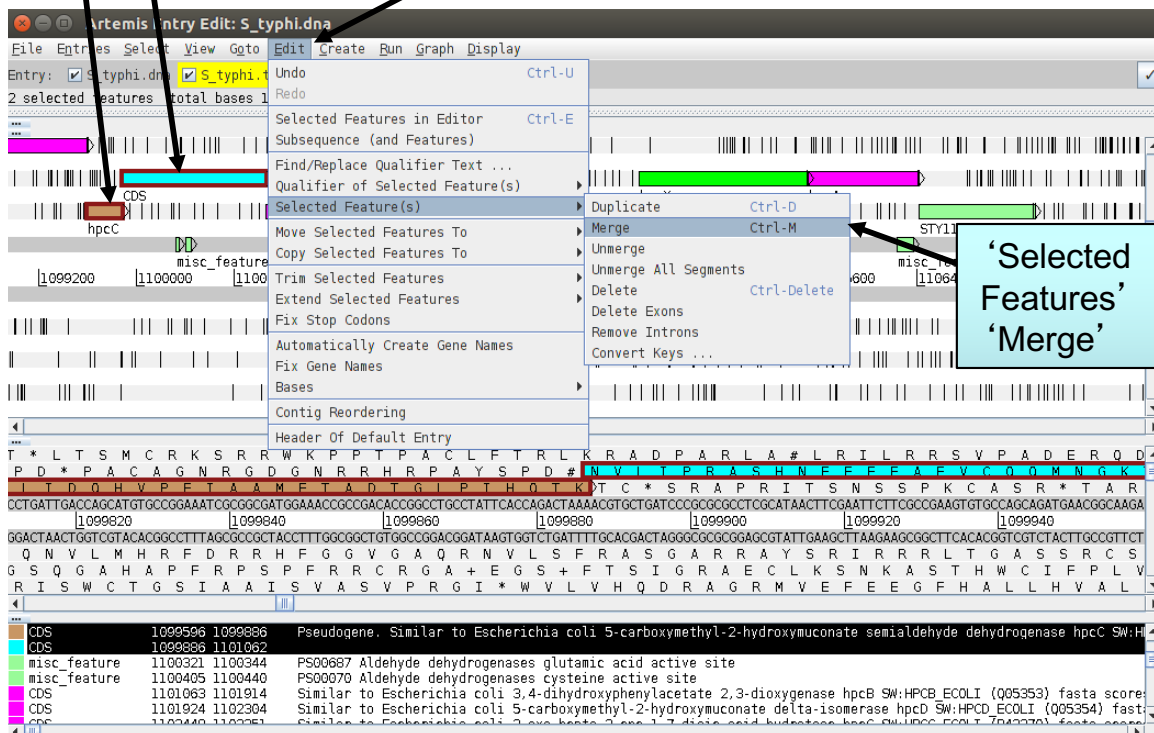
The new CDS feature can then be merged with the original gene as shown below (1-3).

A small window will appear asking you whether you are sure you want to merge these features. Another window will then ask you if you want to 'delete old features'. If you click 'yes' the CDS features you have just merged will disappear leaving the single merged CDS. If you select 'no' all of the three CDS features (the two CDSs you started with plus the merged feature) will be retained.

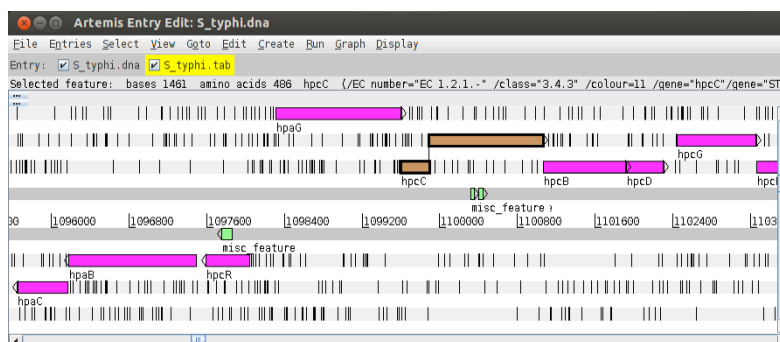
1

Select both the original gene-model and the new CDS feature, which is to be merged with it to form a new gene (to select more than one feature you must hold the shift key down).

2 Click 'Edit'



result





## Artemis Exercise 4 - Second part

In the first part of the exercise you have learned how to correct a gene annotation. But what if you think a gene is missing?

Remember that there are loads of genomes that were submitted to the databases several years ago and in general the annotation is not updated to take into account new data. Sometimes it is worth checking regions which look strange to you.

Go to position 2,248,400 by using one of the different methods you have learned so far. If you look carefully you will notice a region shown below which there is no predicted gene. This type of non-coding region in *Salmonella* is very unusual (this is also true for other bacteria). To determine if this non-coding region is truly as published, load the codon usage information for *Salmonella* into Artemis by following the figure below.

The file 'S\_typhi.cod' contains codon usage information taken from a public website (see below).

1

Click Graph

2

Click 'Add usage plots' and select 'S\_typhi.cod'

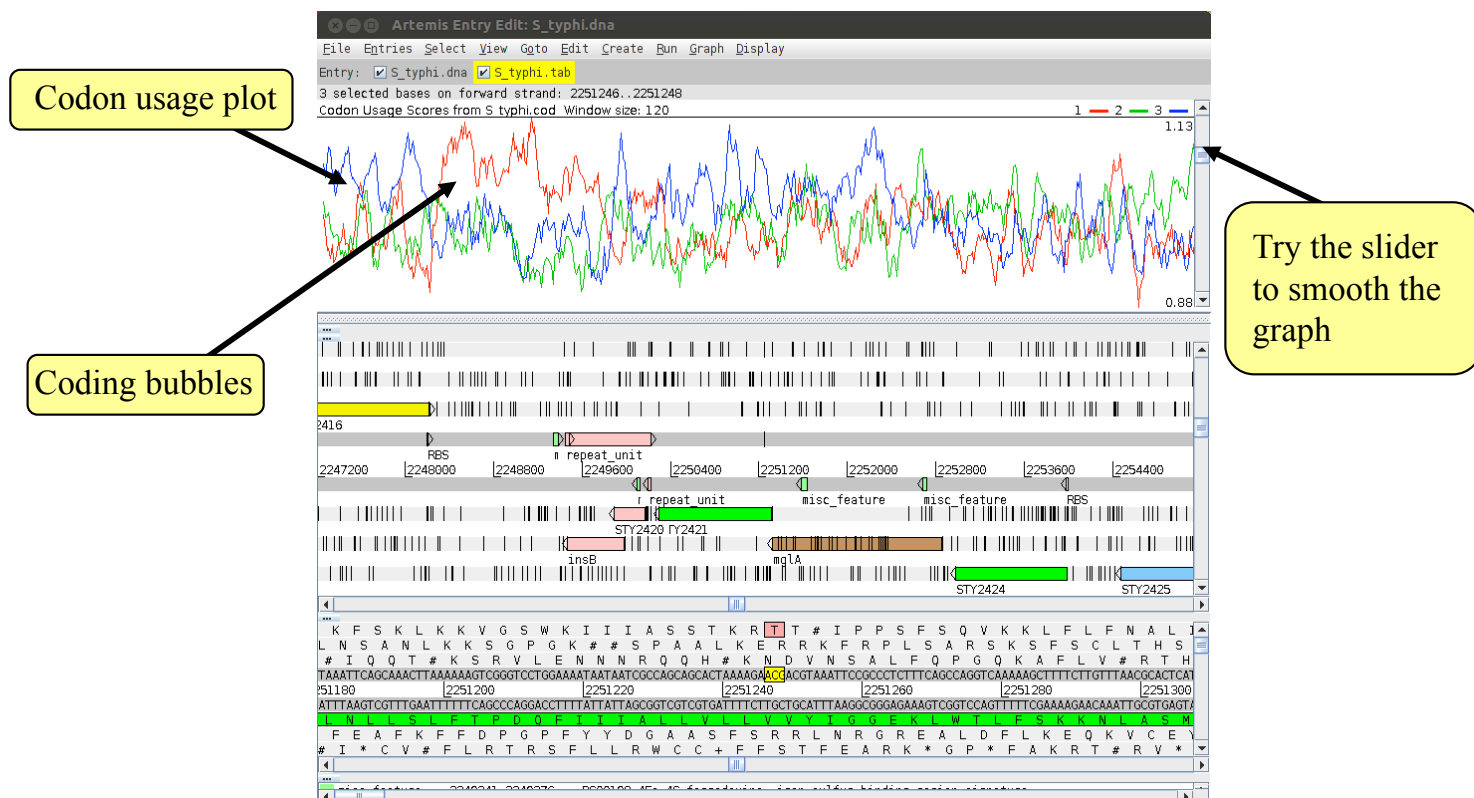
Non-coding

The screenshot shows the Artemis genome browser interface. At the top, the 'Graphs' menu is open, with 'Add Usage Plots...' selected. Below the menu, a list of graph types is visible, including 'GC Content (%)', 'GC Content (%) With A 2.5 SD Cutoff', 'AG Content (%)', 'GC Frame Plot', 'Reverse GC Frame Plot', 'Correlation Scores', 'Reverse Correlation Scores', 'GC Deviation (G-C)/(G+C)', 'AT Deviation (A-T)/(A+T)', and 'Karlin Signature Difference'. On the right side of the interface, a 'Codon usage table' is displayed, showing various codon usage metrics for different organisms. The table includes columns for organism names and their corresponding codon usage values. The table is titled 'Codon usage table taken from: www.kazusa.or.jp/codon'.

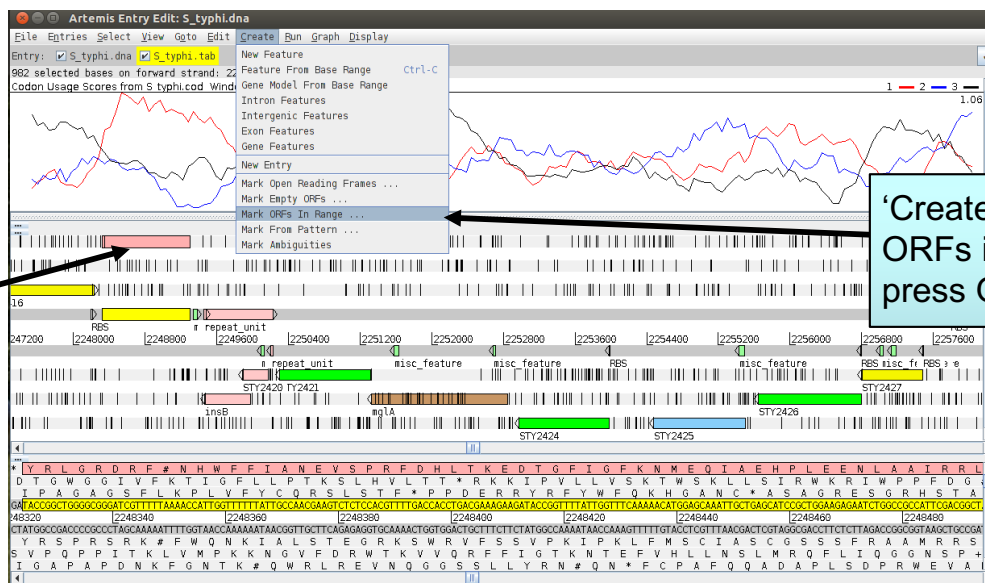
UUU	24.4	(2445)	UCU	12.1	(1212)	UAU	18.6	(1864)	UGU	6.5	(651)
UUC	15.6	(1563)	UCC	10.5	(1054)	UAC	13.3	(1335)	UGC	5.8	(585)
UUA	15.0	(1502)	UCA	15.3	(1534)	UAA	1.6	(158)	UGA	1.4	(145)
UUG	12.2	(1222)	UCG	9.5	(952)	UAG	0.5	(47)	UGG	13.3	(1326)
CUU	16.1	(1614)	CCU	10.5	(1055)	CAU	11.4	(1143)	CGU	14.3	(1426)
CUC	11.2	(1120)	CCC	6.8	(681)	CAC	7.2	(725)	CGC	11.7	(1170)
CUA	6.6	(656)	CCA	9.3	(930)	CAA	13.8	(1381)	CGA	6.7	(666)
CUG	34.1	(3411)	CCG	14.1	(1415)	CAG	27.2	(2716)	CGG	8.2	(822)
AUU	27.3	(2729)	ACU	14.8	(1476)	AUU	27.0	(2699)	AUG	13.2	(1319)
AUC	20.4	(2037)	ACC	20.1	(2012)	AAG	22.7	(2266)	AGC	16.4	(1639)
AUA	9.9	(988)	ACA	14.4	(1445)	AAA	35.4	(3546)	AGA	6.0	(599)
AUG	26.0	(2597)	ACG	15.5	(1552)	AAG	17.5	(1752)	AGG	4.7	(472)
GUU	20.3	(2033)	UCU	17.7	(1770)	GAU	33.9	(3394)	GGU	19.5	(1949)
GUC	15.4	(1541)	GCC	21.6	(2160)	GAC	20.0	(1990)	GGC	21.0	(2090)
GUA	12.0	(1201)	GCA	21.0	(2105)	GAA	34.8	(3481)	GGA	12.6	(1259)
GUU	19.4	(1941)	GCC	19.3	(1931)	GAG	20.8	(2080)	GGG	13.6	(1359)

Codon usage table taken from: [www.kazusa.or.jp/codon](http://www.kazusa.or.jp/codon)

When you first load the codon table into Artemis the graphs calculated for both upper and lower strands will be displayed (not shown). To remove one of these from the view click on the Graph menu and uncheck the box alongside the option 'Reverse Codon Usage Scores from S\_typhi.cod'.

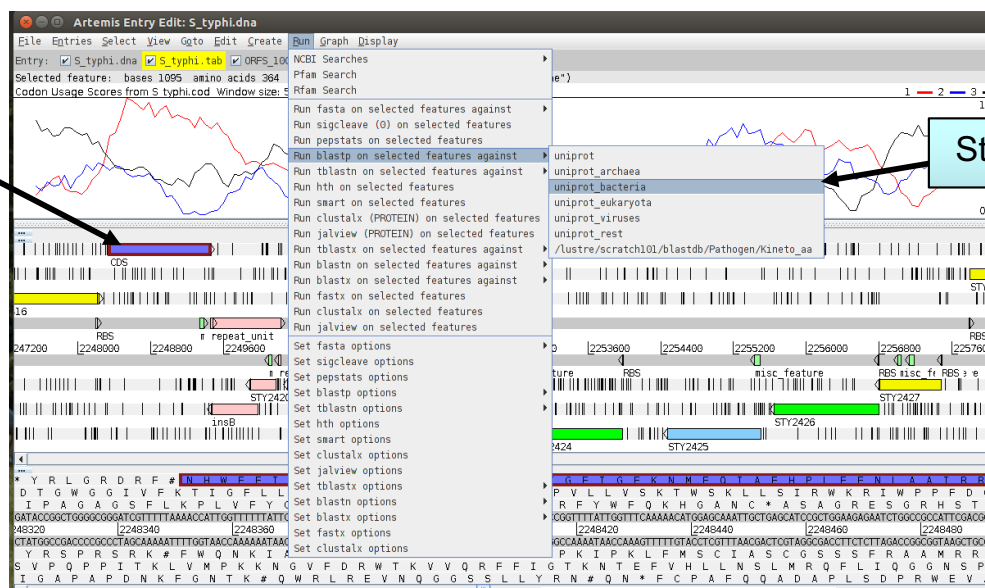


Based on the codon usage table Artemis calculates for each triplet in succession a score based on how well it matches the commonly used codons in that organism. The three lines shown above represent the scores for each reading frame. If the codons for a particular frame match those of the calculated codon usage table a high score is given. Practically speaking this manifests itself as a 'coding bubble' where a gap opens up in the plot indicating that this region is likely to be coding (see above). The plot suggests that this empty region actually encodes a product. So now we have to create the open reading frame (ORF), blast the amino acid sequence and add the annotation. Follow the instruction on the next page to do this.



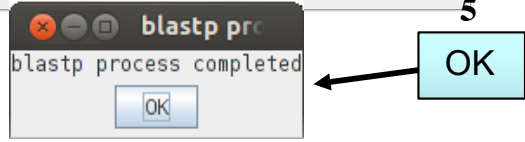
1  
Click and drag to highlight

2  
'Create', 'Mark ORFs in range', press OK



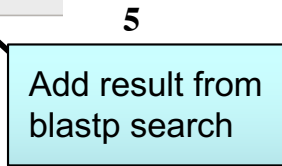
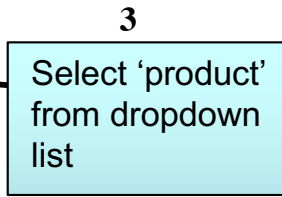
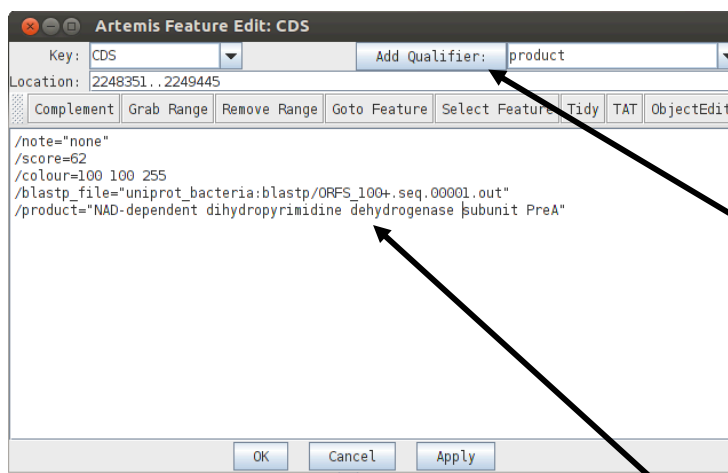
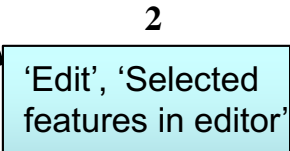
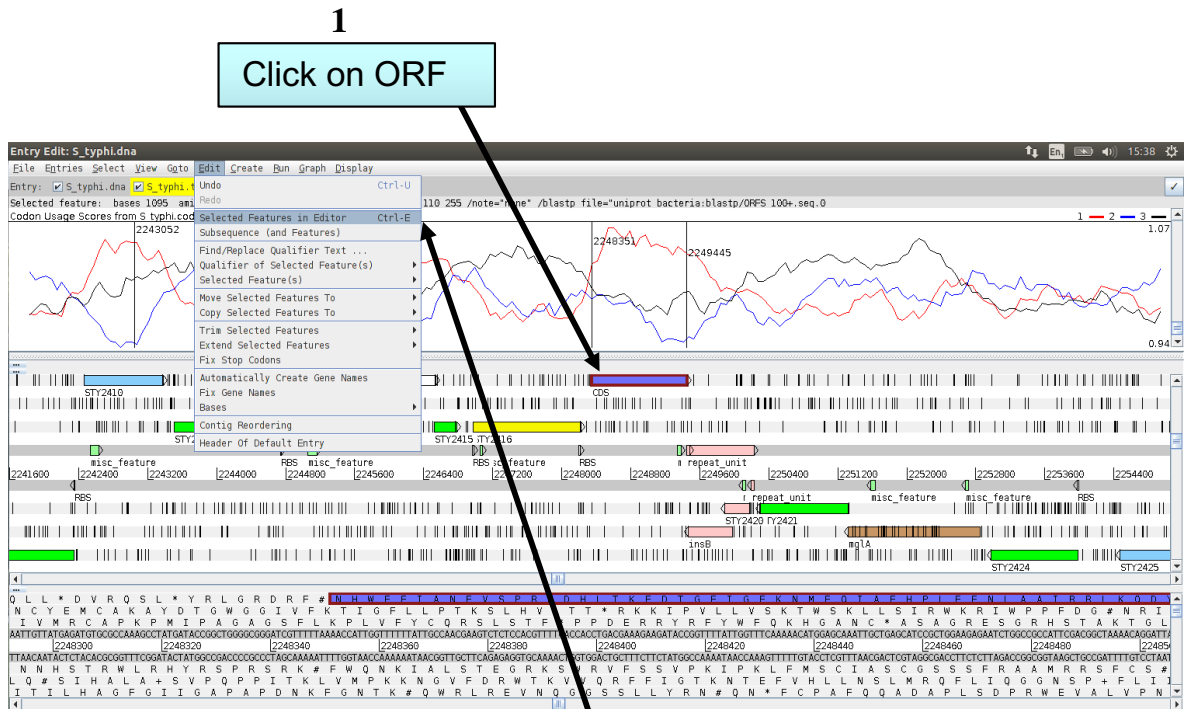
3  
Click on newly created ORF

4  
Start blastp



5  
OK

To view the search results click 'View', then 'Search Results', then 'blastp results'. The results will appear in a scrollable window. You see that the product of the gene is "NAD-dependent dihydropyrimidine dehydrogenase subunit PreA). To add the product to the annotation follow the instruction in the next page.



The annotation of the ORF is now complete. You can add as much information as you want. Have a look at the other qualifiers if some time is left. The last thing you have to do is copy the annotated feature to S\_typhi.tab. To do that select the feature and go to 'Edit', 'Copy selected features to' and click 'S\_typhi.tab'. Don't forget to save the tab file.

# Module 2

## Comparative Genomics

### Introduction

The Artemis Comparison Tool (ACT), also written by Kim Rutherford, was designed to extract the additional information that can only be gained by comparing the growing number of sequences from closely related organisms (Carver *et al.* 2005). ACT is based on Artemis, and so you will already be familiar with many of its core functions, and is essentially composed of three layers or windows. The top and bottom layers are mini Artemis windows (with their inherited functionality), showing the linear representations of the DNA sequences with their associated features. The middle window shows red and blue blocks, which span this middle layer and link conserved regions within the two sequences, in the forward and reverse orientation respectively. Consequently, if you were comparing two identical sequences in the same orientation you would see a solid red block extending over the length of the two sequences in this middle layer. If one of the sequences was reversed, and therefore present in the opposite orientation, there would be a blue 'hour glass' shape linking the two sequences. Unique regions in either of the sequences, such as insertions or deletions, would show up as breaks (white spaces) between the solid red or blue blocks.

In order to use ACT to investigate your own sequences of interest you will have to generate your own pairwise comparison files. Data used to draw the red or blue blocks that link conserved regions is generated by running pairwise BLASTN or TBLASTX comparisons of the sequences. ACT is written so that it will read the output of several different comparison file formats; these are outlined in Appendix III. Two of the formats can be generated using BLAST software freely downloadable from the NCBI, which can be loaded and run on a PC or Mac. Appendix V shows you how to generate comparison files from BLAST. Whilst having a local copy BLAST to generate ACT comparison files can be very useful, it means that you are tied to a particular computer. Another way of generating comparison files for ACT is to use the WebACT web resource (see page 16 of this module). This site allows you to cut and paste or upload your own sequences, and generate ACT readable BLASTN or TBLASTX comparison files.

### Aims

The aim of this Module is for you to become familiar with the basic functions of ACT by using a series of worked examples. Some of these examples will touch on exercises that were used in previous Modules, this is intentional. Hopefully, as well as introducing you to the basics of ACT, this Module will also show you how ACT can be used for not only looking at genome evolution but also to back up, or question, gene models and so on. In this module you will also use a web resource, WebACT, to generate your own comparison files and view them in ACT.

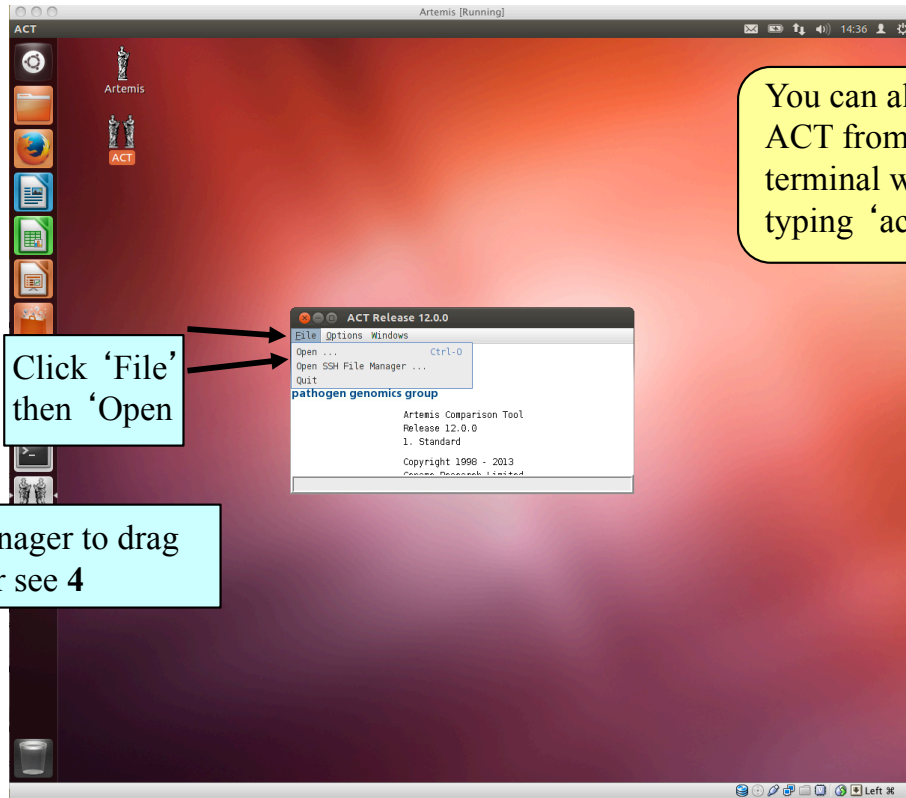
# 1. Starting up the ACT software

Double click the ACT icon on the desktop. A small start up window will appear.

The files you will need for this exercise are: *S\_typhi.dna.gz*

*S\_typhi.dna\_vs\_EcK12.dna.crunch.gz*

*EcK12.dna.gz*

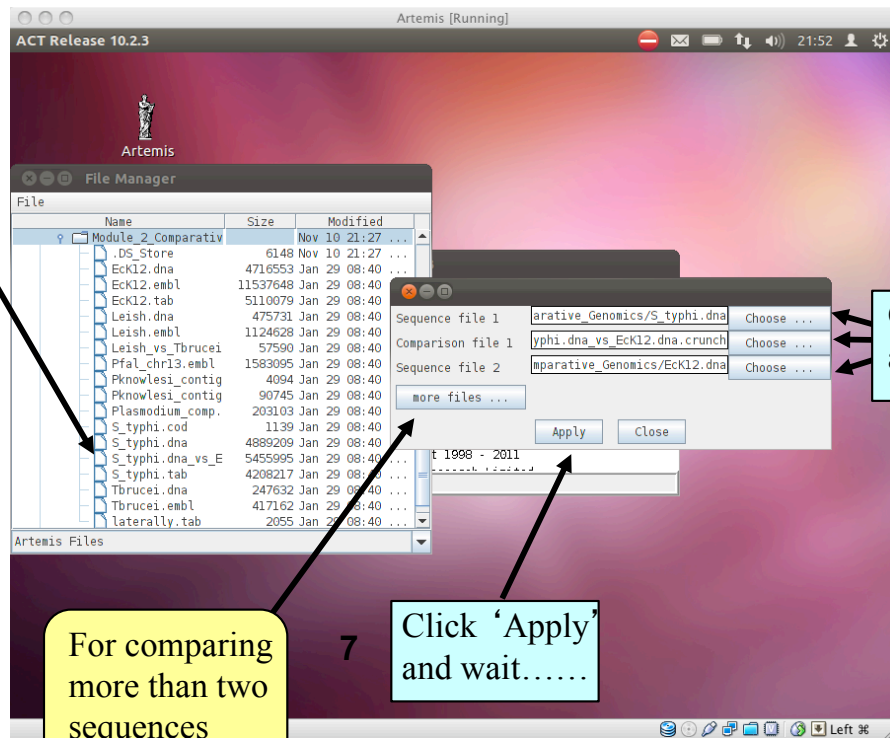


You can also start ACT from the terminal window by typing 'act'

1 Click 'File' then 'Open'

3 Use the File manager to drag and drop files or see 4

Comparison files end with '.crunch.gz'



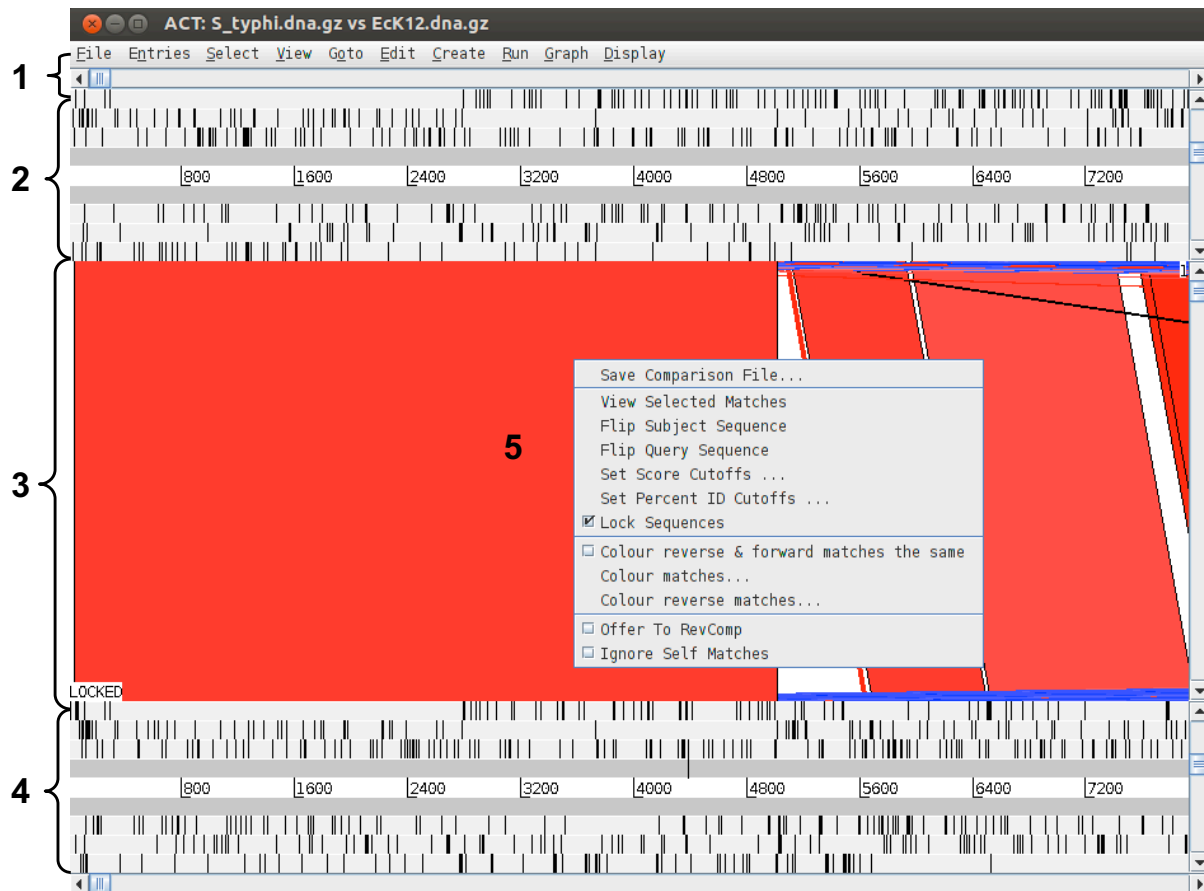
4, 5 & 6 Click and select appropriate files

7 For comparing more than two sequences

7 Click 'Apply' and wait.....

## 2. The basics of ACT

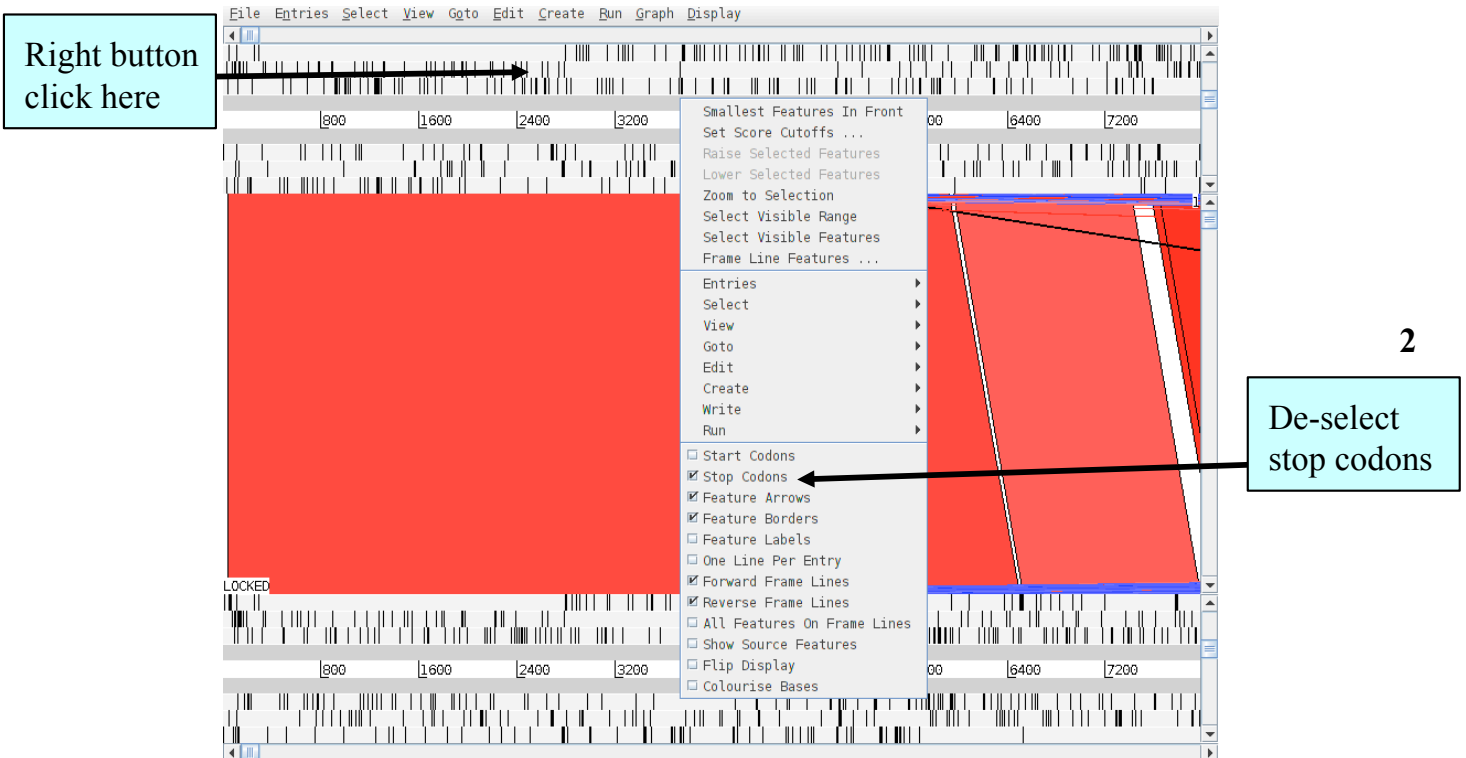
You should now have a window like this so let's see what is there.



1. Drop-down menus. These are mostly the same as in Artemis. The major difference you'll find is that after clicking on a menu header you will then need to select a DNA sequence before going to the full drop-down menu.
2. This is the Sequence view panel for 'Sequence file 1' (Subject Sequence) you selected earlier. It's a slightly compressed version of the Artemis main view panel. The panel retains the sliders for scrolling along the genome and for zooming in and out.
3. The Comparison View. This panel displays the regions of similarity between two sequences. Red blocks link similar regions of DNA with the intensity of red colour directly proportional to the level of similarity. Double clicking on a red block will centralise it. Blue blocks link regions that are inverted with respect to each other.
4. Artemis-style Sequence View panel for 'Sequence file 2' (Query Sequence).
5. Right button click in the Comparison View panel brings up this important ACT-specific menu which we will use later.



1



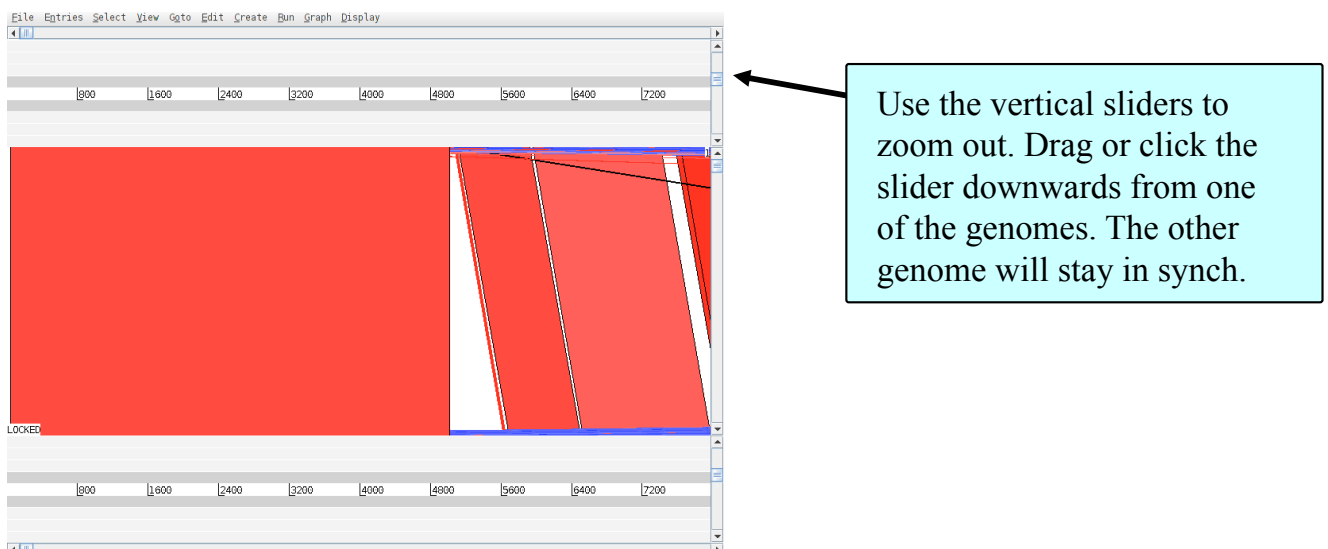
### 3. Exercise 1

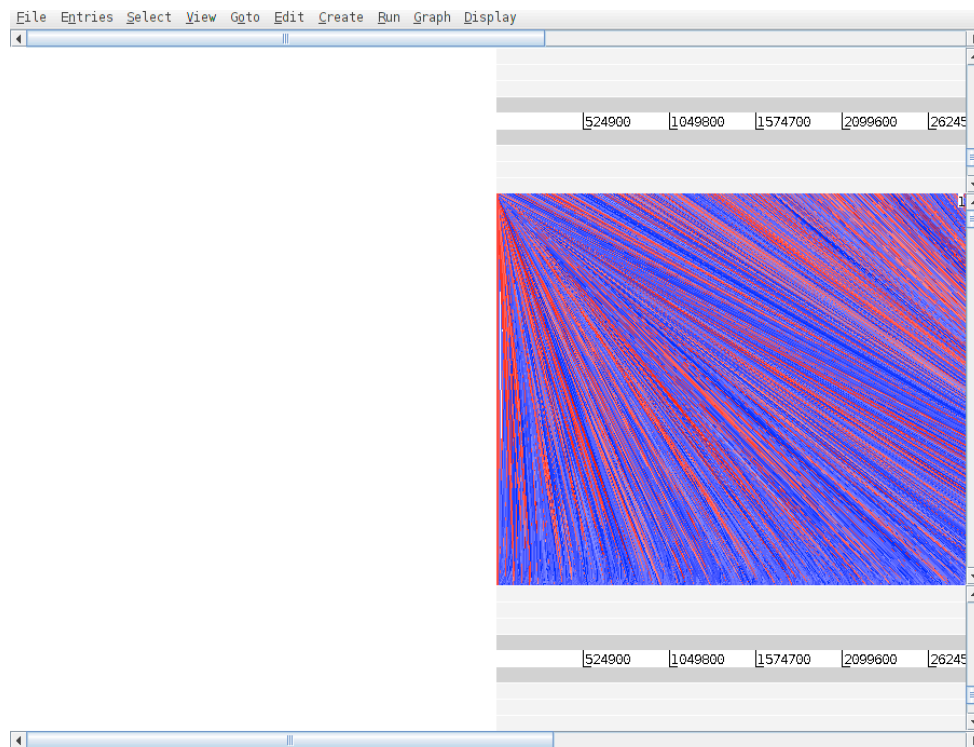
#### Introduction & Aims

In this first exercise we are going to explore the basic features of ACT. Using the ACT session you have just opened we firstly are going to zoom outwards until we can see the entire *S. Typhi* chromosome compared against the *E. coli* K12 chromosome. As for the Artemis exercises we should turn off the stop codons to clear the view and speed up the process of zooming out.

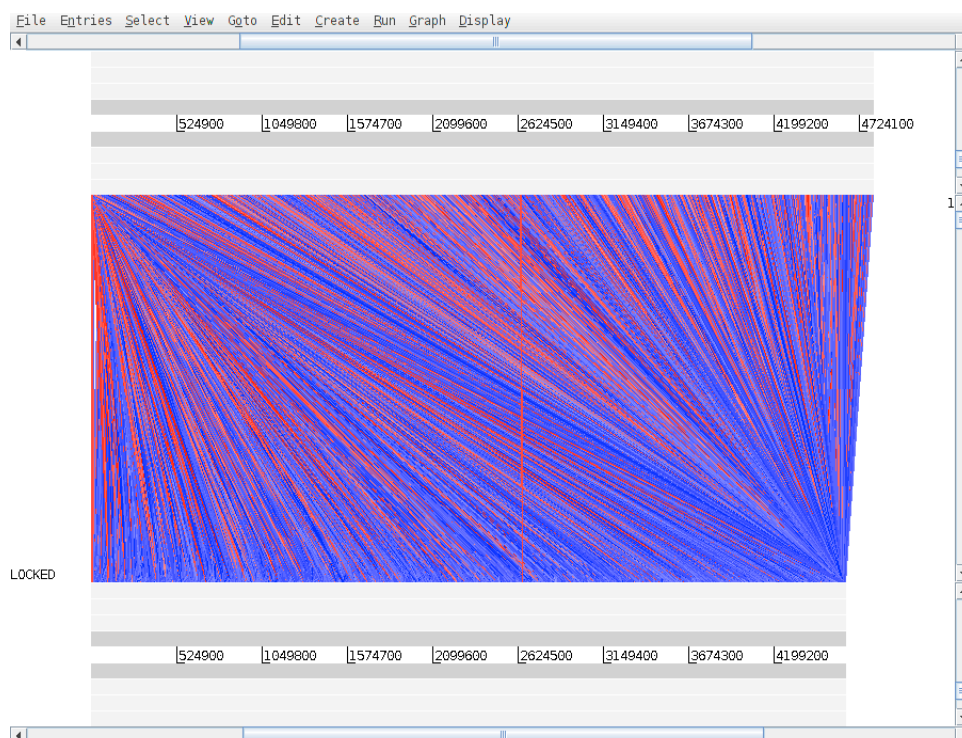
The only difference between ACT and Artemis when applying changes to the sequence views is that in ACT you must click the right mouse button over the specific sequence that you wish to change, as shown above.

Now turn the stop codons off in the other sequence too. Your ACT window should look something like the one below:

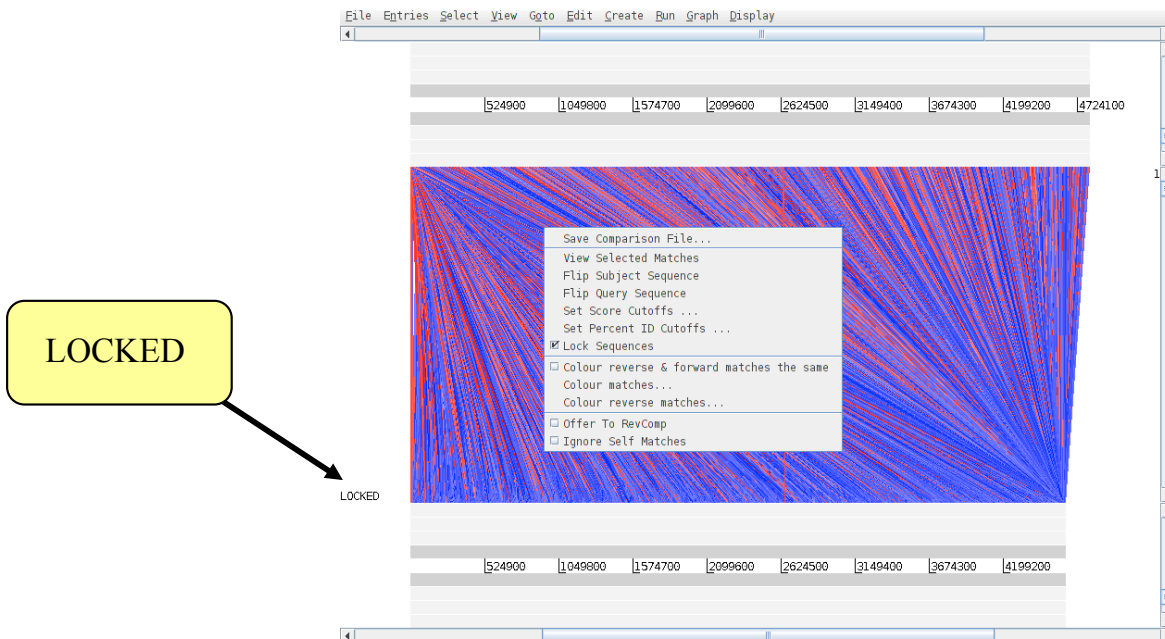




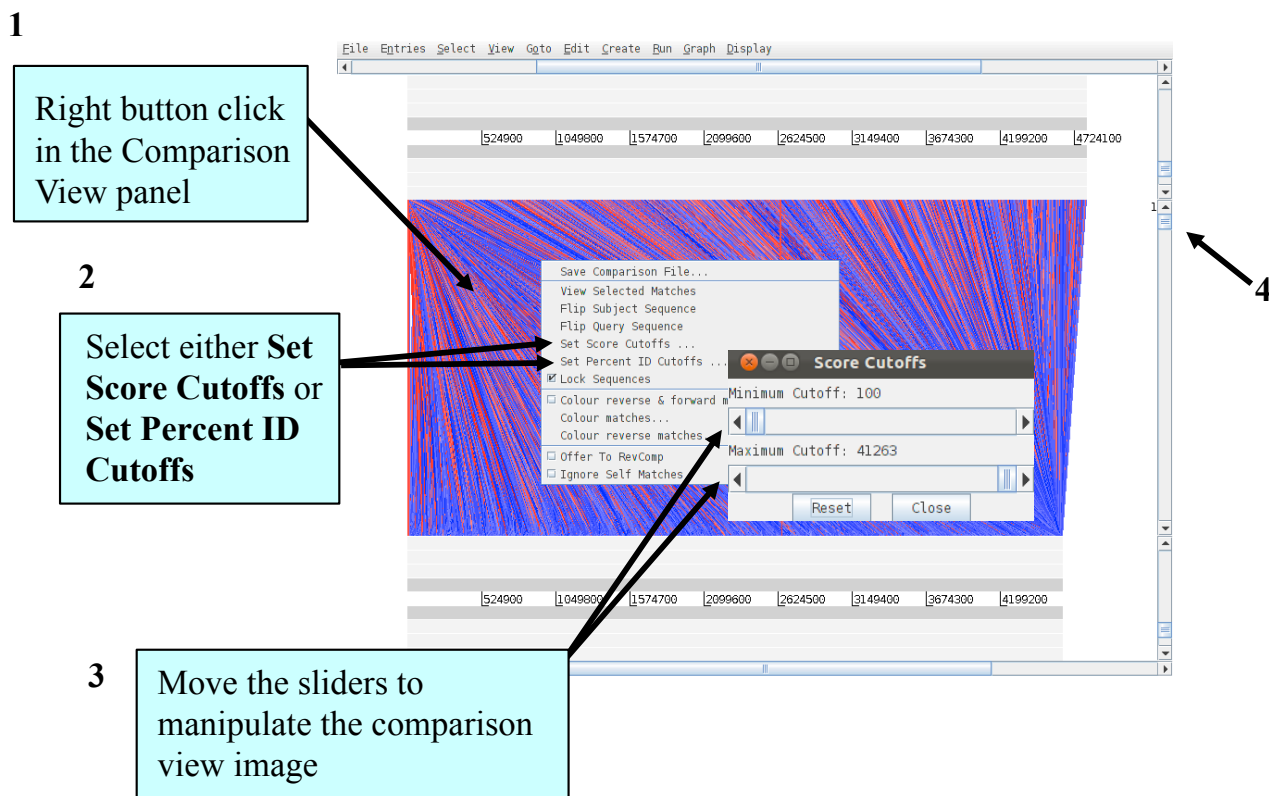
Once zoomed out, your ACT window should look similar to the one shown above. If the genomes in view fall out of view to the right of the screen, use the horizontal sliders to scroll the image and bring the whole sequence into view, as shown below. You may have to play around with the level of zoom to get the whole genomes shown in the same screen as shown below.

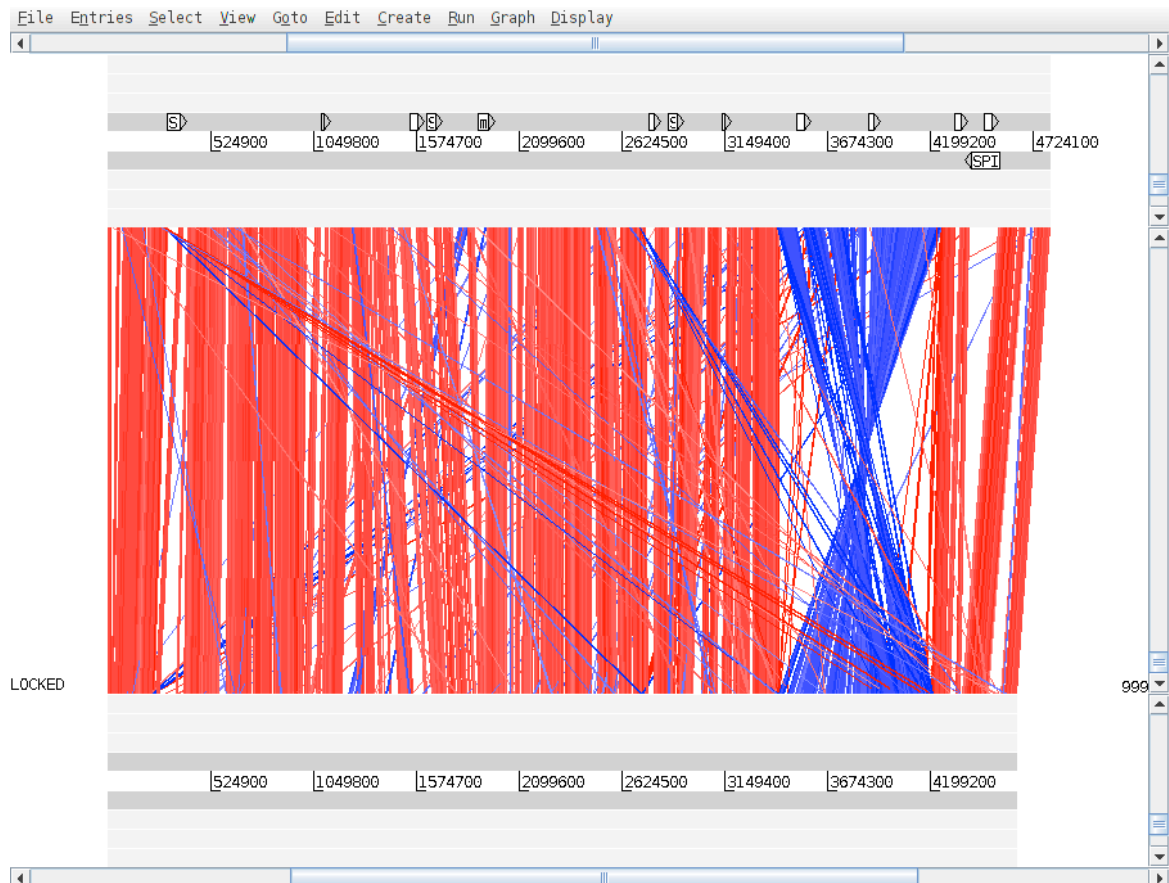


Notice that when you scroll along with either slider both genomes move together. This is because they are 'locked' together. Right click over the middle comparison view panel. A small menu will appear, select **Unlock sequences** and then scroll one of the horizontal sliders. Notice that 'LOCKED' has disappeared from the comparison view panel and the genomes will now move independently



You can optimise your image by either removing 'low scoring' (or percentage ID) hits from view, as shown below **1-3** or by using the slider on the the comparison view panel (**4**). The slider allows you to filter the regions of similarity based on the length of sequence over which the similarity occurs, sometimes described as the "footprint".





#### 4. Things to try out in ACT

Load into the top sequence (*S. typhi*) a '.tab' file called 'laterally.tab.gz'. You will need to use the 'File' menu and select the correct genome sequence ('S.typhi.dna.gz') before you can read in an entry. If you are zoomed out and looking at the whole of both genomes you should see the above. The small white boxes are the regions of atypical DNA covering regions that we looked at in the first Artemis exercise. It is apparent that there is a backbone sequence shared with *E. coli* K12, plus chunks of *S. Typhi* specific DNA, which appear to be insertions relative to *E. coli* K12.

#### 5. More things to try out in ACT

1. Double click red boxes to centralise them.
2. Zoom right in to view the base pairs and amino acids of each sequence.
3. Load annotation files into the sequence view panels.
4. Use some of the other Artemis features e.g., graphs etc.
5. Find an inversion in one genome relative to the other then flip one of the sequences.

Once you have finished this exercise remember to close this ACT session down completely before starting the next exercise



## Exercise 2

### *P. falciparum* and *P. knowlesi*: Genome Comparison

#### Introduction

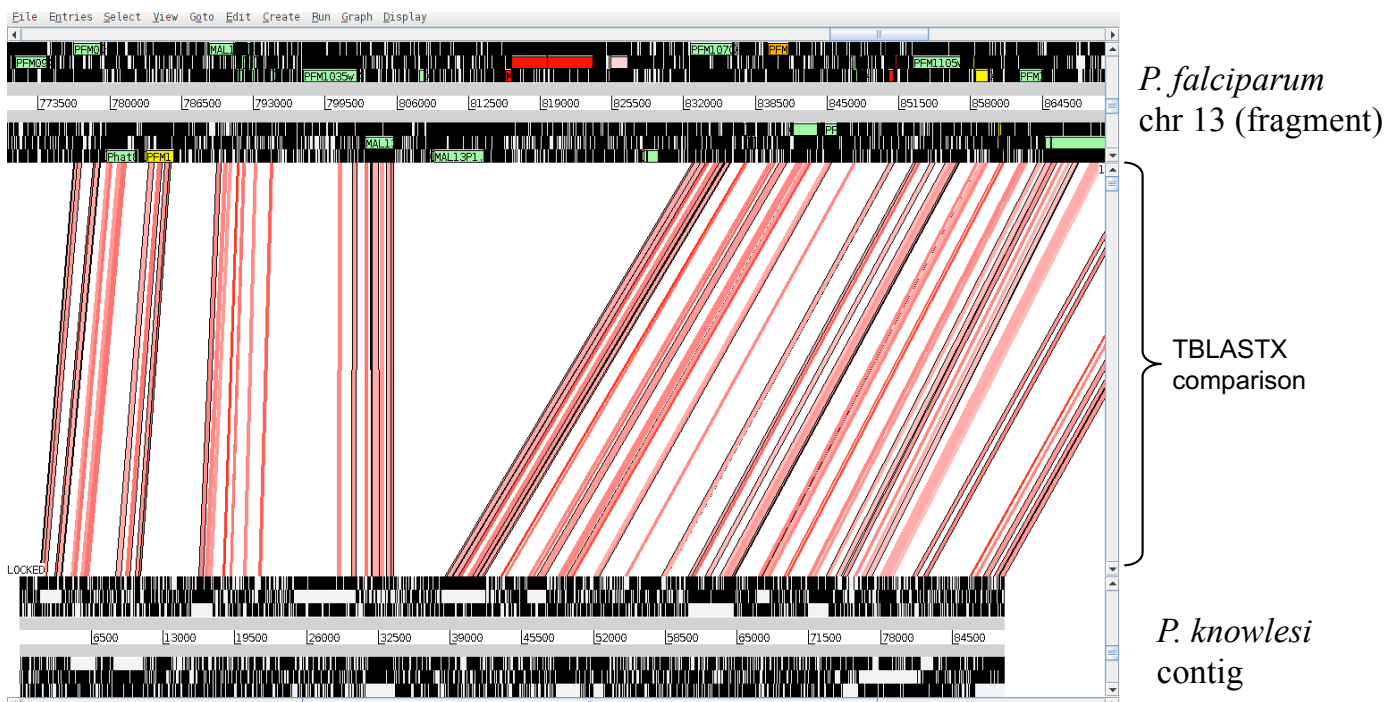
The annotation and analysis of the whole genome of *P. falciparum* 3D7 has been completed and genome sequences of several other malaria parasites are also available. This allows us to perform comparative analysis of the genomes of malaria parasites and understand the basic biology of their parasitism, based on the similarities / dissimilarities between the parasites at DNA / protein level.

#### Aim

You will be looking at the comparison between a genomic DNA fragment of the primate malaria *P. knowlesi* and the previously annotated chromosome 13 of *P. falciparum*. By comparing the two genomic sequences you will be able to study the degree of conservation of gene order (i.e., synteny) and identify genes in *P. knowlesi* genome. As part of the exercise you will also identify an unique region between the two genomic fragments and finally modify the gene model of a multi-exon gene in *P. knowlesi*, using ACT.

The files that you are going to need are:

- Pfal\_chr13.embl.gz - *P. falciparum* annotation file with sequence
- Pknowlesi\_contig.seq.gz - *P. knowlesi* DNA file (without annotation)
- Pknowlesi\_contig.embl.gz - *P. knowlesi* annotation file
- Plasmodium\_comp.crunch.gz - TBLASTX comparison file

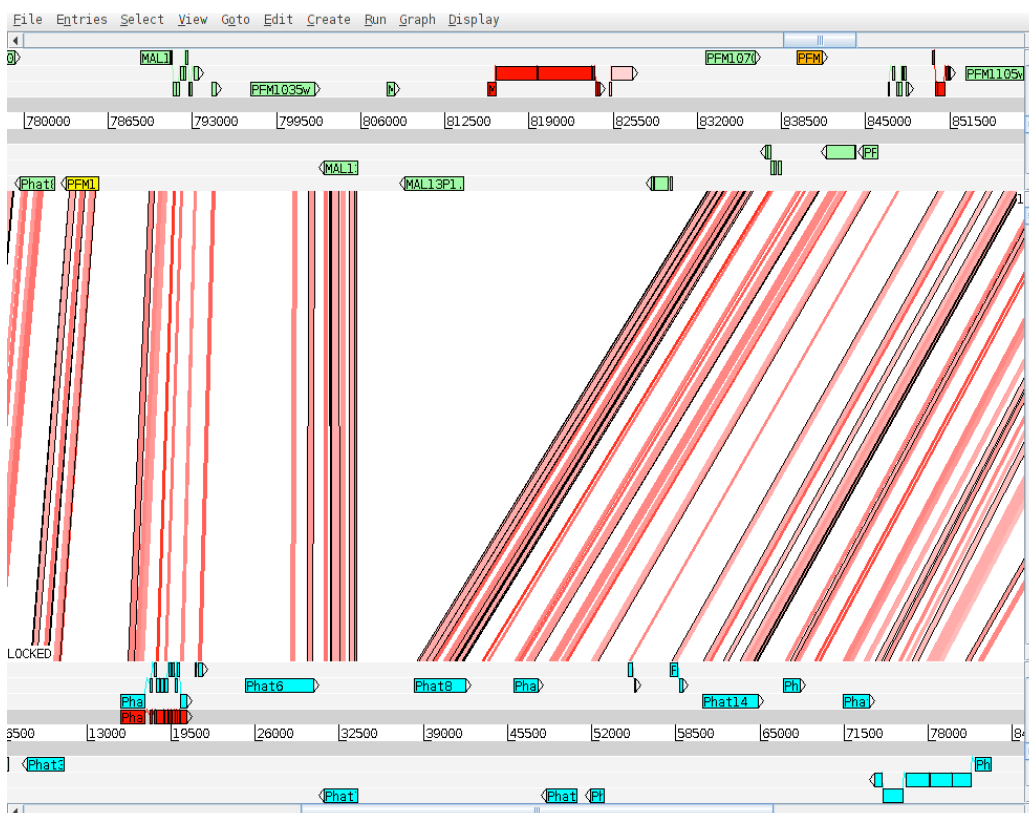


Comparison of *P. knowlesi* contig and the annotated chromosome 13 fragment of *P. falciparum*

## Exercise 2 Part I

### Conservation of gene order (synteny)

- In the ACT start up window load up the files Pfal\_chr13.embl.gz, Pknowlesi\_contig.seq.gz and the comparison file Plasmodium\_comp.crunch.gz. Add the annotation file 'Pknowlesi\_contig.embl.gz' to the Pknowlesi\_contig.seq.gz sequence.
- Use the slider on either sequence view panel to obtain a global view of the sequence comparison. Also use the slider on the comparison view panel to remove the 'shorter' similarity hits. What effects does this have?
- Can you see conserved gene order between the two species?
- Can you see any region where similarity is broken up? Zoom in and look at some of the genes encoded within this unique region in file: Pfal\_chr13.embl.gz (top sequence)
- Example location: Pfal\_chr13.embl.gz, 815823..829969
- What are the predicted products of the genes assigned to this unique location? View the details by clicking on the feature, and then select 'Edit selected feature' from the 'Edit' menu after selecting the appropriate CDS feature.
- Can you identify genes in conserved regions that have not been annotated in the *P. knowlesi* contig, but are present in the *P. falciparum* chromosome 13? This will allow you to see any potential protein coding regions.
- Any thoughts about the possible biological relevance of the comparison?



*P. falciparum*  
Pfal\_chr13.embl.gz

*P. knowlesi*  
Pknowlesi\_contig.embl.gz



## Exercise 2 Part II

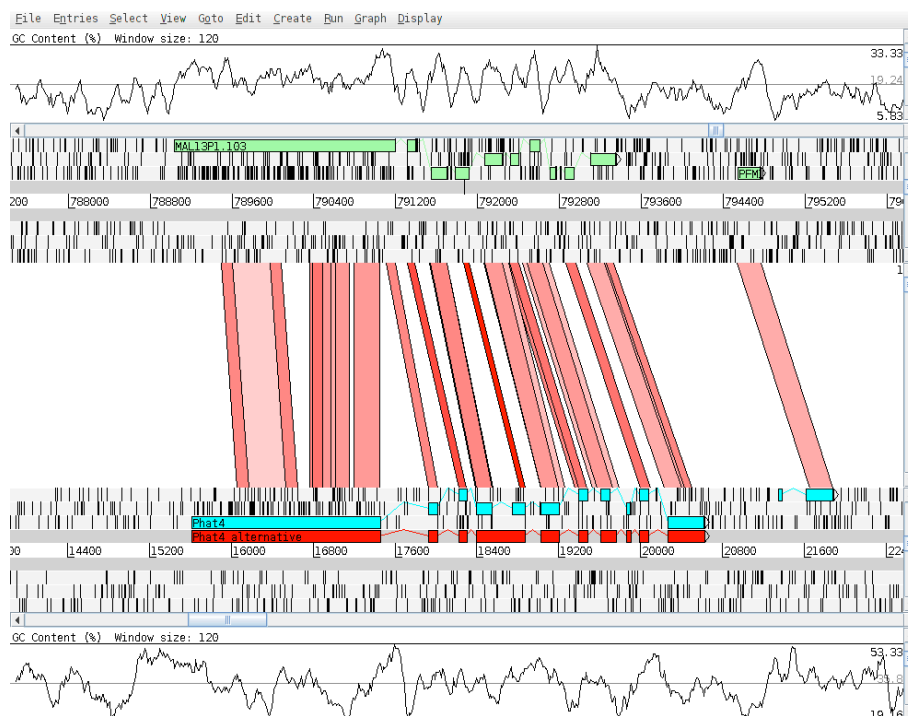
### Prediction of gene models:

There are several computer algorithms, covered earlier in the course, that predict gene models, based on training the algorithm with previously known gene sets with previously known experimentally verified exon-intron structures (in eukaryotes). However, no single programme can predict the gene structure with 100% accuracy and one needs to curate / refine the gene models, generated by automated predictions. We have generated automated gene models for the *P. knowlesi* contig, using PHAT (Pretty Handy Annotation Tool, a gene finding algorithm, see in Mol. Biochem. Parasitol. 2001 Dec; 118(2):167-74) and the automated annotations are saved in Pknowlesi\_contig.embl.gz.

- Zoom into the *P. falciparum* gene labelled PFM1010w shown below. Can you compare the 2 gene models and identify the conserved exon(s) between the 2 species?
- Use the slider on the comparison view panel to include some 'shorter' similarity hits. Can you now identify all the conserved exons of the PFM1010w orthologue in the *P. knowlesi* contig? (For the time being, disregard the misc\_feature for 'Phat4', coloured in red in the 'Pknowlesi\_contig.embl.gz' file)
- Open the 'GC Content (%)' window from 'graph' menu for both the entries. Can you relate the exon-intron boundaries to GC-content for the *P. falciparum* gene labelled PFM1010w? Is it also applicable to the gene model 'Phat4' in the *P. knowlesi* contig?
- Example regions:

**Pfal\_chr13.embl.gz, 789034..793351**

**Pknowlesi\_contig.embl.gz, 15618..20618**



*P. falciparum*  
Pfal\_chr13.embl.gz

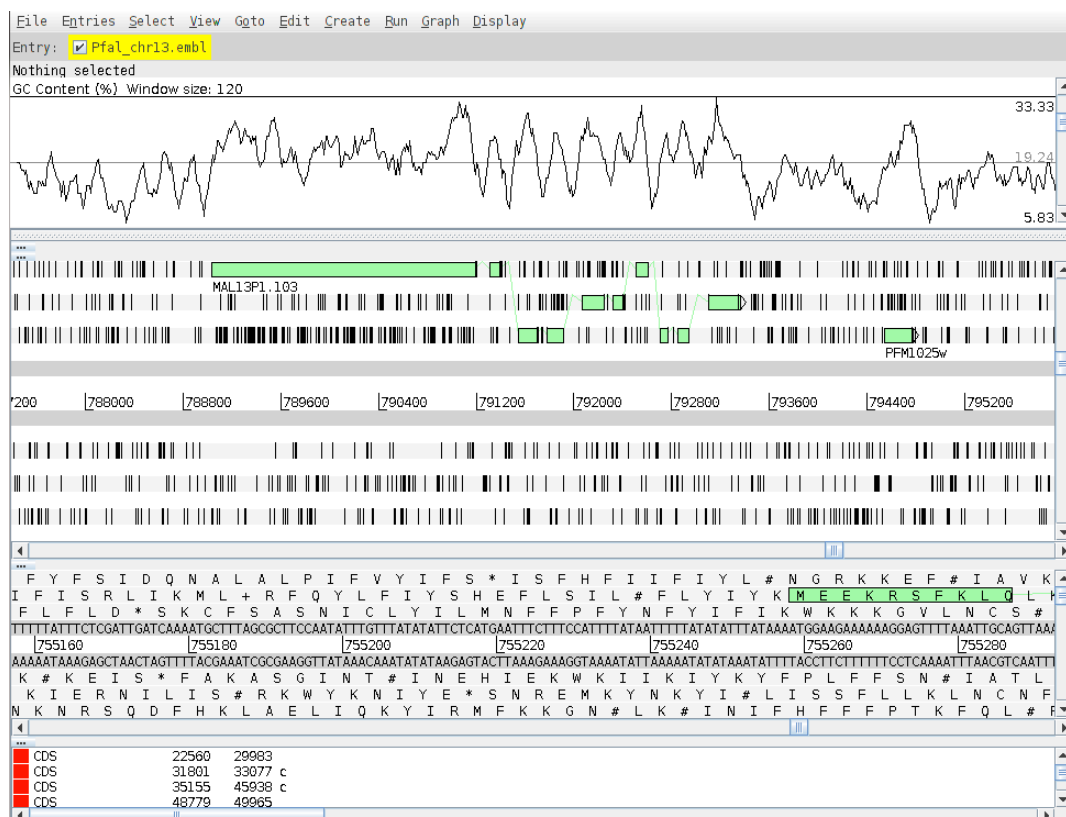
*P. knowlesi*  
Pknowlesi\_contig.embl.gz

Comparison between orthologous genes in *P. falciparum* and *P. knowlesi*

## Exercise 2 Part III (OPTIONAL)

### Gene models for multi-exon genes in *P. falciparum*:

- Use 'File' menu to select entry 'Pfal\_chr13.embl.gz' and select 'Edit In Artemis' to bring up an Artemis window.
- In Artemis window, use 'Graph' menu and switch 'on' the 'GC Content (%)' window.
- Use 'Goto' menu to select 'Navigator' window and within the Navigator window, select 'Goto Feature With This Qualifier Value' and type 'PFM1010w', click then close the dialogue box.
- Go through the annotated gene model for 'PFM1010w' and have a look at the the exon-intron boundaries and compare with the splice site sequences from *P. falciparum* given in **Appendix XI**.
- Also have a glance through a few other gene models for multi-exon genes and have a look at the intron sequences as well. Can you find any common pattern in the putative intron sequences? Hint – look at the complexity of the sequence
- You can delete exon(s) of any gene by selecting the exon(s) and then choosing 'Delete Selected Exons' from 'Edit' menu. Similarly, you can add an exon to a particular gene by co-selecting the exon and the gene (CDS features) followed by selecting 'Merge Selected Features' from the 'Edit' menu.
- Example regions:  
**Pfal\_chr13.embl.gz, 789034..793351, 657638..660023, 672361..673753**



Example location: 789034..793351, in Pfal\_chr13.embl.gz

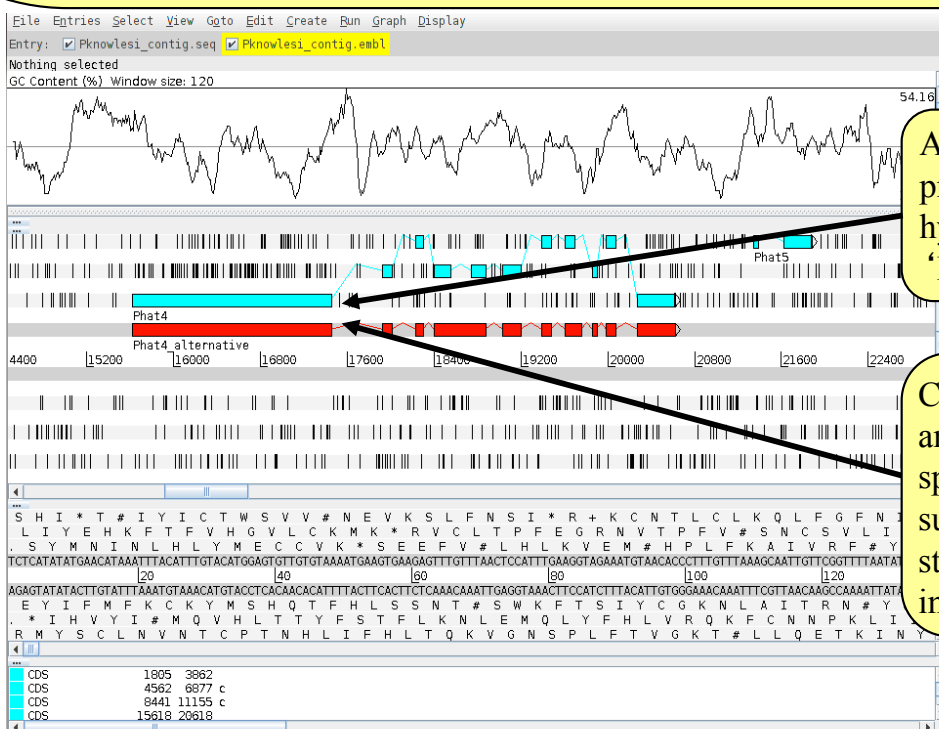
## Exercise 2 Part IV (OPTIONAL)

### Curation of gene models in *P. knowlesi*:

We are now going to edit the gene model for *P. knowlesi*.

- Use 'File' menu from the ACT displaying *P. falciparum* and *P. knowlesi* to select entry 'Pknowlesi\_contig.embl.gz' and select 'Edit In Artemis' to bring up an Artemis window.
- Within the Artemis window, use 'Graph' menu and switch 'on' the 'GC Content (%)' window.
- Use 'Goto' menu to select 'Navigator' window and within the Navigator window, select 'Goto Feature With This Qualifier Value' and type 'Phat4'.
- Go to the first ACT window (first small window that appears when starting up ACT), and use the 'Options' menu to select 'Enable Direct Editing'.
- Go through the gene model of 'Phat4' and have a glance through the exon-intron boundaries. Can you suggest any alternative gene model, after consulting the Table provided in **Appendix XI**, containing several examples of experimentally verified splice site sequences for *P. falciparum*?
- Example modifications:

Have a look at the 'misc\_feature', coloured in red (location: 15618..20618). Can you spot any difference in the red gene model of 'Phat4' at the exon-intron boundaries? Select the red feature, click on 'Edit' menu and select 'Edit Selected Features' and in the new window that pops out, change the 'Key' from misc\_feature to 'CDS' and click on 'OK' button to close the window. Now you can compare the automatically created blue gene model and the curated red gene models at protein level and predict any alternative splicing pattern.



Automated gene prediction for hypothetical gene 'Phat4'

Can you suggest any alternative splicing pattern such as the gene structure shown in red?

Example location: 15618..20618, in Pknowlesi\_contig.embl.gz

## Exercise 3

### Introduction

Having familiarised yourselves with the basics of ACT, we are now going to use it to look at a region of synteny between *T. brucei* and *Leishmania*.

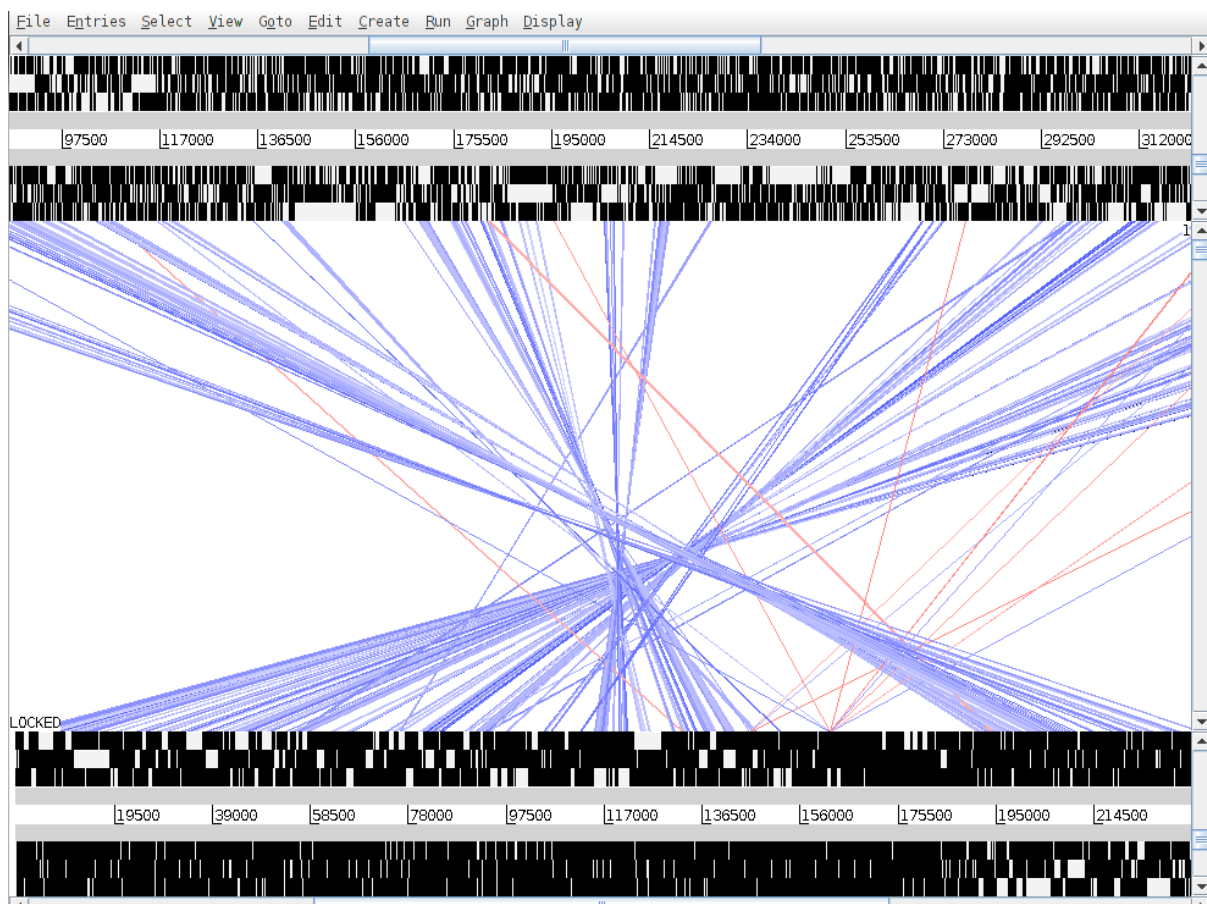
### Aim

By looking at a comparison of the annotated sequences of *T. brucei* and *L. major* you will be able to analyse, in detail, those genes that are found in both organisms as well as spot the differences. You will also see how ACT can be used to study the different chromosome architecture of these two parasite species.

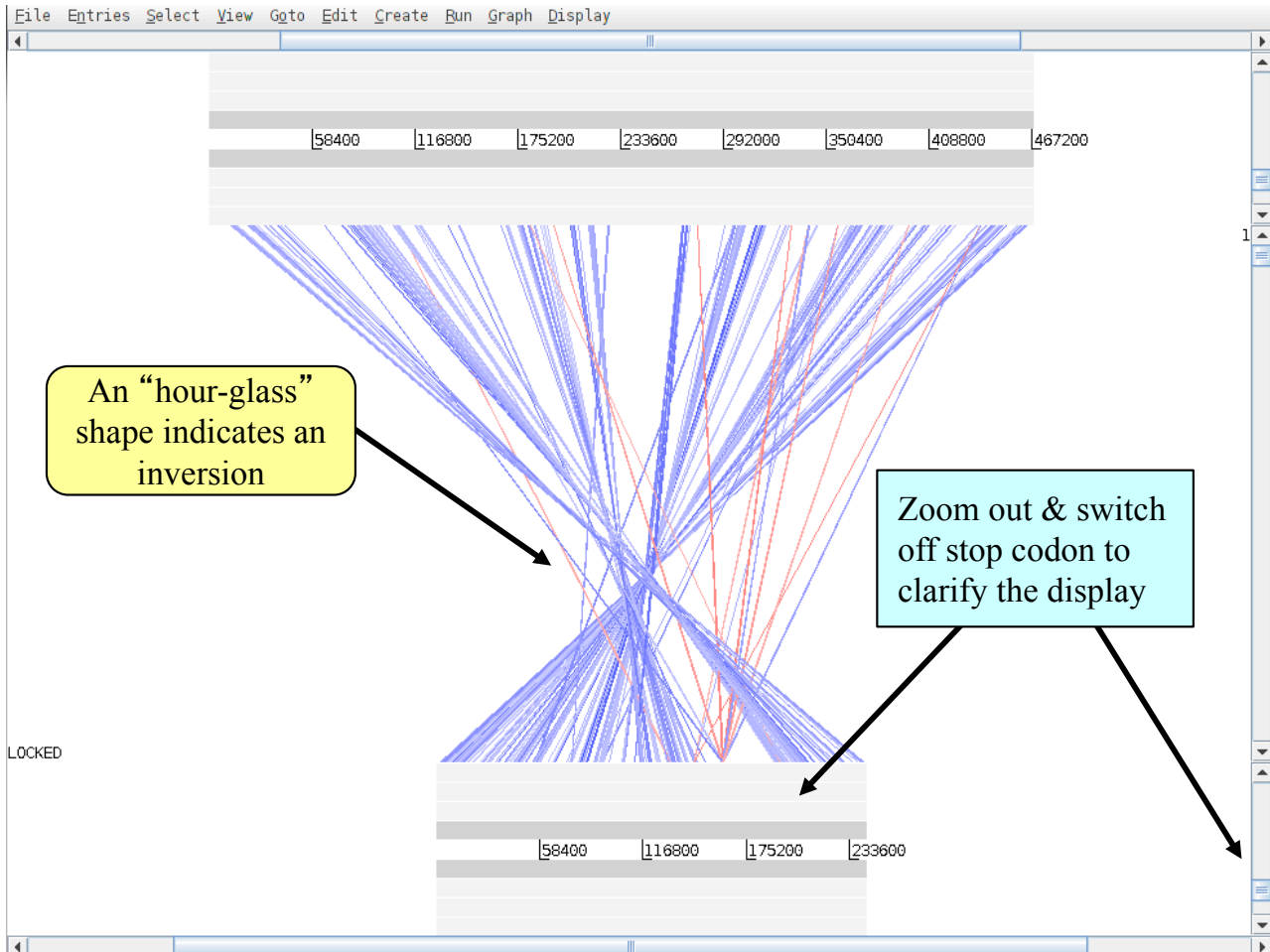
The files that you are going to need are:

sequence	Tbrucei.dna.gz	- <i>T. brucei</i>
	Tbrucei.embl.gz	- <i>T. brucei</i> annotation
	Leish_vs_Tbrucei.tblastx.gz	- comparison file
	Leish.dna.gz	- <i>L. major</i> sequence
	Leish.embl.gz	- <i>L. major</i> annotation

First, load up the sequence files for *T. brucei* and *L. major* and the comparison file in ACT.



Next, you need to find the regions of synteny between the sequences.



When you have determined where there is synteny, zoom in to the region for a detailed look. At this point you can add the annotation from the files called **Leish.embl.gz** and **Tbrucei.embl.gz**.

Can you see conserved gene order between the two species?

Can you see any region where similarity is broken up? Zoom in and look at some of the genes encoded within these regions.

What are the predicted products of the genes assigned to these locations? View the details by clicking on the feature, and then select *'Edit selected feature'* from the *'Edit'* menu after selecting the appropriate CDS feature.

Can you identify any genes in one organism that have hits to, or are similar to, regions in the other organism but which don't appear to be predicted? If so, add these to your annotation.





## Exercise 4

### Introduction

If you do not have access to BLAST software running on a local computer, there is a web resource WebACT (**Appendix VI** for the URL) that can be used for generating ACT comparison files. WebACT allows you to cut and paste, or upload, your own sequences, and generate ACT readable BLASTN or TBLASTX comparison files. WebACT also has a large selection of recomputed comparison files for bacterial genomes, which can be downloaded along with the EMBL sequence entries and viewed in ACT.

For the purposes of this exercise we are going to focus on the Gram-negative bacterial pathogens *Burkholderia pseudomallei* and *Burkholderia mallei*. Both of these organisms are category B bio-threat agents and cause the diseases Melioidosis and Glanders respectively. The two species are closely related (DNA-DNA identity is >99%, multi locus sequence typing (MLST) predicts that *B. mallei* is a clone of *B. pseudomallei*), however they differ markedly in the environmental niches that they occupy.

*B. pseudomallei* is found in S.E. Asia and northern Australia, and is prevalent in the soil in Melioidosis endemic areas. Inhalation, or direct contact with cuts or breaks in the skin, by soil-borne *B. pseudomallei* is the cause of Melioidosis in humans and higher mammals. In contrast, *B. mallei* is a zoonotic pathogen that is host restricted to horses and cannot be isolated from the environment. Comparative genomic analysis has provided insights into evolution of these two pathogens and the genetic basis for ecological and pathological differences of these two pathogens.

The genomes of these two organisms both consist of two circular chromosomes. Comparisons of the genomes reveals that the genome of *B. pseudomallei* is ~1.31 Mb larger than that of *B. mallei*; 16% of chromosome 1, and 32% of chromosome 2, are unique in *B. pseudomallei* with respect to *B. mallei*.

### Aim

You are going to use a web resource, WebACT, to generate a comparison file of the smaller chromosomes of *B. pseudomallei* and *B. mallei*. From the WebACT site you will download a pre-computed ACT comparison comparison file, along with the appropriate EMBL sequence and annotation files, which you will then open in ACT. Using this comparison you can then investigate some of the the genotypic differences that differentiate these closely related pathogens, and look for the basis of structural differences in these chromosomes. We have not provided files for this exercise - you are on your own.

Open up a web browser and go to the URL: [www.webact.org](http://www.webact.org)

**WebACT**

WebACT provides a database of sequence comparisons between all publicly available prokaryotic genome sequences, allowing the on-line visualisation of comparisons between up to five genomic sequences, using the **Artemis Comparison Tool (ACT)** developed by the Sanger Institute.

Sequence comparisons can also be generated 'on the fly' for up to five user-entered sequences, by either uploading sequences, or querying public databases using sequence identifiers.

All pre-computed and user-generated comparisons can be viewed on-line using a webstart version of ACT or can be downloaded.

WebACT currently contains 671 sequences from 273 species representing a total of 225456 comparisons.

Centre for Bioinformatics  
Department of Infectious Disease Epidemiology

WebAct has been developed and is hosted at Imperial College London  
ACT is developed and maintained by the Sanger Centre Pathogen Sequencing Unit

Click on the Pre-computed tab.

The 'Pre-computed' page contains genomic sequences that have been compared using BLASTN to each other. By selecting the desired sequences from the sequence lists, the appropriate sequence and comparison files can be downloaded

WebACT can display pairwise comparison between up to 5 sequences. Click here if you want to increase the number from the default of 2.

In addition to the chromosome sequences, plasmids can also be displayed by clicking in this box

**WebACT** | Select Sequences

How many sequences do you wish to compare?

Show plasmid sequences?

Please select your sequences from the lists below

**Sequence 1**

- Acidobacterium bacterium (strain Elin345) chromosome. (CP000360)
- Acinetobacter sp. (strain ADP1) chromosome (CR543861)
- Aeropyrum pernix (strain K1) chromosome (BA000002)
- Agrobacterium tumefaciens (strain C58 / ATCC 33970, sub\_strain Cereon) circular chromosome (AE007869)
- Agrobacterium tumefaciens (strain C58 / ATCC 33970, sub\_strain Cereon) linear chromosome (AE007870)

**Sequence 2**

- Acidobacterium bacterium (strain Elin345) chromosome. (CP000360)
- Acinetobacter sp. (strain ADP1) chromosome (CR543861)
- Aeropyrum pernix (strain K1) chromosome (BA000002)
- Agrobacterium tumefaciens (strain C58 / ATCC 33970, sub\_strain Cereon) circular chromosome (AE007869)
- Agrobacterium tumefaciens (strain C58 / ATCC 33970, sub\_strain Cereon) linear chromosome (AE007870)

You are going to compare the smaller chromosomes of *B. pseudomallei* and *B. mallei*.

In the **Sequence 1** list select *Burkholderia pseudomallei* chromosome 2 (accession number BX571966)

In the **Sequence 2** list select *Burkholderia mallei* chromosome 2 (accession number CP000011)

Once you have selected the sequences click the **Next** button

In this window you can specify the regions in the selected sequences to generate the comparison over. It is possible to query the sequences on gene name or coordinates. The default setting is for the whole sequence, and this is what we want for this exercise as you are going to compare the whole chromosomes.

Click the **Next** button

Artemis [Running]

WebACT: Results - Mozilla Firefox

WebACT: Results

www.webact.org/WebACT/prebuilt

Pre-computed Generate Reload Instructions

WebACT

WebACT | Select Sequences | Select Region | Results [Contact us](#)

Results

Overview of Selection (Mouse over for sequence details)

BX571966

CP000011

View Comparison -

Open overview in separate window:

Show hits outside selected sequence

Select e-value cut-off: 0.01

If your browser asks you either Open or Save a .jnlp file, select 'Open' to view the comparison

Start ACT Download files Back

In the **Overview of Selection** you can see a schematic representation of the relative size of the two sequence that have been chosen to be compared.

The Expect (E) value cut-off can be changed in this box. The default value is 0.01, but the range is from 10.0 to 0.0001.

Click the **Download files** button

In addition to downloading the comparison files and sequence file it is also possible to view the comparison in a webstart version of ACT. This will run locally on your machine and does not require ACT to be previously loaded, as a webstart version of ACT will be included in the download. You are not going to use this option in this exercise.

The comparison file and sequences files will contained in a folder. For the ease of downloading the folder is zipped.

WebACT: Download - Mozilla Firefox

www.webact.org/WebACT/download?CGISESSID=99e470e44654f2d8363d28929914

WebACT

WebACT Comparison

Download files -

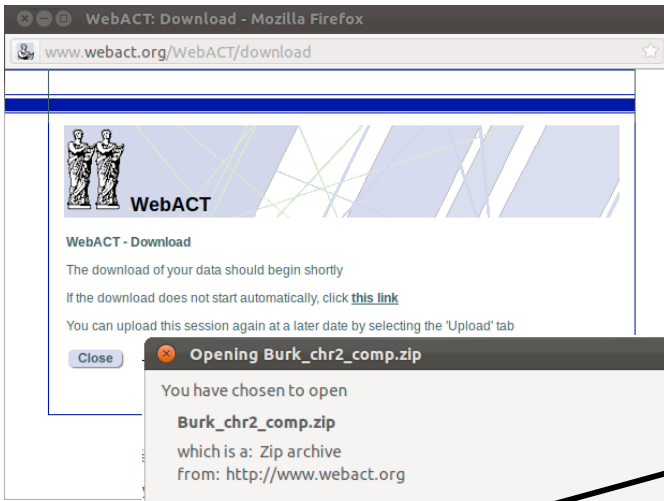
Enter filename: Burk\_chr2\_comp.zip

Include data for offline use:

Download files Close

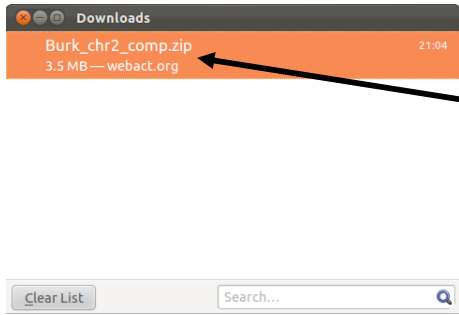
In the filename box you can type the file name of the zip file containing the sequence and comparison files. For this exercise call the file: **Burk\_chr2\_comp.zip**

Click the **Download files** button



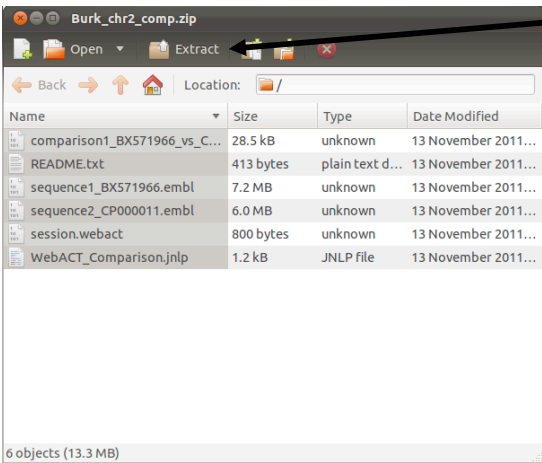
You may get a window appearing asking you what Firefox should do with the Burk\_chr2comp.zip file? Save the file to disk.

Click the **OK** button



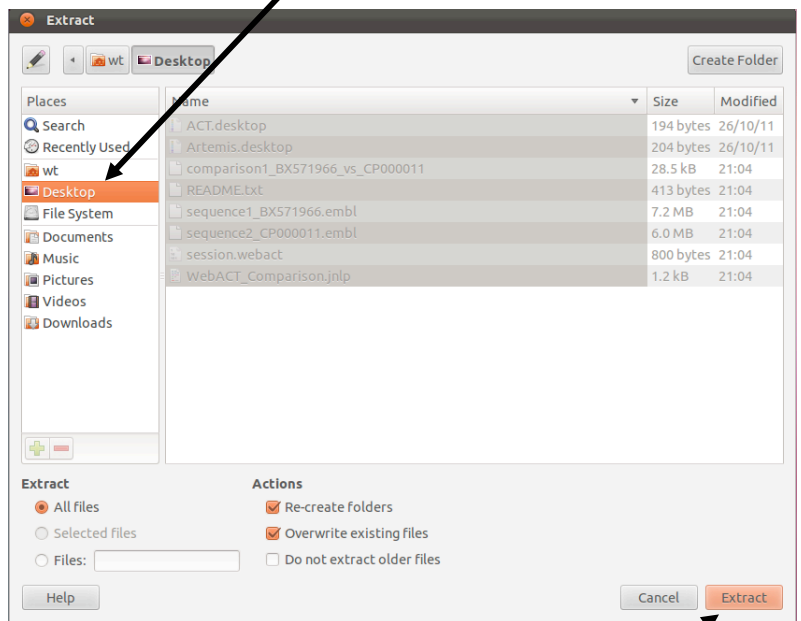
Burk\_chr2\_comp.zip should now be in the **Downloads** directory

To unzip the file, double click with the left mouse button on the file name



Click the **Extract** button

Select a location to extract the files to, such as **Desktop**

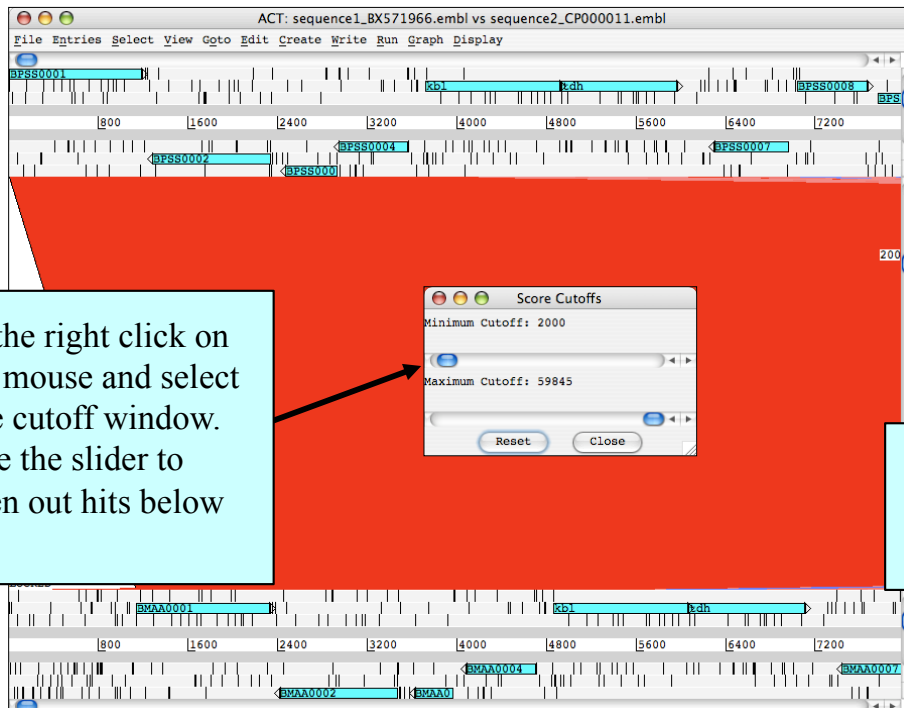


Click the **Extract** button

The files contained in the unzipped directory should include: comparison1\_BX571966\_vs\_CP000011, sequence1\_BX571966.embl and sequence2\_CP000011.embl. These are the ACT comparison file and the *B. pseudomallei* and *B. mallei* chromosome 2 EMBL annotation and sequence files respectively.



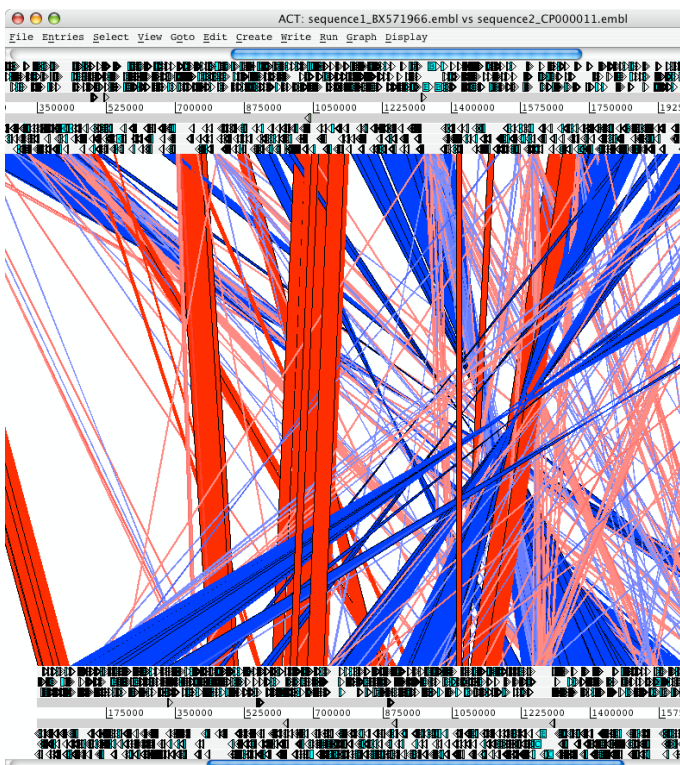
Open up ACT, and load up the comparison (comparison1\_BX571966\_vs\_CP000011) along with the two EMBL sequence and annotation files (sequence1\_BX571966.embl and sequence2\_CP000011.embl). If you get a warnings window asking if you want to read warning, click **No**.



Use the right click on your mouse and select score cutoff window. Move the slider to screen out hits below 2000

Move the slider to 200 to show only BLASTN matches greater than 200

Now remove the stop codons for both entries, and then zoom out you will see the overall conservation of the structure of the small chromosomes is poor.



If you were to look at the comparison for the large chromosomes you would see a similar picture. The lack of conservation is the result of intra-chromosomal rearrangements. What do you think caused this? Zoom into the regions on the edge of the rearranged matches and look at the annotation in the *B. mallei* chromosome.

What is the function of the CDSs consistently found in these regions. Are there matches in the *B. pseudomallei* chromosome?

Try selecting CDSs in *B. pseudomallei* that match these regions and look how many matches there are in *B. mallei*. Are these regions repeated throughout the chromosome?



If you have time, you may want to generate, and view in ACT, comparisons for your own sequences. If you do not have any loaded on your workshop computer, why not try and download some. Sequence in various formats can be cut and pasted, or up loaded onto the WebACT site. In addition, if you know the accession number of the sequence that you want to compare, you can use that. As the web site will have to run BLAST to generate your comparison file, you may want to limit the size of the sequence that you submit for this exercise to <100 kb. The the web site can handle larger sequences, but it will just take longer.

The screenshot shows the WebACT 'Enter Query' page in a Mozilla Firefox browser. The page has a navigation bar with tabs: 'Pre-computed', 'Generate', 'Reload', and 'Instructions'. The 'Generate' tab is selected. The main content area includes a 'WebACT | Enter Query' header, a 'How many sequences to you wish to compare?' dropdown menu (set to 2), an email notification checkbox, and an email address field. Below this, there are two sequence input sections, 'Sequence 1' and 'Sequence 2'. Each section has three options: 'Paste sequence (raw, EMBL or FASTA format)', 'Upload File (raw, EMBL or FASTA format)', and 'Enter an EMBL or Refseq Accession number'. The 'Paste sequence' option is selected for both. At the bottom, there is a 'Blast Search Options [show]' link and 'Submit' and 'Clear' buttons.

Annotations on the screenshot:

- Clicking on the 'Generate' tab will take you to this page**: Points to the 'Generate' tab in the navigation bar.
- Number of sequences to compare**: Points to the dropdown menu showing '2'.
- Cut and paste sequence**: Points to the text input field for 'Sequence 1'.
- Upload file**: Points to the 'Browse...' button for 'Sequence 1'.
- Type accession number**: Points to the text input field for 'Sequence 1'.
- Click here for BLAST options, such as changing from the default BlastN to TblastX, and altering the BLAST cutoffs**: Points to the 'Blast Search Options [show]' link.
- Once you added the relevant sequence information, submit your query. The comparison file or files are down loaded as shown in the example, and can them be loaded in to ACT.**: Points to the 'Submit' button.

## 0 Unix module

### 0.1 Introducing Unix

Unix is the standard operating system on most large computer systems in scientific research, in the same way that Microsoft Windows is the dominant operating system on desktop PCs.

Unix and MS Windows both perform the important job of managing the computer's hardware (screen, keyboard, mouse, hard disks, network connections, etc...) on your behalf. They also provide you with tools to manage your files and to run application software. They both offer a graphical user interface (desktop). These desktop interfaces look different between the operating systems, use different names for things (e.g. directory versus folder) and have different images but they mostly offer the same functionality.

Unix is a powerful, secure, robust and stable operating system which allows dozens of people to run programs on the same computer at the same time. This is why it is the preferred operating system for large-scale scientific computing. It runs on all kinds of machines, from mobile phones (Android), desktop PCs... to supercomputers.

### 0.2 Why Unix?

Increasingly, the output of biological research exists as *in silico* data, usually in the form of large text files. Unix is particularly suitable for working with such files and has several powerful and flexible commands that can be used to process and analyse this data. One advantage of learning Unix is that many of the commands can be combined in an almost unlimited fashion. So if you can learn just six Unix commands, you will be able to do a lot more than just six things.

Unix contains hundreds of commands, but to conduct your analysis you will probably only need 10 or so to achieve most of what you want to do. In this course we will introduce you to some basic Unix commands followed by some more advanced commands and provide examples of how they can be used in bioinformatics analyses.

### 0.3 Sections of the Unix course

1. Basic unix
2. Files
3. grep
4. awk
5. Bash scripts

### 0.4 Following the course in a terminal

In this course you will use a terminal window to type in your Unix commands. This is similar to the "Command Prompt" window on MS Windows systems, which allows the user to type DOS commands to manage files.

To follow the course by typing the commands into a terminal yourself, run the following cell in this notebook, either by pressing control and enter, or by selecting it with the mouse and then in the menu at the top of the page choosing Cell -> Run.



```
echo cd $PWD
```

```
cd /home/manager/Module_3_Linux_Scripting
```

Now open a new terminal on your computer and type the command that was output by the previous cell followed by the enter key. The command will look similar to this:

```
cd /home/manager/Module_3_Linux_Scripting
```

## 0.5 Cheat sheet

We've also included a [cheat sheet](#). It probably won't make a lot of sense now, but it might be a useful reminder of this module later in the course.

## 1 Basic Unix

### 1.1 The Commandline

The commandline or 'terminal' is an interface you can use to run programs and analyse your data. If this is your first time using one it will seem pretty daunting at first but, with just a few commands, you'll start to see how it helps you to get things done much quicker. You're probably more familiar with software which uses a graphical user interface, also known as a GUI; unfortunately most of the best bioinformatics software has not been programmed with this capability.

### 1.2 Getting started

Before we get started, let's check that you're in the right place. Please click on the cell below and press the `ctrl` and `Enter` keys. If you're not sure what this command does, don't worry for now; we'll explain it in more detail later.



```
echo "cd $PWD"
```

```
cd /home/manager/Module_3_Linux_Scripting/basic
```

It should say something like `cd /home/manager/Module_3_Linux_Scripting/basic`. Type whatever it said into your terminal and press `Enter`.

Then continue through the course, entering any commands that you encounter into your terminal window.

However, before getting started there are some general points to remember that will make your life easier:

- Unix is case sensitive - typing `ls` is not the same as typing `LS`.
- Often when you have problems with Unix, it is due to a spelling mistake. Check that you have not missed or added a space. Pay careful attention when typing commands across a couple of lines.

### 1.3 Files and directories

*Directories* are the Unix equivalent of folders on a PC or Mac. They are organised in a hierarchy, so directories can have sub-directories and so on. Directories are very useful for organising your work and keeping your account tidy - for example, if you have more than one project, you can organise the files for each project into different directories to keep them separate. You can think of directories as rooms in a house. You can only be in one room (directory) at a time. When you are in a room you can see everything in that room easily. To see things in other rooms, you have to go to the appropriate door and crane your head around. Unix works in a similar manner, moving from directory to directory to access files. The location or directory that you are in is referred to as the current working directory.

If there is a file called `genome.seq` in the `dna` directory its location or full pathname can be expressed as `/nfs/dna/genome.seq`.

### Directory structure example

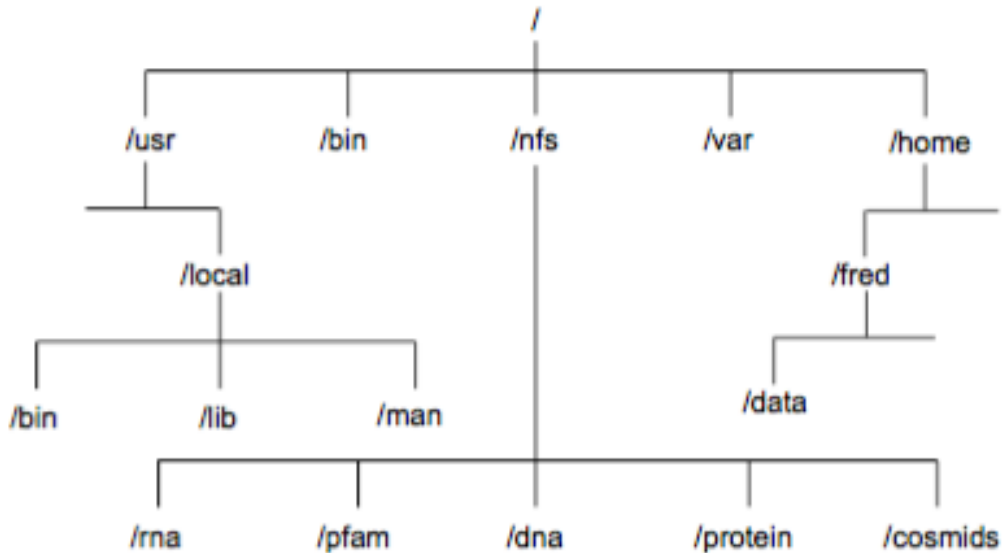


Figure 1: Hierarchy

## 1.4 pwd - find where you are

The command `pwd` stands for print working directory. A *command* (also known as a *program*) is something which tells the computer to do something. Commands are therefore often the first thing that you type into the terminal (although we'll show you some advanced exceptions to this rule later).

As described above, directories are arranged in a hierarchical structure. To determine where you are in the hierarchy you can use the `pwd` command to display the name of the current working directory. The current working directory may be thought of as the directory you are in, i.e. your current position in the file-system tree.

To find out where you are, type this into your terminal.



```
pwd
```

```
/home/manager/Module_3_Linux_Scripting/basic
```

Remember that Unix is case sensitive, `PWD` is not the same as `pwd`.

`pwd` will list each of the folders you would need to navigate through to get from the root of the file system to your current directory. This is sometimes referred to as your 'absolute path' to distinguish that it gives a complete route rather than a 'relative path' which tells you how to get from one folder to another. More on that shortly.

## 1.5 ls - list the contents of a directory

The command `ls` stands for list. The `ls` command can be used to list the contents of a directory.

To list the contents of your current working directory use:



```
ls
```

```
basic.ipynb directory_structure.png Pfalciparum Styphi
```

You should see that there are 6 items in this directory.

To list the contents of a directory with extra information about the items use:



```
ls -l
```

```
total 56
-rwxrwx--- 1 manager manager 19478 Dec 22 09:59 basic.ipynb
-rwxrwx--- 1 manager manager 28513 Dec 22 09:59 directory_structure.png
drwxrwx--- 4 manager manager 4096 Dec 22 09:59 Pfalciparum
drwxrwx--- 2 manager manager 4096 Dec 22 09:59 Styphi
```

Instead of printing out a simple list, this should have printed out additional information about each file. Note that there is a space between the command `ls` and the `-l`. There is no space between the dash and the letter `l`.

`-l` is our first example of an *option*. Many commands have options which change their behaviour but are not always required.

What do each of the columns represent?

To list all contents of a directory including hidden files and directories use:



```
ls -a -l
```

```
total 64
drwxrwx--- 4 manager manager 4096 Dec 22 09:59 .
drwxrwxr-x 9 manager manager 4096 Dec 22 09:59 ..
-rwxrwx--- 1 manager manager 19478 Dec 22 09:59 basic.ipynb
-rwxrwx--- 1 manager manager 28513 Dec 22 09:59 directory_structure.png
-rwxrwx--- 1 manager manager 0 Dec 22 09:59 .hidden1
-rwxrwx--- 1 manager manager 0 Dec 22 09:59 .hidden2
drwxrwx--- 4 manager manager 4096 Dec 22 09:59 Pfalciparum
drwxrwx--- 2 manager manager 4096 Dec 22 09:59 Styphi
```

This is an example of a command which can take multiple options at the same time. Different commands take different options and sometimes (unhelpfully) use the same letter to do different things.

How many hidden files and directories are there?

Try the same command but with the `-h` option:





```
ls -alh
```

```
total 64K
drwxrwx--- 4 manager manager 4.0K Dec 22 09:59 .
drwxrwxr-x 9 manager manager 4.0K Dec 22 09:59 ..
-rwxrwx--- 1 manager manager 20K Dec 22 09:59 basic.ipynb
-rwxrwx--- 1 manager manager 28K Dec 22 09:59 directory_structure.png
-rwxrwx--- 1 manager manager 0 Dec 22 09:59 .hidden1
-rwxrwx--- 1 manager manager 0 Dec 22 09:59 .hidden2
drwxrwx--- 4 manager manager 4.0K Dec 22 09:59 Pfalcciparum
drwxrwx--- 2 manager manager 4.0K Dec 22 09:59 Styphi
```

You'll also notice that we've combined `-a -l -h` into what appears to be a single `-alh` option. It's almost always ok to do this for options which are made up of a single dash followed by a single letter.

What does the `-h` option do?

To list the contents of the directory called `Pfalcciparum` with extra information use:



```
ls -l Pfalcciparum/
```

```
total 24472
drwxrwx--- 2 manager manager 4096 Dec 22 09:59 annotation
drwxrwx--- 2 manager manager 4096 Dec 22 09:59 fasta
-rwxrwx--- 1 manager manager 654069 Dec 22 09:59 MAL1.fa
-rwxrwx--- 1 manager manager 962943 Dec 22 09:59 MAL2.fa
-rwxrwx--- 1 manager manager 23241585 Dec 22 09:59 Malaria.fa
-rwxrwx--- 1 manager manager 183279 Dec 22 09:59 Pfalcciparum.bed
```

In this case we gave `ls` an *argument* describing the *relative path* to the directory `Pfalcciparum` from our current working directory. Arguments are very similar to options (and I often use the terms interchangeably) but they often refer to things which are not prefixed with dashes.

How many files are there in this directory?

## 1.6 Tab completion

Typing out file names is really boring and you're likely to make typos which will at best make your command fail with a strange error and at worst overwrite some of your carefully crafted analysis. *Tab completion* is a trick which normally reduces this risk significantly.

Instead of typing out `ls Pfalcciparum/`, try typing `ls P` and then press the tab character (instead of `Enter`). The rest of the folder name should just appear. If you have two folders with similar names (e.g. `my_awesome_scripts/` and `my_awesome_results/`) then you might need to give your terminal a bit of a hand to work out which one you want. In this case you would type `ls -l m`, when you press tab the terminal would read `ls -l my_awesome_`, you could then type `s` followed by another tab and it would work out that you meant `my_awesome_scripts/`

## 1.7 File permissions

Every file and directory have a set of permissions which restrict what can be done with a file or directory.

- Read (r): permission to read from a file/directory
- Write (w): permission to modify a file/directory
- Execute (x): Tells the operating system that the file contains code for the computer to run, as opposed to a file of text which you open in a text editor.

The first set of permissions (characters 2,3,4) refer to what the owner of the file can do, the second set of permissions (5,6,7) refers to what members of the Unix group can do and the third set of permissions (8,9,10) refers to what everyone else can do.

## 1.8 cd - change current working directory

The command `cd` stands for change directory.

The `cd` command will change the current working directory to another, in other words allow you to move up or down in the directory hierarchy.

To move into the `Styphi` directory type the following. Note, you'll remember this more easily if you type this into the terminal rather copying and pasting. Also remember that you can use tab completion to save typing all of it.



```
cd Styphi/
```

(no output)

Now use the `pwd` command to check your location in the directory hierarchy and the `ls` command to list the contents of this directory.



```
pwd  
ls
```

```
/home/manager/Module_3_Linux_Scripting/basic/Styphi  
Styphi.fa Styphi.gff Styphi.noseq.gff
```

You should see that there are 3 files called: `Styphi.fa`, `Styphi.gff`, `Styphi.noseq.gff`

## 1.9 Tips

There are some short cuts for referring to directories:

- `.` Current directory (one full stop)
- `..` Directory above (two full stops)
- `~` Home directory (tilde)
- `/` Root of the file system (like `C:` in Windows)

Try the following commands, what do they do?



```
ls .
```

```
Styphi.fa Styphi.gff Styphi.noseq.gff
```



```
ls ..
```

```
basic.ipynb directory_structure.png Pfalciparum Styphi
```



```
ls ~
```

```
bin                pathogen-informatics-training
Desktop            pathogens-vm
Documents          pathogens-vm.tar
Downloads          Pictures
Linux              Public
Module_1_ArtemisACT scripts
Module_2_Mapping  Task_1_Georeferencing
Module_3_Linux_Scripting Task_2_SexualDevelopment
Module_4_GenomeAssembly Templates
Module_5_Annot_DiffExp texmf
Music              Videos
```

Try moving between directories a few times. Can you get into the Pfalciparum/ and back into Styphi/?

## 1.10 cp - copy a file

The command `cp` stands for copy.

The `cp` command will copy a file from one location to another and you will end up with two copies of the file.

To copy the file `Styphi.gff` to a new file called `StyphiCT18` use:



```
cp Styphi.gff StyphiCT18.gff
```

```
(no output)
```

Use `ls` to check the contents of the current directory for the copied file:



```
ls
```

```
StyphiCT18.gff Styphi.fa Styphi.gff Styphi.noseq.gff
```

## 1.11 mv - move a file

The `mv` command stand for move.

The `mv` command will move a file from one location to another. This moves the file rather than copies it, therefore you end up with only one file rather than two. When using the command, the path or pathname is used to tell Unix where to find the file. You refer to files in other directories by using the list of hierarchical names separated by slashes. For example, the file called `bases` in the directory `genome` has the path `genome/bases`. If no path is specified, Unix assumes that the file is in the current working directory.

To move the file `StyphiCT18.gff` from the current directory to the directory above use:



```
mv StyphiCT18.gff ..
```

(no output)

Use the `ls` command to check the contents of the current directory and the directory above to see that `Styphi.fa` has been moved.



```
ls
```

```
Styphi.fa Styphi.gff Styphi.noseq.gff
```



```
cd ..  
ls
```

```
basic.ipynb directory_structure.png Pfalciparum Styphi StyphiCT18.gff
```

## 1.12 rm - delete a file

The command `rm` stands for remove.

The `rm` command will delete a file permanently from your computer so take care!

To remove the copy of the *S. typhi* genome file, called `StyphiCT18.gff` use:



```
rm StyphiCT18.gff
```

(no output)

Use the `ls` command to check the contents of the current directory to see that the file `StyphiCT18.gff` has been removed.



```
ls
```

```
basic.ipynb directory_structure.png Pfalciparum Styphi
```

Unfortunately there is no "recycle bin" on the command line to recover the file from, so you have to be careful.

### 1.13 find - find a file

The `find` command can be used to find files matching a given expression. It can be used to recursively search the directory tree for a specified name, seeking files and directories that match the given name.

To find all files in the current directory and all its subdirectories that end with the suffix `gff`:



```
find . -name "*.gff"
```

```
./Styphi/Styphi.gff  
./Styphi/Styphi.noseq.gff
```

How many `gff` files did you find?

To find all the subdirectories contained in the current directory use:



```
find . -type d
```

```
.  
./Pfalcciparum  
./Pfalcciparum/annotation  
./Pfalcciparum/fasta  
./Styphi
```

How many subdirectories did you find?

These are just two basic examples of the `find` command but it is possible to use the following `find` options to search in many other ways:

- `-mtime` : search files by modifying date
- `-atime` : search files by last access date
- `-size` : search files by file size
- `-user` : search files by user they belong to

### 1.14 Exercises

Many people panic when they are confronted with a Unix prompt! Don't! All the commands you need to solve these exercises are provided above and don't be afraid to make a mistake. If you get lost ask a demonstrator. If you are a person skilled at Unix, be patient this is only a short exercise.

To begin, open a terminal window and navigate to the `basic` directory in the `Unix_course` directory (remember use the Unix command `cd`) and then complete the exercise below.

1. Use the `ls` command to show the contents of the `basic` directory.
2. How many files are there in the `Pfalcciparum` directory?
3. What is the largest file in the `Pfalcciparum` directory?
4. Move into the `Pfalcciparum` directory.
5. How many files are there in the `fasta` directory?
6. Copy the file `Pfalcciparum.bed` in the `Pfalcciparum` directory into the `annotation` directory.
7. Move all the `fasta` files in the directory `Pfalcciparum` to the `fasta` directory.

8. How many files are there in the `fasta` directory?
9. Use the `find` command to find all `gff` files in the `Unix` directory, how many files did you find?
10. Use the `find` command to find all the `fasta` files in the `Unix` directory, how many files did you find?



## 2 Looking inside files

A common task is to look at the contents of a file. This can be achieved using several different Unix commands, `less`, `head` and `tail`. Let us consider some examples.

But first, change directory into the `Unix/files/` directory (hint: you might need to go up a few directories first using `cd ../../`). Check that the following commands give you a similar output:



```
pwd
ls
```

```
/home/manager/Module_3_Linux_Scripting/files
files.ipynb Pfalciparum.bed Styphi.fa Styphi.gff Styphi.noseq.gff
```

### 2.1 less

The `less` command displays the contents of a specified file one screen at a time. To test this command, open a terminal window on the computer, navigate to the directory `files` in the `Unix_course` directory and type the following command followed by the enter key:

```
less Styphi.gff
```

The contents of the file `Styphi.gff` is displayed one screen at a time, to view the next screen press the space bar. As `Styphi.gff` is a large file this will take a while, therefore you may want to escape or exit from this command. To do this, press the `q` key, this kills the `less` command and returns you to the Unix prompt. `less` can also scroll backwards if you hit the `b` key. Another useful feature is the slash key, `/`, to search for an expression in the file. Try it, search for the gene with locus tag `t0038`. What is the start and end position of this gene?

### 2.2 head and tail

Sometimes you may just want to view the text at the beginning or the end of a file, without having to display all of the file. The `head` and `tail` commands can be used to do this.

The `head` command displays the first ten lines of a file.

To look at the beginning of the file `Styphi.gff` file use:



```
head Styphi.gff
```

```
##gff-version 3
##sequence-region AE014613 1 4791961
#!Date 2011-07-11
#!Type DNA
#!Source-version EMBOSS 6.3.1
AE014613      EMBL  databank_entry 1      4791961 0.000 +      .      ID="AE014613.1";org
AE014613      EMBL  gene 190 255 0.000 +      .      ID="AE014613.2";gene="thrL"
AE014613      EMBL  CDS 190 255 0.000 +      0      ID="AE014613.3";codon_start
AE014613      EMBL  gene 337 2799 0.000 +      .      ID="AE014613.4";gene="thrA"
AE014613      EMBL  CDS 337 2799 0.000 +      0      ID="AE014613.5";codon_start
```

The `tail` command displays the last ten lines of a file.

To look at the end of `Styphi.gff` use:



```
tail Styphi.gff
```

```
tattgaggttttccacacccttgccgacgcgctccacgatgtggatttaccgtagcgac
gacagcccgcagccgggcaaaaatttcattactacgcttcgcccgctgaactggttccctt
attacaggaaaaatcacgctggatgctcatgccgcgctggtttttgccgtgaggattc
cggctgaccaacgacgagctggcgctggcgatgtattgaccggcgctgccgatggcggc
ggattacccttcgctcaatctgggtcaggcggtcatgggtattgctatcaattagcagg
ttaaagcaacagaccccgggaatccgcttgatattgctgatgaatcgagttacaggcggt
acgcgcgcccttttacgctgtaaccactctggaggcggccgatgaccacaaattaac
cgactggctacaacagcgaatcgccctgctgggacagcgagatacggcaatggtgcacg
tttggctcatgatattgaaaaaaaaactaacaataaacgtgtgtgaattttaaaataat
a
```

The amount of the file that is displayed can be increased by adding extra arguments. To increase the number of lines viewed from 10 to 25 add `-n 25` to the command:



```
tail -n 25 Styphi.gff
```

```
tgaatacatcatagccttcgcttcgaaaaacttttcaacgtgttgctgttaccaac
tcgtcttcaacgataagaatgtgcccggctgcatgtttgctacctaaattgccaactaa
atcgaaacaggaagtacaaaagtccttgacctgcctgatgcatgtcgcaattaacatga
tcggcgtaacatgactaaagtagtaattgcttctgatgcactttccatcaacgtcaa
caacatcattagccttggctgctgggtactttccctctggaccgacagtgtaaaaaacggc
tgtcatcctaaccattttaacgacaacataacaggtaagacgtaccggacacctaataa
aactacgcttcgcttgacatatacaagttcaattgtagcacgttaacagtttgatgaaat
catcgtagctaaatgctagccttcatcacaatttgcaatattccaactagttacgtaag
ccaactaataaatgcatgaatccaaagaacaggatctattttaaatataattatcctaa
ataaacagcaggataacgatgttctgttaacataaacagcaatagtacagatacgcaata
gtgtagcgtcttttacgaaatcaaaaatgcttttccagtgatccgtaaaaatgtgta
aatgtgcaagcgtaatatgcttaacaacgccagctaatctcctgtaaatagtaaaaa
gagtaataaagcgtgtaaacatcgcttctgtcgtcccgcagagcggaaaaatcgcg
cgacccgcccgggctatgaagaccatgggattactgacctgctgattgtcgacagcca
ggcgcacctagagcccctaccgctgggtcgcacatggatctggagatattattgataa
tattgaggttttccacacccttgccgacgcgctccacgatgtggatttaccgtagcgac
gacagcccgcagccgggcaaaaatttcattactacgcttcgcccgctgaactggttccctt
attacaggaaaaatcacgctggatgctcatgccgcgctggtttttgccgtgaggattc
cggctgaccaacgacgagctggcgctggcgatgtattgaccggcgctgccgatggcggc
ggattacccttcgctcaatctgggtcaggcggtcatgggtattgctatcaattagcagg
ttaaagcaacagaccccgggaatccgcttgatattgctgatgaatcgagttacaggcggt
acgcgcgcccttttacgctgtaaccactctggaggcggccgatgaccacaaattaac
cgactggctacaacagcgaatcgccctgctgggacagcgagatacggcaatggtgcacg
tttggctcatgatattgaaaaaaaaactaacaataaacgtgtgtgaattttaaaataat
a
```

In this case you've given `tail` an argument in two parts. In this case the `-n` says that you want to specify the number of lines to show and the `25` bit tells it how many. Unlike earlier when we merged arguments like `ls -lha` together, it's not a good idea to merge multiple two part arguments together because otherwise it is ambiguous which value goes with which argument.

`-n` is such a common argument for `tail` and `head` that it even has a shorthand: `-n 25` and `-25` mean the same thing.

## 2.3 Saving time

Saving time while typing may not seem important, but the longer that you spend in front of a computer, the happier you will be if you can reduce the time you spend at the keyboard.

- Pressing the up/down arrows will let you scroll through previous commands entered.
- If you highlight some text, middle clicking on the mouse will paste it on the command line.
- Tab completion doesn't just work on filenames, it also works on commands. Try it by typing `fin` and pressing `tab`...

`fin`

Although tab completion works on commands and filenames, unfortunately it rarely works on options or other arguments.

## 2.4 Getting help man

To obtain further information on any of the Unix commands introduced in this course you can use the `man` command. For example, to get a full description and examples of how to use the `sort` command use the following command in a terminal window.

```
man tail
```

There are several other useful commands that can be used to manipulate and summarise information inside files and we will introduce some of these next, `cat`, `sort`, `wc` and `uniq`.

## 2.5 Writing to files

So far we've been running commands and outputting the results into the terminal. That's obviously useful but what if you want to save the results to another file?

Run this:



```
head -1 Styphi.gff > first_Styphi_line.txt
```

(no output)

It's likely that nothing will have happened. This is because the `>` character has *redirected* the output of the `head` command. Instead of writing to the *standard output* (your terminal) it sent the output into the file `first_Styphi_line.txt`. Note that tab completion works for `Styphi.gff` because it exists but doesn't work for `first_Styphi_line.txt` because it doesn't exist yet.

## 2.6 cat

cat is another way of reading files, but unlike less it just throws the entire contents of the file onto your standard output. Try it on `first_Styphi_line.txt`



```
cat first_Styphi_line.txt
```

```
##gff-version 3
```

We don't need `first_Styphi_line.txt` any more so let's delete it



```
rm first_Styphi_line.txt
```

```
(no output)
```

The cat command can also be given the names of multiple files, one after the other and it will just output both. You can use this and the > symbol to join files together.

Having looked at the beginning and end of the `Styphi.gff` file you should notice that in the GFF file the annotation comes first, then the DNA sequence at the end. If you had two separate files containing the annotation and the DNA sequence, it is possible to concatenate or join the two together to make a single file like the `Styphi.gff` file you have just looked at. The command cat can be used to join two or more files into a single file. The order in which the files are joined is determined by the order in which they appear in the command line.

For example, we have two separate files, `Styphi.noseq.gff` and `Styphi.fa`, that contain the annotation and DNA sequence, respectively for the Salmonella typhi CT18 genome. To join together these files use:



```
cat Styphi.noseq.gff Styphi.fa > Styphi.concatenated.gff
```

```
(no output)
```

The files `Styphi.noseq.gff` and `Styphi.fa` will be joined together and written to a file called `Styphi.concatenated.gff`.

The > symbol in the command line directs the output of the cat program to the designated file `Styphi.concatenated.gff`. Use the command `ls` to check for the presence of this file.



```
ls
```

```
files.ipynb  Styphi.concatenated.gff Styphi.gff  
Pfalciparum.bed Styphi.fa          Styphi.noseq.gff
```

## 2.7 wc - counting

The command `wc` counts lines, words or characters.

There are two ways you could use it:



```
wc -l Styphi.gff
```

```
88961 Styphi.gff
```

or



```
cat Styphi.gff | wc -l
```

```
88961
```

Both give a similar answer. In the first example you tell `wc` the file that you want it to review (`Styphi.gff`) and pass the `-l` option to say that you're only interested in the number of lines.

In the second example you use the `|` symbol which is also known as the *pipe* symbol. This *pipes* the output of `cat Styphi.gff` into the input of `wc -l`. This means that you can also use the same `wc` tool to count other things. For example to count the number of files that are listed by `ls` use:



```
ls | wc -l
```

```
6
```

You can connect as many commands as you want. For example:



```
ls | grep ".gff" | wc -l
```

```
3
```

What does this command do? You will learn more about the `grep` command later in this course.

## 2.8 sort - sorting values

The `sort` lets you sort the contents of the input. When you sort the input, lines with identical content end up next to each other in the output. This is useful as the output can then be fed to the `uniq` command (see below) to count the number of unique lines in the input.

To sort the contents of a BED file use:

```
sort Pfalcciparum.bed
```



```
sort Pfalcciparum.bed | head
```

```
01    104936 105441 PFA0115w    1
01    107429 108580 PFA0120c   -1
01    110984 116033 EBA181  -1
01    11513  12397  RNAzID:13    1
01    119275 121483 FIKK1  -1
01    124752 125719 PFA0135w    1
01    126553 128375 PFA0140c   -1
01    129194 131074 PFA0145c   -1
01    132320 133858 PFA0150c   -1
01    134587 139491 PFA0155c   -1
```



```
sort Pfalciparum.bed | tail
```

```
14 979397 979586 RNAzID:2132 1
14 981211 982551 PF14TR004 1
14 981536 981592 RNAzID:2134 -1
14 982830 982889 RNAzID:2136 -1
14 983283 984503 PF14_0232 -1
14 985307 987697 PF14_0233 -1
14 987657 987729 RNAzID:2137 1
14 989162 992872 PF14_0234 1
14 993594 994242 PF14_0235 1
14 995103 1000448 PF14_0236 -1
```

To sort the contents of a BED file on position, type the following command.

```
sort -k 2 -n Pfalciparum.bed
```

The `sort` command can sort by multiple columns e.g. 1st column and then 2nd column by specifying successive `-k` parameters in the command.



```
sort -k 2 -n Pfalciparum.bed | head
```

```
06 653 1432 PFF0005c -1
14 1394 5344 PF14_0001 1
14 2215 5392 PF14TR001 1
06 3503 12835 VAR 1
09 6841 7670 RNAzID:4487 1
14 7113 7207 RNAzID:1975 1
14 7209 8539 RIF -1
11 8419 9249 RNAzID:585 1
03 8435 8527 RNAzID:2735 -1
14 8936 9033 RNAzID:1976 1
```



```
sort -k 2 -n Pfalciparum.bed | tail
```

```
14 3272513 3273783 RIF 1
14 3274613 3274669 RNAzID:2440 -1
14 3276165 3277436 RIF 1
14 3279435 3280597 RIF 1
14 3282002 3282056 RNAzID:2456 1
14 3282664 3283687 PF14_0771 1
14 3285383 3285466 RNAzID:2463 -1
14 3285835 3286938 RIF 1
14 3289946 3290002 RNAzID:2468 1
14 3290888 3291436 PF14_0773 1
```

Why not have a look at the manual for `sort` to see what these options do? Remember that you can type `/` followed by a search phrase, `n` to find the next search hit, `N` to find the previous search hit and `q` to exit.

```
man sort
```



## 2.9 uniq - finding unique values

The `uniq` command extracts unique lines from the input. It is usually used in combination with `sort` to count unique values in the input.

To get the list of chromosomes in the `Pfalcciparum` bed file use:



```
awk '{ print $1 }' Pfalcciparum.bed | sort | uniq
```

```
01
02
03
04
05
06
07
08
09
10
11
12
13
14
```

How many chromosomes are there? You will learn more about the `awk` command later in this course.

Warning: `uniq` is really stupid; it can only spot that two lines are the same if they are right next to one another. You therefore almost always want to `sort` your input data before using `uniq`.

Do you understand how this command is working? Why not try building it up piece by piece to see what it does?

```
awk '{ print $1 }' Pfalcciparum.bed | less
awk '{ print $1 }' Pfalcciparum.bed | sort | less
awk '{ print $1 }' Pfalcciparum.bed | sort | uniq | less
```

## 2.10 Exercises

Open up a new terminal window, navigate to the `files` directory in the `Unix_course` directory and complete the following exercise:

1. Use the `head` command to extract the first 500 lines of the file `Styphi.gff` and store the output in a new file called `Styphi.500.gff`.
2. Use the `wc` command to count the number of lines in the `Pfalcciparum.bed` file.
3. Use the `sort` command to sort the file `Pfalcciparum.bed` on chromosome and then gene position.
4. Use the `uniq` command to count the number of features per chromosome in the `Pfalcciparum.bed` file. Hint: use the `man` command to look at the options for the `uniq` command. Or peruse the `wc` or `grep` manuals. There's more than one way to do it!

## 3 Searching inside files with grep

A common task is to extract information from large files. This can be achieved using the Unix command `grep`, which stands for "Globally search for a Regular Expression and Print". The meaning of this acronym will become clear later, when we discuss Regular Expressions. First, we will consider simpler examples.

Before we start, change into the `Unix/grep` directory and double check that the following commands gives you a similar output:



```
pwd
ls
```

```
/home/manager/Module_3_Linux_Scripting/grep
answers.md          grep.ipynb  regex_example.txt
exercises.fasta    list_example.1 sequences.fasta
gene_expression.bed list_example.2
gene_expression_sneaky.bed list_example.3
```

### 3.1 Simple pattern matching

We will use a small example file (in "BED" format), which contains the expression levels of some genes. This is a column-based file, with a tab character between each column. There can be more than 10 columns, but only the first three are required to be a valid file. The file format is described in full here: <http://genome.ucsc.edu/FAQ/FAQformat#format1>. We will use the first 5 columns:

1. Sequence name
2. start position (starting from 0, not 1)
3. end position (starting from 0, not 1)
4. feature name
5. score (which is used to store the gene expression level in our examples).

Here is the contents of the first example BED file used in this course:



```
cat gene_expression.bed
```

```
chr1 10 100 gene1 10 +
chr1 350 500 gene2 1000 -
chr2 20 35 gene3 0 +
chr2 110 200 Gene4 4 -
chr3 1000 2000 gene5 100 +
chr10 1 100 gene6 11 -
chrX 60 90 Gene7 2 +
chrY 80 120 GENE8 42 -
```

In reality, such a file could contain 100,000s of lines, so that it is not practical to read manually. Suppose we are interested in all the genes from chromosome 2. We can find all these lines using `grep`:



```
grep chr2 gene_expression.bed
```

```
chr2 20 35 gene3 0 +
chr2 110 200 Gene4 4 -
```

This has shown us all the lines that contain the string "chr2".

We can use a pipe to then just extract the genes that are on the positive strand, using `grep` a second time:



```
grep chr2 gene_expression.bed | grep +
```

```
chr2 20 35 gene3 0 +
```

However, since `grep` is reporting a match to a string *anywhere* on a line, such simple searches can have undesired consequences. For example, consider the result of doing a similar search for all the genes in chromosome 1:



```
grep chr1 gene_expression.bed
```

```
chr1 10 100 gene1 10 +
chr1 350 500 gene2 1000 -
chr10 1 100 gene6 11 -
```

Oops! We found genes in chromosome 10, because "chr1" is a substring of "chr10".

Or consider the following file, where the genes have unpredictable names (which is not unusual for bioinformatics data).



```
cat gene_expression_sneaky.bed
```

```
chr1 10 100 gene1 10 +
chr1 350 500 gene2 1000 -
chr1 350 500 sneaky-gene3 1000 +
chr2 20 35 gene4 0 +
chr2 110 200 gene5 4 -
chr3 1000 2000 gene6 100 +
chr8 20 100 chr11.gene1 1000 -
chr10 1 100 gene7 11 -
chr11 20 100 sneaky-gene8 1000 +
```

Now we try to find genes on chromosome 1 that are on the negative strand. We put the minus sign in quotes, to stop Unix interpreting this as an option to `grep`, as opposed to the string we are searching for:



```
grep chr1 gene_expression_sneaky.bed | grep '-'
```

```
chr1 350 500 gene2 1000 -
chr1 350 500 sneaky-gene3 1000 +
chr8 20 100 chr11.gene1 1000 -
chr10 1 100 gene7 11 -
chr11 20 100 sneaky-gene8 1000 +
```

The extra lines are found by `grep` because of matches in columns we were not expecting to match. Remember, `grep` is reporting these lines because they each contain the strings "chr1" and "-" *some-where*.

We need a way to make searching with `grep` more specific.

## 3.2 Regular expressions

Regular expressions provide the solution to the above problems. They are a way of defining more specific patterns to search for.

### 3.2.1 Matching the start and end of lines

First, we can specify that a match must be at the start of a line using the symbol "^", which means "start of line". Without the ^, we find any match to "chr1":



```
grep chr1 gene_expression_sneaky.bed
```

```
chr1 10 100 gene1 10 +
chr1 350 500 gene2 1000 -
chr1 350 500 sneaky-gene3 1000 +
chr8 20 100 chr11.gene1 1000 -
chr10 1 100 gene7 11 -
chr11 20 100 sneaky-gene8 1000 +
```

However, notice the effect of searching for `^chr1` instead. Note that we put the regular expression in quotes, to avoid Unix errors. Not using quotes may or may not give an error, but it is safest to use quotes for anything but the simplest of searches.



```
grep '^chr1' gene_expression_sneaky.bed
```

```
chr1 10 100 gene1 10 +
chr1 350 500 gene2 1000 -
chr1 350 500 sneaky-gene3 1000 +
chr10 1 100 gene7 11 -
chr11 20 100 sneaky-gene8 1000 +
```

Good! We have removed the match to the badly-named gene "chr11.gene1", which is on chromosome 8. Now we want to avoid matching chromosomes 10 and 11. This can be done by also looking for a "tab" character, which is represented by writing `\t`. For technical reasons, which are beyond the scope of this course, we must also put a dollar sign before the quotes to make any search involving a tab character work.



```
grep '$^chr1\t' gene_expression_sneaky.bed
```

```
chr1 10 100 gene1 10 +
chr1 350 500 gene2 1000 -
chr1 350 500 sneaky-gene3 1000 +
```

To find the genes on the negative strand, all that remains is to match a minus sign at the *end* of the line (so that we do not find "sneaky-gene3"). We can do this using the dollar "\$", which means "end of line".



```
grep '$^chr1\t' gene_expression_sneaky.bed | grep '\-$'
```

```
chr1 350 500 gene2 1000 -
```

### 3.2.2 Wildcards and alphabets

Another special character in regular expressions is the dot: ".". This stands for any character. For example, this finds all matches to chromosomes 1-9, and chromosomes X and Y:



```
grep '$^chr.\t' gene_expression.bed
```

```
chr1 10 100 gene1 10 +
chr1 350 500 gene2 1000 -
chr2 20 35 gene3 0 +
chr2 110 200 Gene4 4 -
chr3 1000 2000 gene5 100 +
chrX 60 90 Gene7 2 +
chrY 80 120 GENE8 42 -
```

In fact, the earlier command that found all genes on chromosome 1 that are on the negative strand, could be found with a single call to `grep` instead of two calls piped together. To do this, we need a regular expression that finds lines that:

- start with chr1, then a tab character
- end with a minus
- have arbitrary characters between.

The asterisk "\*" has a special meaning: it says to match any number (including zero) of whatever character is before the \*. For example, the regular expression 'AC\*G' will match AG, ACG, ACCG, etc. The simpler, improved command is:



```
grep '$^chr1\t.*-$' gene_expression_sneaky.bed
```

```
chr1 350 500 gene2 1000 -
```

As well as matching any character using a dot, we can define any list of characters to match, using square brackets. For example, [12X] means match a 1, 2, or an X. This can be used to find all genes from chromosomes 1, 2 and X:



```
grep '$^chr[12X]\t' gene_expression.bed
```

```
chr1 10 100 gene1 10 +
chr1 350 500 gene2 1000 -
chr2 20 35 gene3 0 +
chr2 110 200 Gene4 4 -
chrX 60 90 Gene7 2 +
```

Or just the autosomes may be of interest. To do this we introduce two new features:

- Ranges can be given in square brackets, for example `[1-5]` will match 1, 2, 3, 4 or 5.
- The plus sign `"+"` has a special meaning that is similar to `"*"`. Instead of any number of matches (including zero), it looks for at least one match. To avoid simply matching a plus sign, it must be preceded by a backslash: `"\+"`. For example, the regular expression `'AC\+G'` will match `ACG`, `ACCG`, `ACCCG` etc (but will not match `AG`).

Warning: Adding a backslash is often called *escaping* (e.g. *escape the plus symbol*). Depending on the software you're using (and the options you give it), you may need to escape the symbol to indicate that you want its special regex meaning (e.g. multiple copies of the last character please) or it's literal meaning (e.g. give me a '+' symbol please). If your command isn't working as you expect, try playing with these options and always test your regular expression before assuming it gave you the right answer.

The command to find the autosomes is:



```
grep $'^chr[0-9]\+\t' gene_expression.bed
```

```
chr1 10 100 gene1 10 +
chr1 350 500 gene2 1000 -
chr2 20 35 gene3 0 +
chr2 110 200 Gene4 4 -
chr3 1000 2000 gene5 100 +
chr10 1 100 gene6 11 -
```

### 3.3 Other grep options

The Unix command `grep` and regular expressions are extremely powerful and we have only scratched the surface of what they can do. Take a look at the manual (by typing `man grep`) to get an idea. A few particularly useful options are discussed below.

#### 3.3.1 Counting matches

A common use-case is counting matches within files. Instead of output each matching line, the option `"-c"` tells `grep` to report the number of lines that matched. For example, the number of genes in the autosomes in the above example can be found by simply adding `-c` to the command.



```
grep -c $'^chr[0-9]\+\t' gene_expression.bed
```

```
6
```

#### 3.3.2 Case sensitivity

By default, `grep` is case-sensitive. It can be useful to ignore the distinction between upper and lower case using the option `"-i"`. Suppose we have a file of sequences, and want to find the sequences that contain the string `ACGT`. It is not unusual to come across files that have a mix of upper and lower case nucleotides. Consider this FASTA file:





```
cat sequences.fasta
```

```
>sequence1  
aACGTaaacaca  
>sequence2  
TacgtAAAAA  
>sequence3  
AAAAAAA  
>sequence4  
agcACgtAA
```

A simple search for ACGT will not return all the results:



```
grep ACGT sequences.fasta
```

```
aACGTaaacaca
```

However, making the search case-insensitive solves the problem.



```
grep -i ACGT sequences.fasta
```

```
aACGTaaacaca  
TacgtAAAAA  
agcACgtAA
```

### 3.3.3 Searching in more than one file

So far, we have restricted to searches in one file, but `grep` can be given a list of files in which to search. As an example, we are given three files called `list_example.1`, `list_example.2`, and `list_example.3`. They are simple lists of genes, for illustrative purposes. For example, the first file looks like this:



```
cat list_example.1
```

```
gene1  
gene2  
gene3  
gene4  
gene5
```

Which files contain "gene1"?



```
grep '^gene1$' list_example.1 list_example.2
```

```
list_example.1:gene1
```

`gene1` only appears in the file `list_example.1`. The output format of `grep` has now changed, because it was given a list of files. The format is:

- filename:line\_that\_matches

ie, the name of the file has been added to the start of each matching line.

For convenience, there's also a way of specifying all of the list examples:



```
echo list_example.*
```

```
list_example.1 list_example.2 list_example.3
```



```
grep '^gene1$' list_example.*
```

```
list_example.1:gene1
```

How about gene42?



```
grep '^gene42$' list_example.*
```

```
list_example.2:gene42
list_example.3:gene42
list_example.3:gene42
```

gene42 appears once in list\_example.2 and twice in list\_example.3.

### 3.3.4 Inverting matches

By default, grep reports all lines that do match the regular expression. Sometimes it is useful to filter a file, by reporting lines that *do not* match the regular expression. Using the option "-v" makes grep "invert" the output. For example, we could exclude genes from autosomes in the BED file from earlier.



```
grep -v $'^chr[0-9]\+\t' gene_expression.bed
```

```
chrX 60 90 Gene7 2 +
chrY 80 120 GENE8 42 -
```

## 3.4 Replacing matches to regular expressions

Finally, we show how to replace every match to a regular expression with something else, using the command "sed". The general form of this is:

```
sed 's/regular expression/new string/' input_file
```

This will output a new version of the input file, with each match to the regular expression replaced with "new string". For example:



```
sed 's/^chr/chromosome/' gene_expression.bed
```

```
chromosome1 10 100 gene1 10 +
chromosome1 350 500 gene2 1000 -
chromosome2 20 35 gene3 0 +
chromosome2 110 200 Gene4 4 -
chromosome3 1000 2000 gene5 100 +
chromosome10 1 100 gene6 11 -
chromosomeX 60 90 Gene7 2 +
chromosomeY 80 120 GENE8 42 -
```

### 3.5 Exercises

The following exercises all use the FASTA file `exercises.fasta`. Before starting the exercises, open a new terminal and navigate to the `grep/` directory, which contains `exercises.fasta`.

Use `grep` to find the answers. Hint: some questions require you to use `grep` twice, and possibly some other Unix commands.

1. Make a `grep` command that outputs just the lines with the sequence names.
2. How many sequences are in the file?
3. Do any sequence names have spaces in them? What are their names?
4. Make a `grep` command that outputs just the lines with the sequences, not the names.
5. How many sequences contain unknown bases (an "n" or "N")?
6. Are there any sequences that contain non-nucleotides (something other than A, C, G, T or N)?
7. How many sequences contain the 5' cut site GCWGC (where W can be an A or T) for the restriction enzyme `AceI`?
8. Are there any sequences that have the same name? You do not need to find the actual repeated names, just whether any names are repeated. (Hint: it may be easier to first discover how many unique names there are).

## 4 File processing with AWK

AWK is a programming language named after the initials of its three inventors: Alfred Aho, Peter Weinberger, and Brian Kernighan. AWK is incredibly powerful at processing files, particularly column-based files, which are commonplace in Bioinformatics. For example, BED, GFF, and SAM files.

Although long programs, put into a separate file, can be written using AWK, we will use it directly on the command line. Effectively, these are very short AWK programs, often called "one-liners".

Before we start, change into the `Unix/awk` directory and double check that the following commands gives you a similar output:



```
pwd
ls
```

```
/home/manager/Module_3_Linux_Scripting/awk
answers.md awk.ipynb exercises.bed genes.gff
```

### 4.1 Extracting columns from files

`awk` reads a file line-by-line, splitting each line into columns. This makes it easy to do simple things like extract a column from a file. We will use the following GFF file for our examples.



```
cat genes.gff
```

```
chr1  source1  gene  100   300   0.5   +     0     name=gene1;product=unknown
chr1  source2  gene  1000  1100  0.9   -     0     name=recA;product=RecA protein
chr1  source5  repeat 10000 14000 1     +     .     name=ALU
chr2  source2  gene  10000 1200  0.95  +     0     name=gene2;product=gene2 protein
chr3  source1  gene  200   210   0.8   .     0     name=gene3
chr4  source3  repeat 300   400   1     +     .     name=ALU
chr10 source2  repeat 60    70    0.78  +     .     name=LINE1
chr10 source2  repeat 150   166   0.84  +     .     name=LINE2
chrX  source1  gene  123   456   0.6   +     0     name=gene4;product=unknown
```

The columns in the GFF file are separated by tabs and have the following meanings:

1. Sequence name
2. Source - the name of the program that made the feature
3. Feature - the type of feature, for example gene or CDS
4. Start position
5. Stop position
6. Score
7. Strand (+ or -)
8. Frame (0, 1, or 2)
9. Optional extra information, in the form `key1=value1;key2=value2;...`

The score, strand, and frame can be set to '.' if it is not relevant for that feature. The final column 9 may or may not be present and could contain any number of key, value pairs.

We can use `awk` to just print the first column of the file. `awk` calls the columns `$1`, `$2`, ... etc, and the complete line is called `$0`.



```
awk -F"\t" '{print $1}' genes.gff
```

```
chr1
chr1
chr1
chr2
chr2
chr3
chr4
chr10
chr10
chrX
```

A little explanation is needed.

- The option `-F"\t"` was needed to tell `awk` that the columns are separated by tabs (more on this later).
- For each line of the file, `awk` does what is inside the curly brackets. In this case, we simply print the first column.

The repeated chromosome names are not nice. It is more likely to want to know just the unique names, which can be found by piping into the Unix command `sort`.



```
awk -F"\t" '{print $1}' genes.gff | sort -u
```

```
chr1
chr10
chr2
chr3
chr4
chrX
```

## 4.2 Filtering the input file

Similarly to `grep`, `awk` can be used to filter out lines of a file. However, since `awk` is column-based, it makes it easy to filter based on properties of any columns of interest. The filtering criteria can be added before the braces. For example, the following extracts just chromosome 1 from the file.



```
awk -F"\t" '$1=="chr1" {print $0}' genes.gff
```

```
chr1 source1 gene 100 300 0.5 + 0 name=gene1;product=unknown
chr1 source2 gene 1000 1100 0.9 - 0 name=recA;product=RecA protein
chr1 source5 repeat 10000 14000 1 + . name=ALU
```

There are two important things to note from the above command:

1. `$1=="chr1"` means that column 1 must be *exactly* equal to "chr1". This means that "chr10" is not found.
2. The `{print $0}` part only happens when the first column is equal to "chr1", otherwise `awk` does nothing (the line gets ignored).

Awk commands are made up of two parts, a *pattern* (e.g. `$1=="chr1"`) and an *action* (e.g. `print $0`) which is contained in curly braces. The *pattern* defines which lines the *action* is applied to.

In fact, the action (the part in curly braces) can be omitted in this example. `awk` assumes that you want to print the whole line, unless it is told otherwise. This gives a simple method of filtering based on columns.



```
awk -F"\t" '$1=="chr1"' genes.gff
```

```
chr1 source1 gene 100 300 0.5 + 0 name=gene1;product=unknown
chr1 source2 gene 1000 1100 0.9 - 0 name=recA;product=RecA protein
chr1 source5 repeat 10000 14000 1 + . name=ALU
```

You might remember using another of `awk`'s defaults in a previous exercise. In that example we supplied an action but no pattern. In this case, `awk` assumes that you want to apply the action to every line in the file. For example:



```
awk -F"\t" '{print $1}' genes.gff
```

```
chr1
chr1
chr1
chr2
chr2
chr3
chr4
chr10
chr10
chrX
```

Multiple patterns can be combined using `&&` to mean "and". For example, to find just the genes from chromosome 1:



```
awk -F"\t" '$1=="chr1" && $3=="gene"' genes.gff
```

```
chr1 source1 gene 100 300 0.5 + 0 name=gene1;product=unknown
chr1 source2 gene 1000 1100 0.9 - 0 name=recA;product=RecA protein
```

The entire line need not be printed (remember, if not specified, `awk` assumes a `print $0`). Suppose we want only the sources of the genes on chromosome 1:



```
awk -F"\t" '$1=="chr1" && $3=="gene" {print $2}' genes.gff | sort -u
```



```
source1
source2
```

Similarly to using "&&" for "and", there is "||" to mean "or". To find features that are repeats or made by the tool "source2":



```
awk -F"\t" '$2=="source2" || $3=="repeat" genes.gff
```

```
chr1 source2 gene 1000 1100 0.9 - 0 name=recA;product=RecA protein
chr1 source5 repeat 10000 14000 1 + . name=ALU
chr2 source2 gene 10000 1200 0.95 + 0
chr4 source3 repeat 300 400 1 + . name=ALU
chr10 source2 repeat 60 70 0.78 + . name=LINE1
chr10 source2 repeat 150 166 0.84 + . name=LINE2
```

So far, we have only used strings for the filtering. Numbers can also be used. We could ask awk to return all the genes on chromosome 1 that start before position 1100:



```
awk -F"\t" '$1=="chr1" && $3=="gene" && $4 < 1100' genes.gff
```

```
chr1 source1 gene 100 300 0.5 + 0 name=gene1;product=unknown
chr1 source2 gene 1000 1100 0.9 - 0 name=recA;product=RecA protein
```

Instead of looking for exact matches to strings, regular expressions can be used. The symbol "~" is used instead of "==". For example, to find all the autosomes, we need to use a regular expression for matches to the first column. The regular expression is written between forward slashes.



```
awk -F"\t" '$1 ~ /^chr[0-9]+$/ genes.gff
```

```
chr1 source1 gene 100 300 0.5 + 0 name=gene1;product=unknown
chr1 source2 gene 1000 1100 0.9 - 0 name=recA;product=RecA protein
chr1 source5 repeat 10000 14000 1 + . name=ALU
chr2 source2 gene 10000 1200 0.95 + 0
chr2 source1 gene 50 900 0.4 - 0 name=gene2;product=gene2 protein
chr3 source1 gene 200 210 0.8 . 0 name=gene3
chr4 source3 repeat 300 400 1 + . name=ALU
chr10 source2 repeat 60 70 0.78 + . name=LINE1
chr10 source2 repeat 150 166 0.84 + . name=LINE2
```

Like with grep, matches can be inverted. grep has the option -v, but with awk we use "!~" to mean "does not match". This inverts the previous example:



```
awk -F"\t" '$1 !~ /^chr[0-9]+$/ genes.gff
```

```
chrX source1 gene 123 456 0.6 + 0 name=gene4;product=unknown
```

If we do not specify a column, awk looks for a match anywhere in the whole line (it assumes we wrote \$0 ~ /regex/). So, in some sense, awk can be used as a replacement for grep:



```
awk '/repeat/' genes.gff
```

```
chr1  source5 repeat 10000 14000 1 + . name=ALU
chr4  source3 repeat 300 400 1 + . name=ALU
chr10 source2 repeat 60 70 0.78 + . name=LINE1
chr10 source2 repeat 150 166 0.84 + . name=LINE2
```

(the `-F"\t"` was omitted because the match is to the whole line, so how the columns are separated is not relevant.)



```
grep repeat genes.gff
```

```
chr1  source5 repeat 10000 14000 1 + . name=ALU
chr4  source3 repeat 300 400 1 + . name=ALU
chr10 source2 repeat 60 70 0.78 + . name=LINE1
chr10 source2 repeat 150 166 0.84 + . name=LINE2
```

However, with `awk` we can easily pull out information from the matching lines. Suppose we want to know which chromosomes have repeats. It is easy with `awk`.



```
awk -F"\t" '/repeat/ {print $1}' genes.gff | sort -u
```

```
chr1
chr10
chr4
```

### 4.3 Sanity checking files

Never, ever trust the contents of Bioinformatics files (even if you made them!). We now have enough skills to do some basic sanity checking of a GFF file. For example, to check that every gene has been assigned a strand:



```
awk -F"\t" '$3=="gene" && !($7 == "+" || $7 == "-)' genes.gff
```

```
chr3  source1 gene 200 210 0.8 . 0 name=gene3
```

Something went wrong when this file was made: `gene3` has an unknown strand.

Do the start and end coordinates of all the features make sense?



```
awk -F"\t" '$5 < $4' genes.gff
```

```
chr2  source2 gene 10000 1200 0.95 + 0
```

According to the file, this gene starts at position 10000 and ends at position 1200, which does not make sense. Also, it has no name (the final optional column is empty). We could check if there are any other genes with no name. One way to do this is to use the special variable `"NF"`, which is the number of columns (fields) in the current line. Since the final column is optional, each line might have 8 or 9 columns. We need to write a command that will check:

- If the feature is a gene, and if it is:
- check if the number of columns is less than 9. When there are 9 columns, check if there is a name defined.



```
awk -F"\t" '$3=="gene" && (NF<9 || $NF !~/name/)' genes.gff
```

```
chr2  source2  gene  10000  1200  0.95  +  0
```

Note the distinction between `NF` (the number of columns) and `$NF` (the contents of the final column).

As promised earlier, we now consider the relevance of the option `-F"\t"`, to tell `awk` that the columns in the input file are separated with tab characters. If we forgot to use this option, then `awk` will use its default behaviour, which is to separate on *any* whitespace (which usually means tabs and/or spaces). However, consider the final column of the file - it can contain whitespace, which means that messy things happen. Suppose we try to extract the optional extra final column of the file, when it is present. Compare the effect of running `awk` with and without `-F"\t"`.



```
awk -F"\t" 'NF>8 {print $NF}' genes.gff
```

```
name=gene1;product=unknown
name=recA;product=RecA protein
name=ALU
name=gene2;product=gene2 protein
name=gene3
name=ALU
name=LINE1
name=LINE2
name=gene4;product=unknown
```



```
awk 'NF>8 {print $NF}' genes.gff
```

```
name=gene1;product=unknown
protein
name=ALU
protein
name=gene3
name=ALU
name=LINE1
name=LINE2
name=gene4;product=unknown
```

One more sanity check: each line should have 8 or 9 columns (remembering to use `-F"\t"!`)



```
awk -F"\t" 'NF<8 || NF>9' genes.gff
```

```
(no output)
```

There was no output, which means that every line does indeed have 8 or 9 columns.

## 4.4 Changing the output

In addition to filtering, `awk` can be used to change the output.

Every value in a column could be changed to something else, for example suppose we want to change the source column (column number 2) to something else.



```
awk -F"\t" '{ $2="new_source"; print $0}' genes.gff
```

```
chr1 new_source gene 100 300 0.5 + 0 name=gene1;product=unknown
chr1 new_source gene 1000 1100 0.9 - 0 name=recA;product=RecA protein
chr1 new_source repeat 10000 14000 1 + . name=ALU
chr2 new_source gene 10000 1200 0.95 + 0
chr2 new_source gene 50 900 0.4 - 0 name=gene2;product=gene2 protein
chr3 new_source gene 200 210 0.8 . 0 name=gene3
chr4 new_source repeat 300 400 1 + . name=ALU
chr10 new_source repeat 60 70 0.78 + . name=LINE1
chr10 new_source repeat 150 166 0.84 + . name=LINE2
chrX new_source gene 123 456 0.6 + 0 name=gene4;product=unknown
```

This is close, but look carefully at the output. What happened? The output is not tab-separated, but is instead separated with spaces. To restore the tabs, we need to use another special variable called "OFS" (Output Field Separator), and change it before `awk` does any processing of the input file. This can be achieved by adding "BEGIN{OFS="\t"}", as in the next example. Before `awk` reads any lines of the file it runs the `BEGIN` block of code, which in this case changes `OFS` to be a tab character.



```
awk -F"\t" 'BEGIN{OFS="\t"} {$2="new_source"; print $0}' genes.gff
```

```
chr1 new_source gene 100 300 0.5 + 0 name=gene1;product=unknown
chr1 new_source gene 1000 1100 0.9 - 0 name=recA;product=RecA pro
chr1 new_source repeat 10000 14000 1 + . name=ALU
chr2 new_source gene 10000 1200 0.95 + 0
chr2 new_source gene 50 900 0.4 - 0 name=gene2;product=gene2 p
chr3 new_source gene 200 210 0.8 . 0 name=gene3
chr4 new_source repeat 300 400 1 + . name=ALU
chr10 new_source repeat 60 70 0.78 + . name=LINE1
chr10 new_source repeat 150 166 0.84 + . name=LINE2
chrX new_source gene 123 456 0.6 + 0 name=gene4;product=unknow
```

## 4.5 Processing the data

More in-depth processing is possible. For example, we could print the length of each repeat (and then sort the results numerically)



```
awk -F"\t" '$3=="repeat" {print $5 - $4 + 1}' genes.gff | sort -n
```

```
11
17
101
4001
```

Perhaps we would like to know the total length of the repeats. We need to use a variable to add up the total lengths and print the final total. In the same way that `awk` has a `BEGIN` block, it can also be given an `END` block that is only run when `awk` has finished reading all lines of the input file.



```
awk -F"\t" 'BEGIN{sum=0} $3=="repeat" \
           {sum = sum + $5 - $4 + 1} \
           END{print sum}' genes.gff
```

4130

The total repeat length was stored in a variable called `sum`. The previous `awk` command can be broken down into three parts:

1. The `BEGIN{sum=0}` sets `sum` to zero before any lines of the file are read.
2. `awk` reads each line of the file. Each time a repeat is found, the length of that repeat is added to `sum`.
3. Once all lines of the file have been read, `awk` runs the `END` block: `END{print sum}`. This prints the value of `sum`.

In fact, the command can be shortened a little. Adding a number to a variable is so common, that there is a shorthand way to write it. Instead of

```
sum = sum + $5 - $4 + 1
```

we can use

```
sum += $5 - $4 + 1
```

to get the same result.



```
awk -F"\t" 'BEGIN{sum=0} \
           $3=="repeat" {sum += $5 - $4 + 1} \
           END{print sum}' genes.gff
```

4130

Maybe we would like to know the mean score of the genes. We need to calculate the total score, and divide this by the number of genes. To keep track of the number of genes, we use a variable called `count`. Each time a new gene is found, 1 must be added to `count`. This could be done by writing

```
count = count + 1
```

but instead we will use the shorthand

```
count++
```



```
awk -F"\t" 'BEGIN{sum=0; count=0} \
           $3=="gene" {sum += $6; count++} \
           END{print sum/count}' genes.gff
```

0.691667

Finally, `awk` has a default behaviour that means we do not even need the `BEGIN` block. It can be completely omitted in this example because we are setting `sum` and `count` to zero. The first time `awk` sees a variable being used, it will set it to zero by default. For example, when `awk` reads the first line of the file, the piece of code

```
count++
```

tells `awk` to add 1 to `count`. However, if `awk` has not encountered the variable `count` before, it assumes it is zero (as if we had written `BEGIN{count=0}`), then adds 1 to it. The result is that `count` is equal to 1. Similar comments apply to the variable `sum`.



```
awk -F"\t" '$3=="gene" {sum += $6; count++} \
END{print sum/count}' genes.gff
```

```
0.691667
```

If this confuses you, then be explicit and use the `BEGIN` block of code. The result is the same.

## 4.6 Exercises

The following exercises all use the BED file `exercises.bed`. Before starting the exercises, open a new terminal and navigate to the `awk/` directory, which contains `exercises.bed`.

Use `awk` to find the answers to the following questions about the file `exercises.bed`. Many questions will require using pipes (eg `"awk ... | sort -u"` for question 1).

1. What are the names of the contigs in the file?
2. How many contigs are there?
3. How many features are on the positive strand?
4. How many features are on the negative strand?
5. How many genes are there?
6. How many genes have no strand assigned to them (ie the final column is not there)?
7. Are any gene names repeated? (Hint: you do not need to find their names, just a yes or no answer. Consider the number of unique gene names.)
8. What is the total score of the repeats?
9. How many features are in contig-1?
10. How many repeats are in contig-1?
11. What is the mean score of the repeats in contig-1?

## 5 BASH scripts

So far, we have run single commands in a terminal. However, it is useful to be able to run multiple commands that process some data and produce output. These commands can be put into a separate file (ie a script), and then run on the input data. This has the advantage of reproducibility, so that the same analysis can be run on many input data sets.

### 5.1 First script

It is traditional when learning a new language (in this case BASH), to write a simple script that says "Hello World!". We will do this now.

First, open a terminal and make a new directory in your home called `scripts`, by typing

```
cd
mkdir ~/scripts
```

Next open a text editor, which you will use to write the script. What text editors are available will depend on your system. For example, `gedit` in Linux. Do not try to use a word processor, such as Word! If you don't already have a favorite, try `gedit` by running the following command:

```
gedit &
```

Type this into the text editor:

```
echo Hello World!
```

and save this to a file called `hello.sh` in your new `scripts` directory. This script will print `Hello World!` to the screen when we run it. First, in your terminal, check that the script is saved in the correct place.



```
cd scripts
ls hello.sh
```

```
hello.sh
```

If everything is OK, then next try to run the script. For now, we need to tell Unix that this is a bash script, and where it is:



```
bash hello.sh
```

```
Hello World!
```

### 5.2 Setting up a scripts directory

It would be nice if all our scripts could simply be run from anywhere in the filesystem, without having to tell Unix where the script is, or that it is a BASH script. This is how the built-in commands work, such as `cd` or `ls`.

To tell Unix that the script is a BASH script, make this the first line of the script:



```
#!/usr/bin/env bash
```

and remember to save the script again. This special line at the start of the file tells Unix that the file is a bash script, so that it expects bash commands throughout the file. There is one more change to be made to the file to tell Unix that it is a program to be run (it is "executable"). This is done with the command `chmod`. Type this into the terminal to make the file executable:



```
chmod +x hello.sh
```

(no output)

Now, the script can be run, but we must still tell Unix where the script is in the filesystem. In this case, it is in the current working directory, which is called `./`.



```
./hello.sh
```

Hello World!

The final thing to do is change our setup so that Unix can find the script without us having to explicitly say where it is. Whenever a command is typed into Unix, it has a list of directories that it searches through to look for the command. We need to add the new scripts directory to that list of directories. Try typing

```
echo $PATH
```

It returns a list of directories, which are all the places Unix will look for a command. Before we add the scripts directory to this list, check what happens if we try to run the script without telling Unix where it is:

```
hello.sh
bash: hello.sh: command not found
```

Unix did not find it! The command to run to add the scripts directory to `$PATH` is:

```
export PATH=$PATH:~/scripts/
```

If you want this change to be permanent, ie so that Unix finds your scripts after you restart or logout and login again, add that line to the end of a file called `~/.bashrc`. If you are using a Mac, then the file should instead be `~/.bash_profile`. If the file does not already exist, then create it and put that line into it.

The following command is only here so that this notebook finds scripts correctly and the remaining examples work. **Do not type the next command into your terminal.**



```
PATH=$PATH:$PWD # do not type this into your terminal!
```

(no output)

Now the script works, no matter where we are in the filesystem. Unix will check the scripts directory and find the file `hello.sh`. You can be *anywhere* in your filesystem, and simply running

```
hello.sh
```

will always work. Try it now.



```
hello.sh
```

```
Hello World!
```

In general, when making a new script, you can now copy and edit an existing script, or make a new one like this:

```
cd ~/scripts
touch my_script.sh
chmod +x my_script.sh
```

and then open `my_script.sh` in a text editor.

### 5.3 Getting options from the terminal and printing a help message

Usually, we would like a script to read in options from the user, such as the name of an input file. This would mean a script can be run like this:

```
my_script.sh input_file
```

Inside the script, the parameters provided by the user are given the names `$1`, `$2`, `$3` etc (do not confuse these with column names used by `awk`!). Here is a simple example that expects the user to provide a filename and a number. The script simply prints the filename to the screen, and then the first few lines of the file (the number of lines is determined by the number given by the user).



```
cat options_example.sh
```

```
#!/usr/bin/env bash

echo filename is: $1
echo

echo First $2 lines of file $1 are:
head -n $2 $1
```



```
options_example.sh test_file 2
```

```
filename is: test_file

First 2 lines of file test_file are:
test file line 1
test file line 2
```

The options have been used by the script, but the script itself is not very readable. It is better to use names instead of `$1` and `$2`. Here is an improved version of the script that does exactly the same as the previous script, but is more readable.



```
cat options_example.2.sh
```

```
#!/usr/bin/env bash
filename=$1
number_of_lines=$2

echo filename is: $filename
echo

echo First $number_of_lines lines of file $filename are:
head -n $number_of_lines $filename
```

## 5.4 Checking options from the user

The previous scripts will have strange behaviour if the input is not as expected by the script. Many things could go wrong. For example:

- The wrong number of options are given by the user
- The input file does not exist.

Try running the script with different options and see what happens.

A convention with scripts is that it should output a help message if it is not run correctly. This shows anyone how the script should be run (including you!) without having to look at the code inside the script.

A basic check for this script would be to verify that two options were supplied, and if not then print a help message. The code looks like this:

```
if [ $# -ne 2 ]
then
    echo "usage: options_example.3.sh filename number_of_lines"
    echo
    echo "Prints the filename, and the given first number of lines of the file"
    exit
fi
```

You can copy this code into the start of any of your scripts, and easily modify it to work for that script. A little explanation:

- A special variable `$#` has been used, which is the number of options that were given by the user.
- The whole block of code has the form `"if [ $# -ne 2 ] then ... fi"`. This only runs the code between the `then` and `fi`, if `$#` (the number of options) is not 2.
- The line `exit` simply makes the script end, so that no more code is run.



```
options_example.3.sh
```

```
usage: options_example.3.sh filename number_of_lines
```

```
Prints the filename, and the given first number of lines of the file
```

Another check is that the input file really does exist. If it does not exist, then there is no point in trying to run any more code. This can be checked with another `if ... then ... fi` block of code:

```
if [ ! -f $filename ]
then
    echo "File '$filename' not found! Cannot continue"
    exit
fi
```

Putting this all together, the script now looks like this:



```
cat options_example.3.sh

#!/usr/bin/env bash
set -eu

# check that the correct number of options was given.
# If not, then write a message explaining how to use the
# script, and then exit.
if [ $# -ne 2 ]
then
    echo "usage: options_example.3.sh filename number_of_lines"
    echo
    echo "Prints the filename, and the given first number of lines of the file"
    exit
fi

# Use sensibly named variables
filename=$1
number_of_lines=$2

# check if the input file exists
if [ ! -f $filename ]
then
    echo "File '$filename' not found! Cannot continue"
    exit
fi

# If we are still here, then the input file was found
echo filename is: $filename
echo

echo First $number_of_lines lines of file $filename are:
head -n $number_of_lines $filename
```

Two new features have also been introduced in this file:

1. The second line is "set -eu". Without this line, if any line produces an error, the script will carry on regardless to the end of the script. Using the -e option, an error anywhere in the file will result in the script stopping at the line that produced the error, instead of continuing. In general, it is best that the script stops at any error. The -u creates an error if you try to use a variable which doesn't exist. This helps to stop typos doing bad things to your analysis.
2. There are several lines starting with a hash #. These lines are "comment lines" that are not run. They are used to document the code, containing explanations of what is happening. It is good practice to comment your scripts!

The above script provides a template for writing your own scripts. The general method is:

1. Tell Unix that this is a BASH script, and to stop at the first error.
2. Check if the user ran the script correctly. If not, output a message telling the user how to run the script.
3. Check the input looks OK (in this case, that the input file exists).
4. Process the input.

## 5.5 Using variables to store output from commands

It can be useful to run a command and put the results into a variable. Recall that we stored the input from the user in sensibly named variables:

```
filename=$1
```

The part after the equals sign could actually be any command that returns some output. For example, running this in Unix

```
wc -l filename | awk '{print $1}'
```

returns the number of lines. In case you are wondering why the command includes `| awk '{print $1}'`, check what happens with and without the pipe to `awk`:



```
wc -l options_example.3.sh
```

```
31 options_example.3.sh
```



```
wc -l options_example.3.sh | awk '{print $1}'
```

```
31
```

With a small change, this can be stored in a variable and then used later.



```
filename=options_example.3.sh
line_count=$(wc -l $filename | awk '{print $1}')
echo There are $line_count lines in the file $filename
```

```
There are 31 lines in the file options_example.3.sh
```

## 5.6 Repeating analysis with loops

It is common in Bioinformatics to run the same analysis on many files. Suppose we had a script that ran one type of analysis, and wanted to repeat the same analysis on 100 different files. It would be tedious, and error-prone, to write the same command 100 times. Instead we can use a loop. As an example, we will just run the Unix command `wc` on each file but instead, in reality this would be a script that runs in-depth analysis. We can run `wc` on each of the files in the directory `loop_files/` with the following command.



```
for filename in loop_files/*; do wc $filename; done
```

```
2 8 28 loop_files/file.1
5 20 70 loop_files/file.2
6 24 84 loop_files/file.3
1 4 14 loop_files/file.4
0 0 0 loop_files/file.5
```

## 5.7 Exercises

1. Write a script that gets a filename from the user. If the file exists, it prints a nice human-readable message telling the user how many lines are in the file.
2. Use a loop to run the script from Exercise 1 on the files in the directory `loop_files/`.
3. Write a script that takes a GFF filename as input. Make the script produce a summary of various properties of the file. There is an example input file provided called `bash_scripts/exercise_3.gff`. Use your imagination! You could have a look back at the `awk` section of the course for inspiration. Here are some ideas you may wish to try:
  - Does the file exist?
  - How many records (ie lines) are in the file?
  - How many genes are in the file?
  - Is the file badly formatted in any way (eg wrong number of columns, do the coordinates look like numbers)?

## 6 UNIX Quick Reference Guide

### 6.1 Looking at files and moving them around

```
pwd # Tell me which directory I'm in
ls # What else is in this directory
ls .. # What is in the directory above me
ls foo/bar/ # What is inside the bar directory which is inside the foo/ directory
ls -lah foo/ # Give the the details (-l) of all files and folders (-a) using human
              # readable file sizes (-h)
cd ../.. # Move up two directories
cd ../foo/bar # Move up one directory and down into the foo/bar/ subdirectories
cp -r foo/ baz/ # Copy the foo/ directory into the baz/ directory
mv baz/foo .. # Move the foo directory into the parent directory
rm -r ../foo # remove the directory called foo/ from the parent directory
find foo/ -name "*.gff" # find all the files with a gff extension in the directory foo/
```

### 6.2 Looking in files

```
less bar.bed # scroll through bar.bed
grep chrom bar.bed | less -S # Only look at lines in bar.bed which have 'chrom' and
                              # don't wrap lines (-S)
head -20 bar.bed # show me the first 20 lines of bar.bed
tail -20 bar.bed # show me the last 20 lines
cat bar.bed # show me all of the lines (bad for big files)
wc -l bar.bed # how many lines are there
sort -k 2 -n bar.bed # sort by the second column in numerical order
awk '{print $1}' bar.bed | sort | uniq # show the unique entries in the first column
```

### 6.3 Grep

```
grep foo bar.bed # show me the lines in bar.bed with 'foo' in them
grep foo baz/* # show me all examples of foo in the files immediately within baz/
grep -r foo baz/ # show me all examples of foo in baz/ and every subdirectory within it
grep '^foo' bar.bed # show me all of the lines beginning with foo
grep 'foo$' bar.bed # show me all of the lines ending in foo
grep -i '^[acgt]$\n' bar.bed # show me all of the lines which only have the characters
                              # a,c,g and t (ignoring their case)
grep -v foo bar.bed # don't show me any files with foo in them
```

### 6.4 Awk

```
awk '{print $1}' bar.bed # just the first column
awk '$4 ~ /^foo/' bar.bed # just rows where the 4th column starts with foo
awk '$4 == "foo" {print $1}' bar.bed # the first column of rows where the 4th column is foo
awk -F"\t" '{print $NF}' bar.bed # ignore spaces and print the last column
```



```
awk -F"\t" '{print $(NF-1)}' bar.bed # print the penultimate column
awk '{sum+=$2} END {print sum}' bar.bed # print the sum of the second column
awk '/^foo/ {sum+=$2; count+=1} END {print sum/count}' bar.bed # print the average of the
# second value of lines starting
# with foo
```

## 6.5 Piping, redirection and more advanced queries

```
grep -hv '^#' bar/*.gff | awk -F"\t" '{print $1}' | sort -u
# grep => -h: don't print file names
#           -v: don't give me matching files
#           '^#': get rid of the header rows
#           'bar/*.gff': only look in the gff files in bar/
# awk => print the first column
# sort => -u: give me unique values
```

```
awk 'NR%10 == 0' bar.bed | head -20
# awk => NR: is the row number
#       NR%10: is the modulo (remainder) of dividing my 10
#       awk is therefore giving you every 10th line
# head => only show the first 20
```

```
awk '{l=($3-$2+1)}; (l<300 && $2>200000 && $3<250000)' exercises.bed
# Gives:
# contig-2 201156 201359 gene-67 24.7 -
# contig-4 245705 245932 gene-163 24.8 +
# Finds all of the lines with features less than 300 bases long which start
# after base 200,000 and end before base 250,000
# Note that this appears to have the action before the pattern. This is
# because we need to calculate the length of each feature before we use it
# for filtering. If they were the other way around, you'd get the line
# immediatly after the one you want:
awk '(l<300 && $2>200000 && $3<250000) {l=($3-$2+1); print $0}' exercises.bed
# Gives:
# contig-2 201156 201359 gene-67 24.7 -
# contig-2 242625 243449 gene-68 46.5 +
```

## 6.6 A script

```
#!/usr/bin/env bash

set -e # stop running the script if there are errors
set -u # stop running the script if it uses an unknown variable
set -x # print every line before you run it (useful for debugging but annoying)

if [ $# -ne 2 ]
```

```
then
    echo "You must provide two files"
    exit 1 # exit the programme (and number > 0 reports that this is a failure)
fi

file_one=$1
file_two=$2

if [ ! -f $file_one ]
then
    echo "The first file couldn't be found"
    exit 2
fi

if [ ! -f $file_two ]
then
    echo "The second file couldn't be found"
    exit 2
fi

# Get the lines which aren't headers,
# take the first column and return the unique values
number_of_contigs_in_one=$(awk '$1 !~ /^#/ {print $1}' $file_one | sort -u | wc -l)
number_of_contigs_in_two=$(awk '/^[^#]/ {print $1}' $file_two | sort -u | wc -l)

if [ $number_of_contigs_in_one -gt $number_of_contigs_in_two ]
then
    echo "The first file had more unique contigs than the second"
    exit
elif [ $number_of_contigs_in_one -lt $number_of_contigs_in_two ]
then
    echo "The second file had more unique contigs"
    exit
else
    echo "The two files had the same number of contigs"
    exit
fi
```

## 6.7 Pro tips

- Always have a quick look at files with `less` or `head` to double check their format
- Watch out for data in headers and that you don't accidentally `grep` some if you don't want them
- Watch out for spaces, especially if you're using `awk`; if in doubt, use `-F"\t"`
- Regular expressions are wierd, build them up slowly bit by bit
- If you did something smart but can't remember what it was, try typing `history` and it might have a record

- `man the_name_of_a_command` often gives you help
- Google is normally better at giving examples (prioritise [stackoverflow.com](https://stackoverflow.com) results, they're normally good)

## 6.8 Build commands slowly

If you wanted me to calculate the sum of all of the scores for genes on contig-1 in a bed file, I'd probably run each of the following commands before moving onto the next:

```
head -20 bar.bed # check which column is which and if there are any headers
head -20 bar.bed | awk '{print $5}' # have a look at the scores
awk '{print $1}' bar.bed | sort -u | less # check the contigs don't look wierd
awk '{print $4}' bar.bed | sort -u | less # check the genes don't look wierd
awk '$4 ~ /gene-/' bar.bed | head -20 # check that I can spot genes
awk '($1 == "contig-1" && $4 ~ /gene-/) bar.bed | head -20 # check I can find
                                     # genes on contig-1

# check my algorithm works on a subset of the data
head -20 bar.bed | awk '($1 == "contig-1" && $4 ~ /gene-/) {sum+=$5}; END {print sum}'
# apply the algorithm to all of the data
awk '($1 == "contig-1" && $4 ~ /gene-/) {sum+=$5}; END {print sum}' bar.bed
```

## 6.9 Which tool should I use?

You should probably use `awk` if:

- your data has columns
- you need to do simple maths

You should probable use `grep` if:

- you're looking for files which contain some specific text (e.g. `grep -r foo bar/`: look in all the files in `bar/` for any with the word 'foo')

You should use `find` if:

- you know something about a file (like it's name or creation date) but not where it is
- you want a list of all the files in a subdirectory and its subdirectories etc.

You should write a script if:

- your code doesn't fit on one line
- it's doing something you might want to do again in 3 months
- you want someone else to be able to do it without asking loads of questions
- you're doing something sensitive (e.g. deleting loads of files)
- you're doing something lots of times

You should probably use `less` or `head`:

- always, you should always use `less` or `head` to check intermediary steps in your analysis

# Module 4

# Mapping Short Reads

## Introduction

The **re-sequencing** of a genome typically aims to capture information on **Single Nucleotide Polymorphisms (SNPs)**, **INsertions and DELETions (INDELs)** and **Copy Number Variants (CNVs)** between representatives of the same species, usually in cases where a reference genome already exists (at least for a very closely related species). Whether one is dealing with different bacterial isolates, with different strains of single-celled parasites, or indeed with genomes of different human individuals, the principles are essentially the same. Instead of assembling the newly generated sequence reads *de novo* to produce a new genome sequence, it is easier and much faster to **align or map the new sequence data to the reference genome** (please note that we will use the terms “aligning” and “mapping” interchangeably). One can then readily identify SNPs, INDELs, and CNVs that distinguish closely related populations or individual organisms and may thus learn about genetic differences that may cause drug resistance or increased virulence in pathogens, or changed susceptibility to disease in humans. One important prerequisite for the mapping of sequence data to work is that the reference and the re-sequenced subject have the same genome architecture. Once you are familiar with viewing short read mapping data you may also find it helpful for quality checking your sequencing data and your *de novo* assemblies.

The computer programme **Artemis** allows the user to view and edit **genomic sequences** and EMBL/GenBank (NCBI) **annotation** entries in a highly interactive graphical format. Artemis also allows the user to view “**Next Generation Sequencing**” (NGS) data from Illumina, 454 or Solid machines.

## Aims

- 1) To introduce the biology & workflow
- 2) To introduce mapping software, BWA, SAMtools, SAM/BAM and FASTQ file format
- 3) To show how **Next Generation Sequencing data** can be viewed in Artemis alongside your chosen reference using *Chlamydia* as an example: navigation, read filtering, read coverage, views
- 4) To show how **sequence variation data** such as SNPs, INDELs, CNVs can be viewed in single and multiple BAM files, and BCF variant filtering
- 5) To show how short-read mapping can be executed with a script, and working with NGS data in eukaryotes: *Plasmodium*

# Background

## Biology

To learn about sequence read mapping and the use of Artemis in conjunction with NGS data we will work with real data from the bacterial pathogen *Chlamydia* as well as the eukaryotic single-celled parasites *Plasmodium* that cause malaria.

### *Chlamydia trachomatis*

*C. trachomatis* is one of the most prevalent human pathogens in the world, causing a variety of infections. It is the leading cause of **sexually transmitted infections (STIs)**, with an estimated 91 million new cases in 1999. Additionally, it is also the leading cause of preventable infectious blindness with some 84 million people thought to have active disease. The STI strains can be further subdivided into those that are restricted to the genital tract and the more invasive type known as the lymphogranuloma venereum or LGV biovar. Despite the large differences in the site of infection and the disease severity and outcome there are few whole-gene differences that distinguish any of the different types of *C. trachomatis*. As you will see most of the variation lies at the level of SNPs.

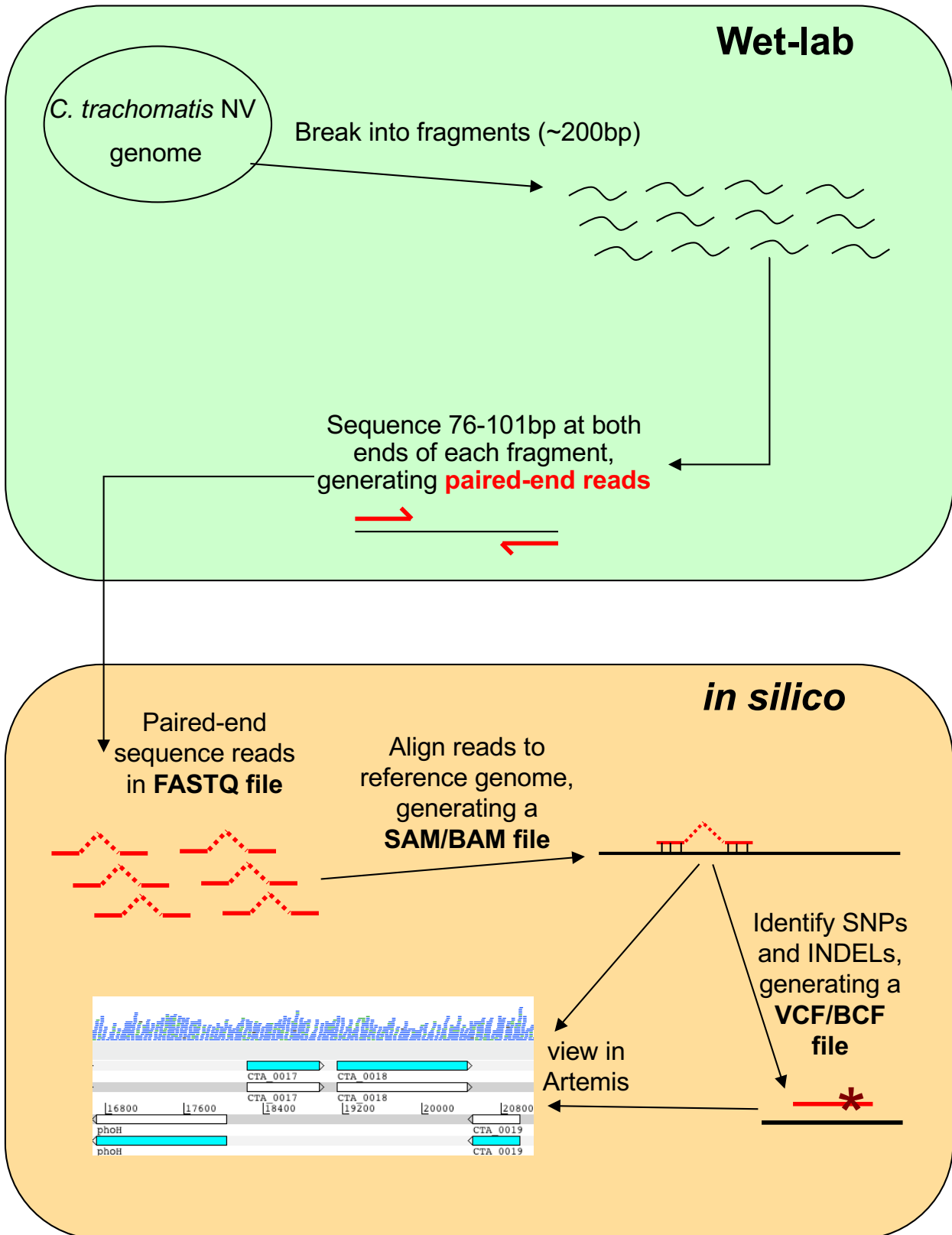
In this part of the course we will align against a reference sequence (L2) the Illumina reads from a recently isolated genital tract new variant Swedish STI *C. trachomatis* strain (known as NV) that caused a European health alert in 2006. During this time it became the dominant strain circulating in some European countries and began to spread world wide. The reason for this was that it **evaded detection by the widely used PCR-based diagnostic test**. During the course of this exercise you will identify the reason why this isolate confounded the standard assay.

### *Plasmodium falciparum*

*P. falciparum* is the causative agent of **the most dangerous form of malaria in humans**. The reference genome for *P. falciparum* strain 3D7 was determined and published about 10 years ago (Gardener et al., 2002). Since then the genomes of several other species of *Plasmodium* that infect humans or animals have been elucidated. Malaria is widespread in tropical and subtropical regions, including parts of Asia, Africa, and the Americas. Each year, there are approximately 350–500 million cases of malaria killing more than one million people, the majority of whom are young children in sub-Saharan Africa.

To date, the genomes of several strains of *P. falciparum* have been sequenced completely. For this exercise we will examine 76bp paired-end sequence read data from the malaria strains Dd2 and IT. In particular the *P. falciparum* Dd2 strain is well known for its **resistance to commonly used antimalarial drugs such as chloroquine**. Working with the mapped sequence data and Artemis we will have a closer look at some SNPs and CNVs that contribute directly to the drug-resistance phenotype of this deadly parasite.

## Workflow of re-sequencing, alignment, and *in silico* analysis



## Short-Read Alignment Software

There are multiple short-read alignment programs each with its own strengths, weaknesses, and caveats. Wikipedia has a good list and description of each. Search for “Short-Read Sequence Alignment” if you are interested. We are going to use BWA:

### BWA: Burrows-Wheeler Aligner

I quote from <http://bio-bwa.sourceforge.net/> the following:

“BWA is a software package for mapping low-divergent sequences against a large reference genome, such as the human genome. It consists of three algorithms: BWA-backtrack, BWA-SW and BWA-MEM. The first algorithm is designed for Illumina sequence reads up to 100bp, while the rest two for longer sequences ranged from 70bp to 1Mbp. BWA-MEM and BWA-SW share similar features such as long-read support and split alignment, but BWA-MEM, which is the latest, is generally recommended for high-quality queries as it is faster and more accurate. BWA-MEM also has better performance than BWA-backtrack for 70-100bp Illumina reads.”

Although BWA does not call Single Nucleotide Polymorphisms (SNPs) like some short-read alignment programs, e.g. MAQ, it is thought to be more accurate in what it does do and it outputs alignments in the SAM format which is supported by several generic SNP callers such as SAMtools and GATK.

BWA has a manual that has much more details on the commands we will use. This can be found here: <http://bio-bwa.sourceforge.net/bwa.shtml>

Li H. and Durbin R. (2009) Fast and accurate short read alignment with Burrows-Wheeler Transform. *Bioinformatics*, 25:1754-60. [PMID: 19451168]

The first thing we are going to do in this Module is to align or map raw sequence read data that is in a standard short-read format (FASTQ) against a reference genome. This will allow us to determine the differences between our sequenced strain and the reference sequence without having to assemble our new sequence data *de novo*.

The FASTQ sequence format is shown over-page.





# 1. Exercise with data from *Chlamydia trachomatis*

To map the reads using BWA follow the following series of commands which you will type on the command line when you have opened up your terminal and navigated into the correct directory. Do a quick check to see if you are in the correct directory: when you type the UNIX command 'ls' you should see the following folders (in blue) and files (in white) in the resulting list.

```
Terminal
manager@pathogens-vm:~/Module_4_Mapping$ ls
L2b.bam  L2b.bam.bai  L2_cat.embl  L2_cat.fasta  malaria  NV_1.fastq.gz  NV_2.fastq.gz
```

## Stage 1:

Our **reference sequence** for this exercise is a *Chlamydia trachomatis* LGV strain called **L2**. The sequence file against which you will align your reads is called **L2\_cat.fasta**.

This file contains a concatenated sequence in FASTA format consisting of the genome and a plasmid. To have a quick look at the first 10 lines of this file, type:

```
head L2_cat.fasta
```

Most alignment programs need to index the reference sequence against which you will align your reads before you begin. To do this for BWA type:

```
bwa index L2_cat.fasta
```

The command and expected output are shown below. Be patient and wait for the command prompt (□~/Module\_2\_Mapping\$) to return before proceeding to Stage 2.

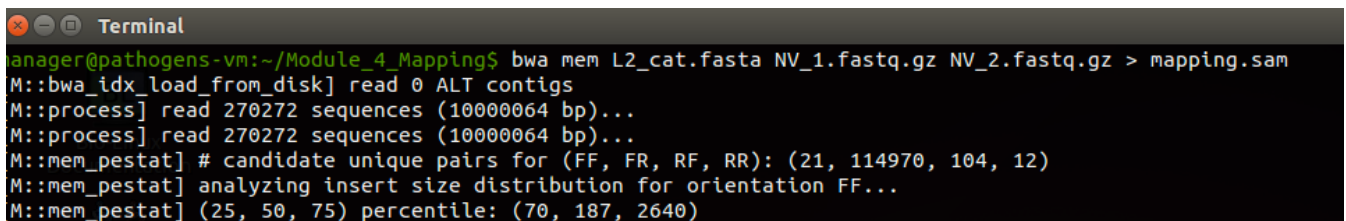
```
Terminal
manager@pathogens-vm:~/Module_4_Mapping$ bwa index L2_cat.fasta
[bwa_index] Pack FASTA... 0.01 sec
[bwa_index] Construct BWT for the packed sequence...
[bwa_index] 0.21 seconds elapse.
[bwa_index] Update BWT... 0.01 sec
[bwa_index] Pack forward-only FASTA... 0.00 sec
[bwa_index] Construct SA from BWT and Occ... 0.11 sec
[main] Version: 0.7.12-r1039
[main] CMD: bwa index L2_cat.fasta
[main] Real time: 0.792 sec; CPU: 0.346 sec
manager@pathogens-vm:~/Module_4_Mapping$
```

**Stage 2:**

We will now align both the forward and the reverse reads against our now indexed reference sequence. The forward and reverse reads are contained in files NV\_1.fastq.gz and NV\_2.fastq.gz, and the output will be saved in SAM format.

Perform the alignment with the following command and wait for it to finish running (it may take a few minutes):

```
□ bwa mem L2_cat.fasta NV_1.fastq.gz NV_2.fastq.gz > mapping.sam
```



```

Terminal
manager@pathogens-vm:~/Module_4_Mapping$ bwa mem L2_cat.fasta NV_1.fastq.gz NV_2.fastq.gz > mapping.sam
M::bwa_idx_load_from_disk] read 0 ALT contigs
M::process] read 270272 sequences (10000064 bp)...
M::process] read 270272 sequences (10000064 bp)...
M::mem_pestat] # candidate unique pairs for (FF, FR, RF, RR): (21, 114970, 104, 12)
M::mem_pestat] analyzing insert size distribution for orientation FF...
M::mem_pestat] (25, 50, 75) percentile: (70, 187, 2640)

```

**Please note:**

The fastq input files provided have been gzipped to compress the large fastq files, many types of software like BWA will accept gzipped files as input.

The last part of the command line `> mapping.sam` determines the name of the output file that will be created in SAM format.

**SAM (Sequence Alignment/Map) format** is a generic format for storing large nucleotide sequence alignments that is illustrated on the next page. Creating our output in SAM format allows us to use a complementary software package called SAMtools.

**SAMtools** is a collection of utilities for manipulating alignments in SAM format. See <http://samtools.sourceforge.net/> for more information. There are numerous options that control the way the SAMtools utilities run, a few of which are explained below. To get brief explanations of the various utilities and the different options or flags that control each utility, type `samtools` or `samtools` followed by one particular utility on the command line like e.g.:

```
samtools
samtools view
```

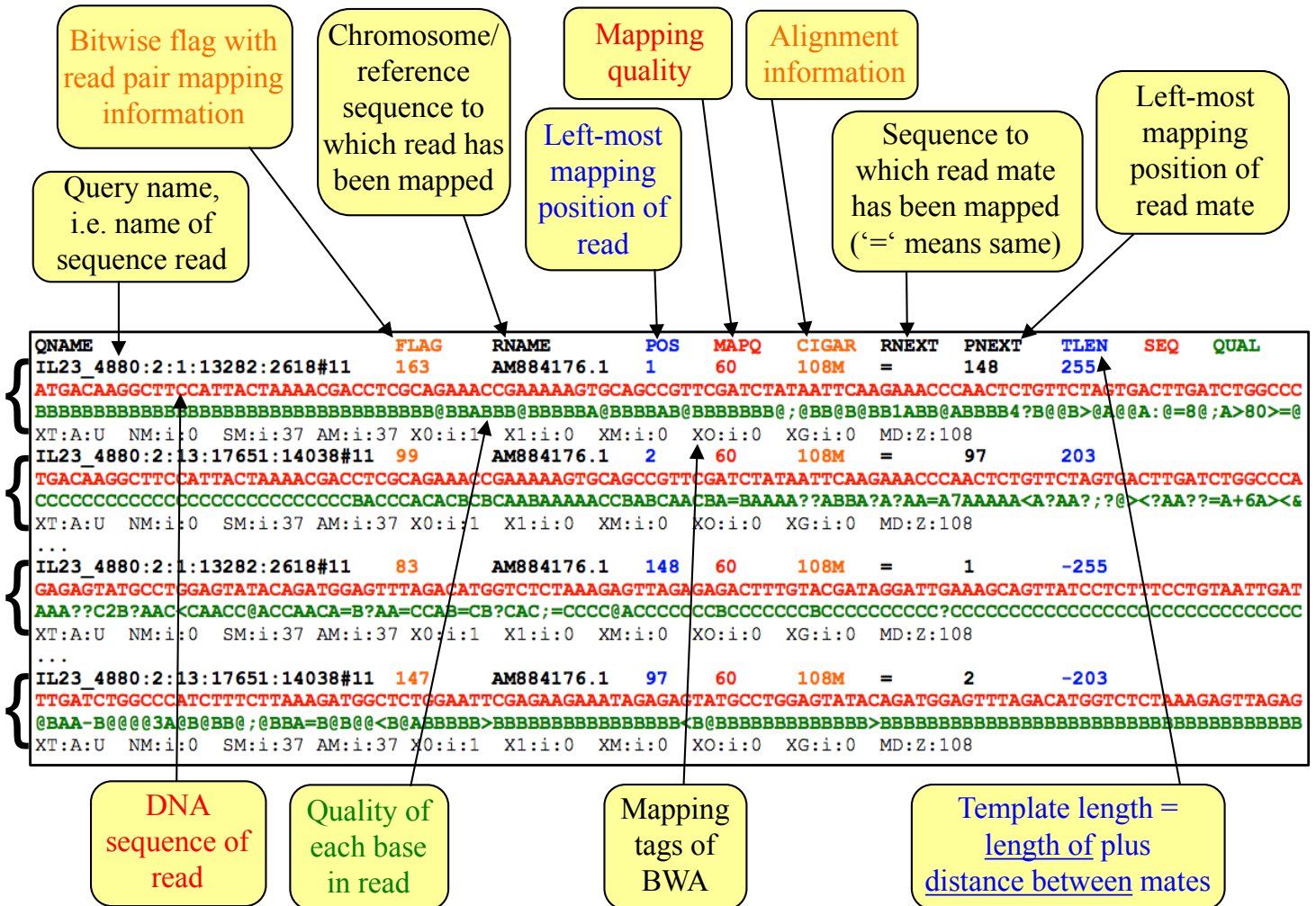
To have a quick look at the first lines of the SAM file you just generated, type:

```
head mapping.sam
```

The SAM/BAM file format is illustrated on the next page.

## File format: SAM / BAM (each line: one aligned sequence read)

The SAM/BAM file format is very powerful. It is unlikely that you will need to work with the contents of a SAM/BAM file directly, but it is very informative to visualize it in a viewer and it is a great format to do further analysis with. The format specifications are at <http://samtools.sourceforge.net/SAM1.pdf>. Below is a brief overview of the information contained in such files.



Next we want to change the file format from SAM to BAM. While files in SAM format store their information as plain text, the BAM format is a binary representation of that same information. One reason to keep the alignment files in BAM rather than in SAM format is that the binary files are a lot smaller than the plain text files, i.e. the BAM format saves expensive storage space (sequence data are generated at an ever increasing rate!) and reduces the time the computer has to wait for slow disk access to read or write data.

Many visualization tools can read BAM files. But first a BAM file has to be sorted (by chromosome/reference sequence and position) and indexed, which enables fast working with the alignments.

**Stage 3:**

To convert our SAM format alignment into BAM format run the following command:

```
samtools view -q 15 -b -S mapping.sam > mapping.bam
```



Flag: output in BAM format    Flag: input in SAM format – **note: this is a capital S**

```
Terminal
manager@pathogens-vm:~/Module_4_Mapping$ samtools view -q 15 -b -S mapping.sam > mapping.bam
[samopen] SAM header is present: 2 sequences.
```

Note the 'flag' `-q 15` tells the program to discard sequence reads that are below a minimum quality score. Poor quality reads will therefore not be aligned.

**Stage 4:**

Next we need to sort the mapped read sequences in the BAM file by typing this command:

```
samtools sort mapping.bam NV
```

← Prefix for output file

This will take a little time to run.

By default the sorting is done by chromosomal/reference sequence and position.

```
Terminal
manager@pathogens-vm:~/Module_4_Mapping$ samtools sort mapping.bam NV
```

**Stage 5:**

Finally we need to index the BAM file to make it ready for viewing in Artemis:

```
samtools index NV.bam
```

```
Terminal
Files
manager@pathogens-vm:~/Module_4_Mapping$ samtools index NV.bam
```

**Stage 6:**

We are now ready to open up Artemis and view our newly mapped sequence data.

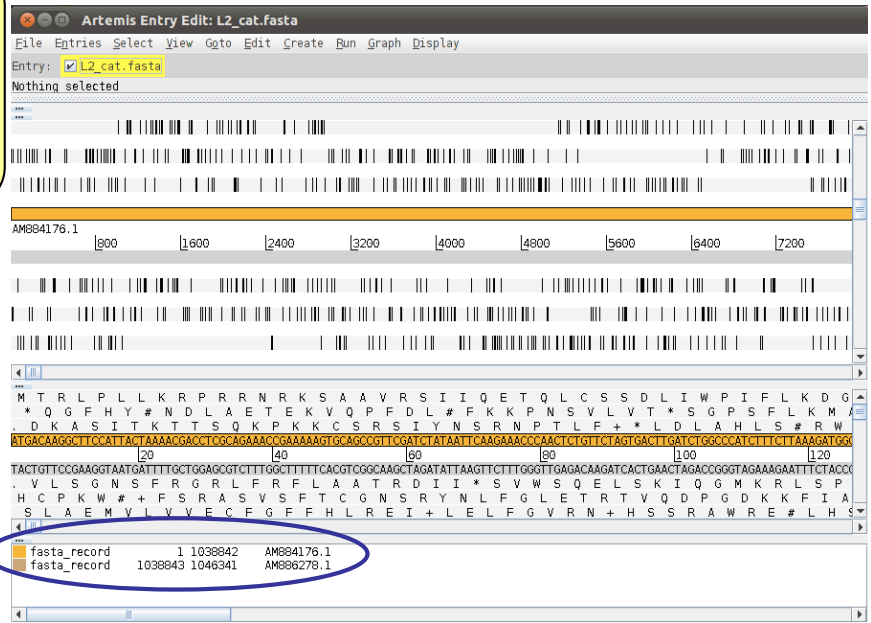
### 1. Start up Artemis.

Double click on the Artemis Icon or type 'art &' on the command line of your terminal window and press return. We will read the reference sequence into Artemis that we have been using as a reference up until now.

Once you see the initial Artemis window, open the file `L2_cat.fasta` via **File – Open**. Just to remind you, this file contains a concatenated sequence consisting of the *C. trachomatis* LGV strain 'L2' chromosome sequence along with its plasmid.

Hopefully you will now have an Artemis window like this!  
If not, please ask a demonstrator for assistance.

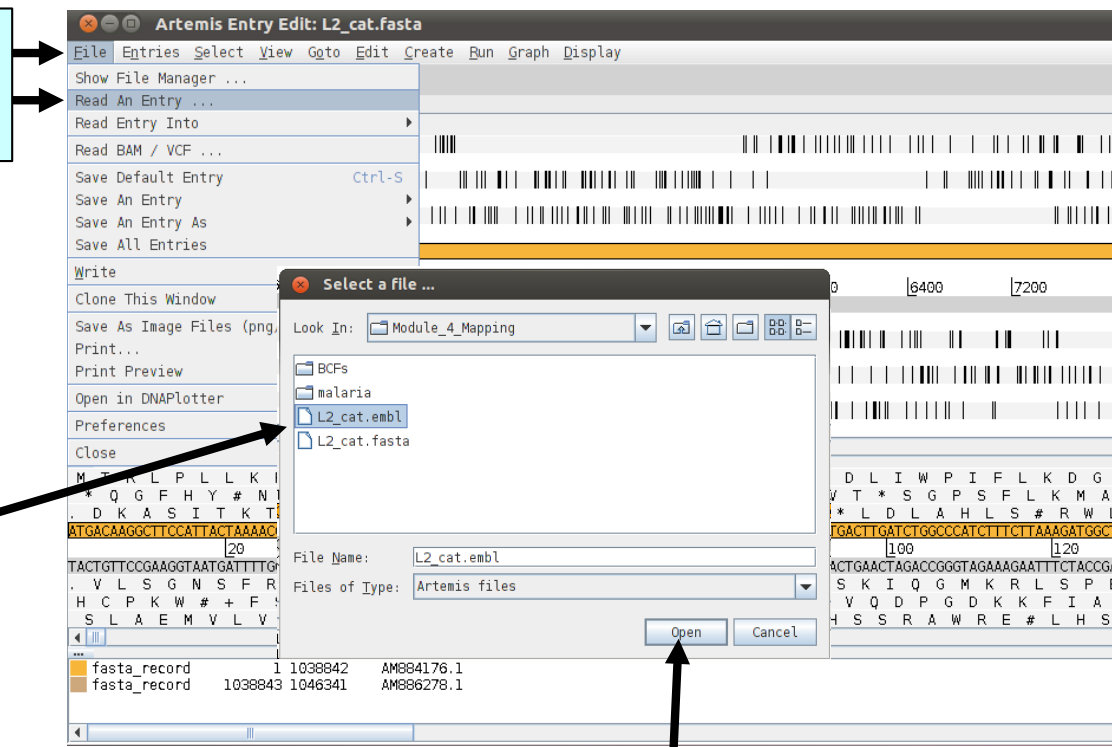
Since the `L2_cat.fasta` is a concatenation of two DNA sequences (chromosome and plasmid) it draws two features automatically to represent them, one in orange and the other brown.



### 2. Now load up the annotation file for the *C. trachomatis* LGV strain L2 chromosome.

1 Click 'File' then 'Read An Entry'

2 Single click to select EMBL file

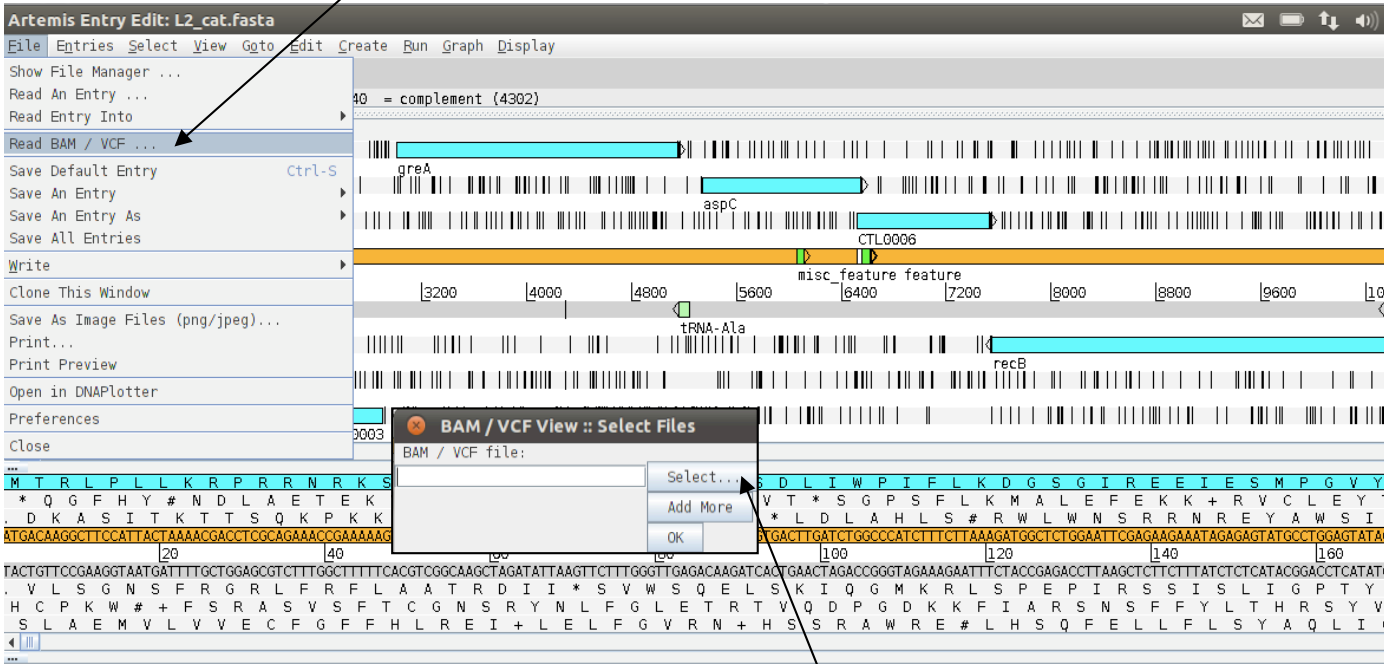


3 Single click to open file in Artemis then wait

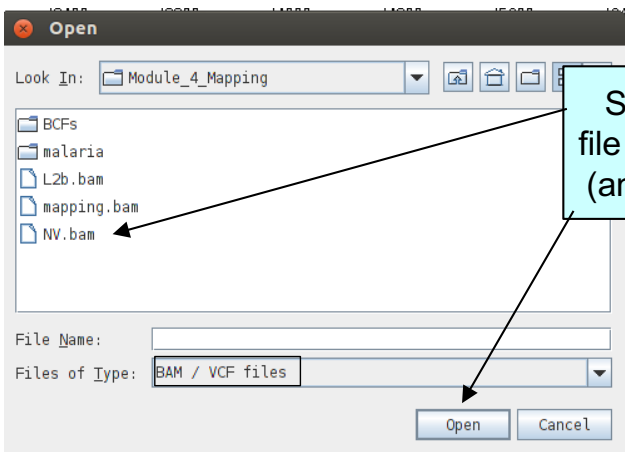


To examine the read mapping we have just performed we are going to read our BAM file containing the mapped reads into Artemis as described below. Please make sure you do not go to a zoomed-out view of Artemis, but stay at this level, as display of BAM files does take time to load!

1 Read in a BAM file

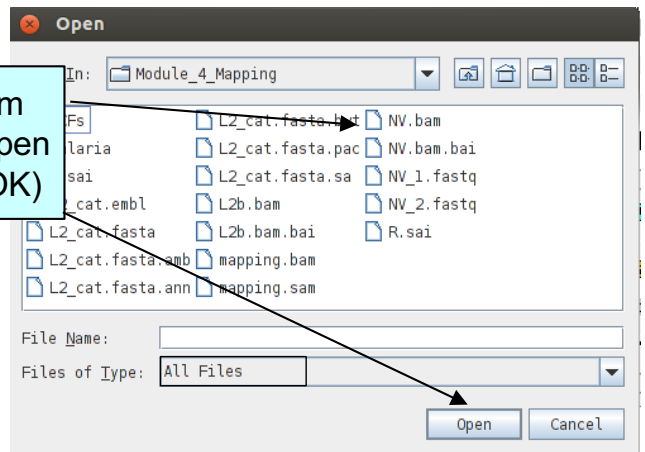


2 Click Select



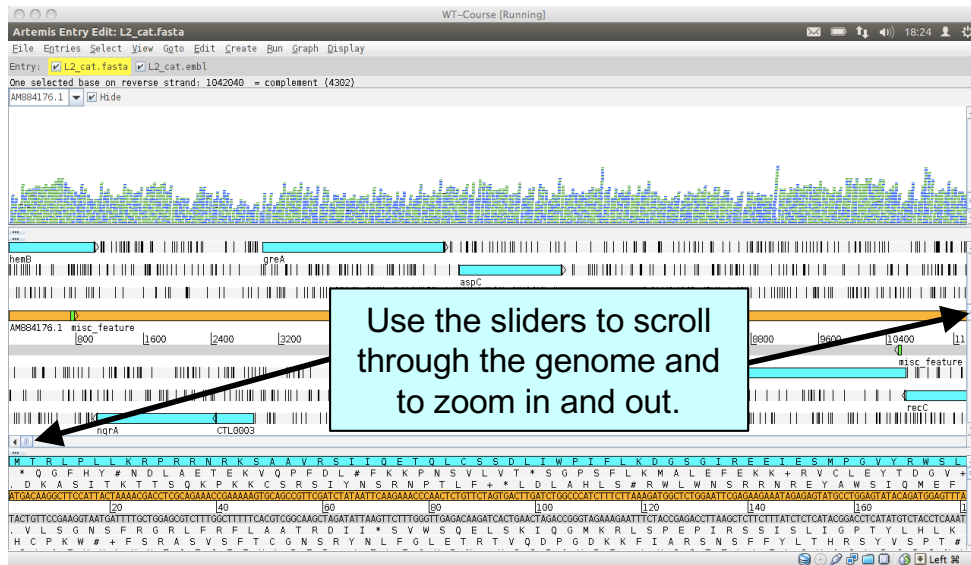
3

Select NV.bam file and click Open (and then on OK)

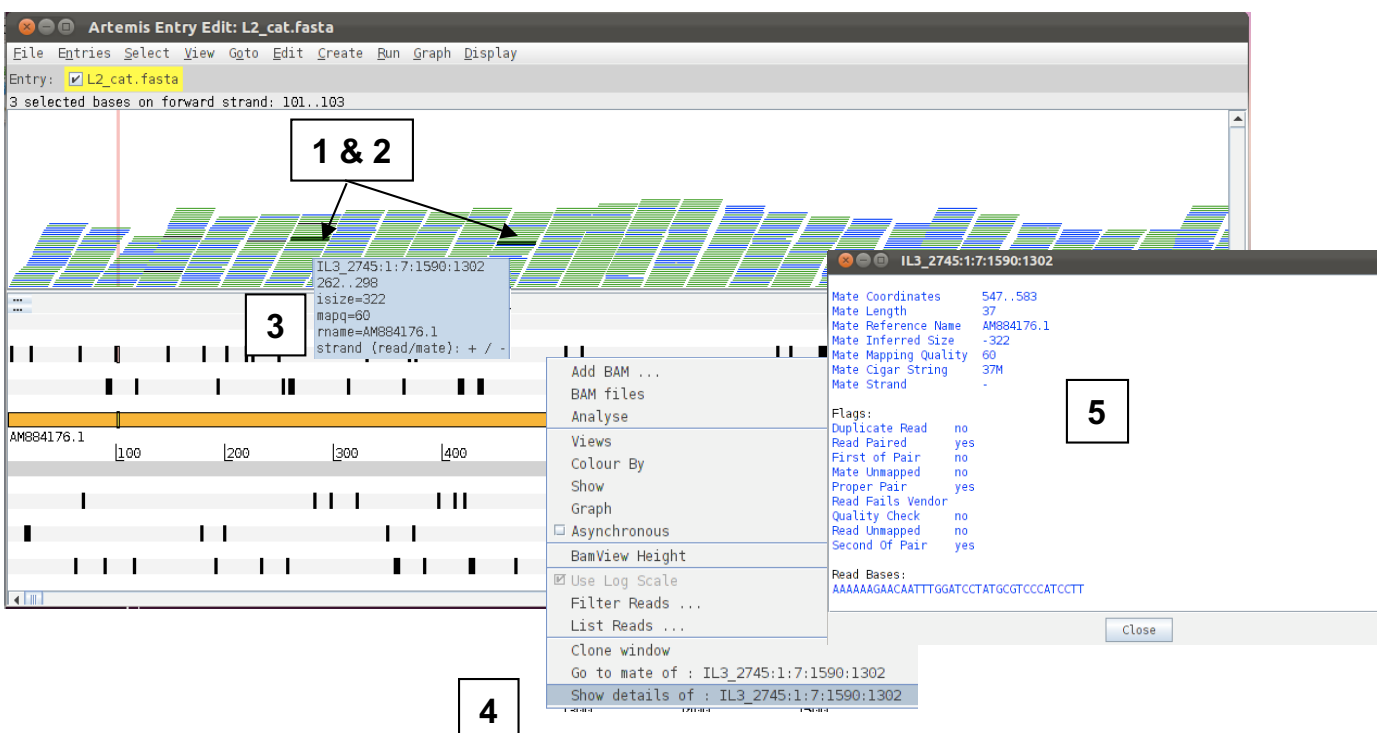




You should see the BAM window appear as in the screen shot below. Remember these reads are of the Swedish NV strain mapped against the LGV strain L2 reference genome. In the top panel of the window each little horizontal line represents a sequencing read. Notice that some reads are blue which indicates that these are unique reads, whereas green reads represent “duplicated” reads that have been mapped to exactly the same position on the reference sequence. To save space, if there are duplicated reads only one is shown, which means that there could be a large number of duplicated reads at a given position but the software only depicts one.

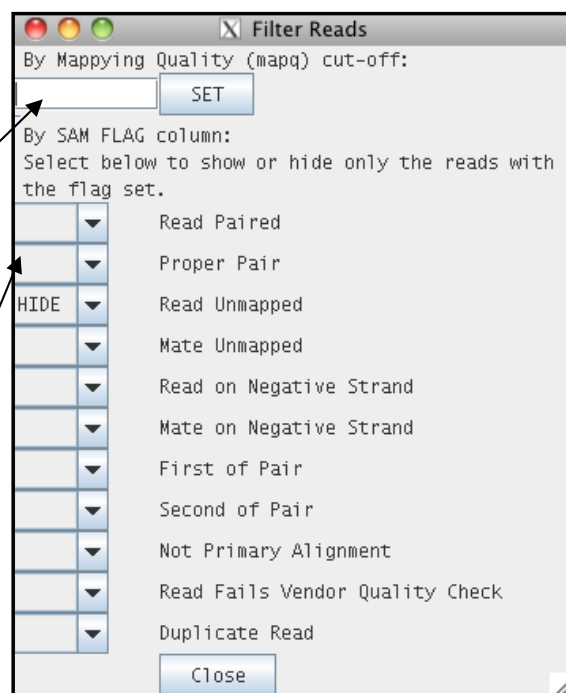
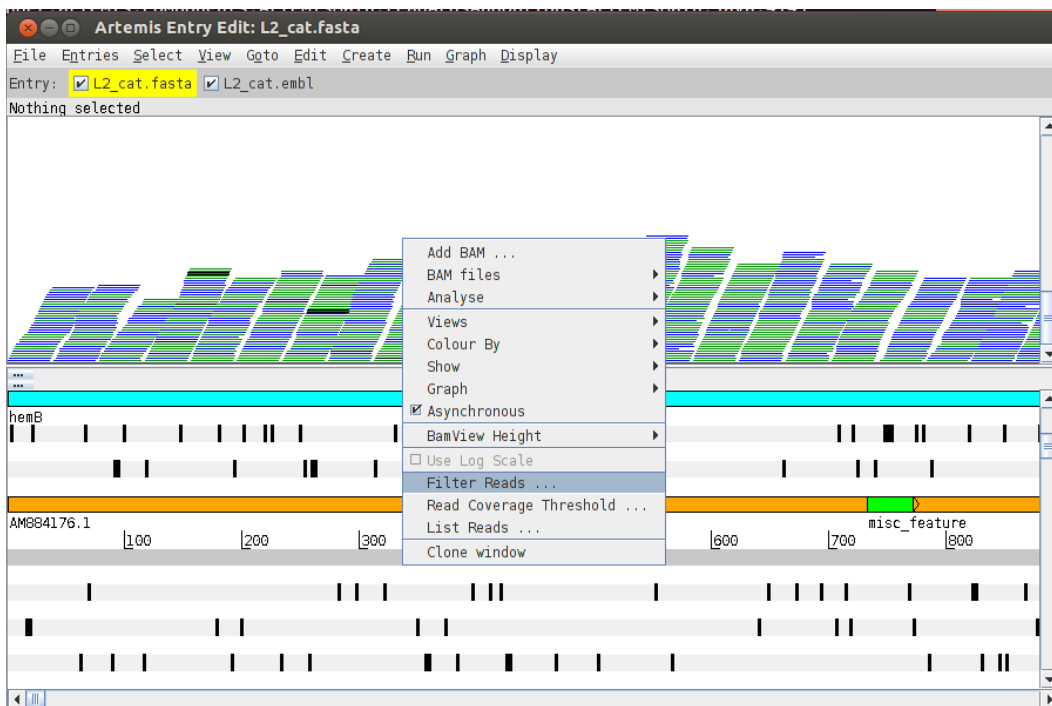


If you click a read (1 & 2) its mate pair will also be selected. Also note that if the cursor hovers over a read for long enough details of that read will appear in a small box (3). If you want to know more then right-click and select ‘Show details of: READ NAME’ from the menu (4). A window will appear (5) detailing the mapping quality (see over page), coordinates, whether it’s a duplicated read etc. If this read(s) covers a region of interest, being able to access this information easily can be really helpful.



**“Mapping quality”** - The mapping quality depends on the number of mismatches between the read and the reference sequence as well as the repetitiveness of the reference sequence. The maximum quality value is 99, whereas a value of 0 means that the read mapped equally well to at least one other location and is therefore not reliably mapped.

You can actually use several details relating to the mapping of a read to filter the reads from the BAM file that are shown in the window. To do this, right-click again over the stack plot window showing the reads and select “Filter Reads...”. A window will appear with many options for filtering, as shown below.

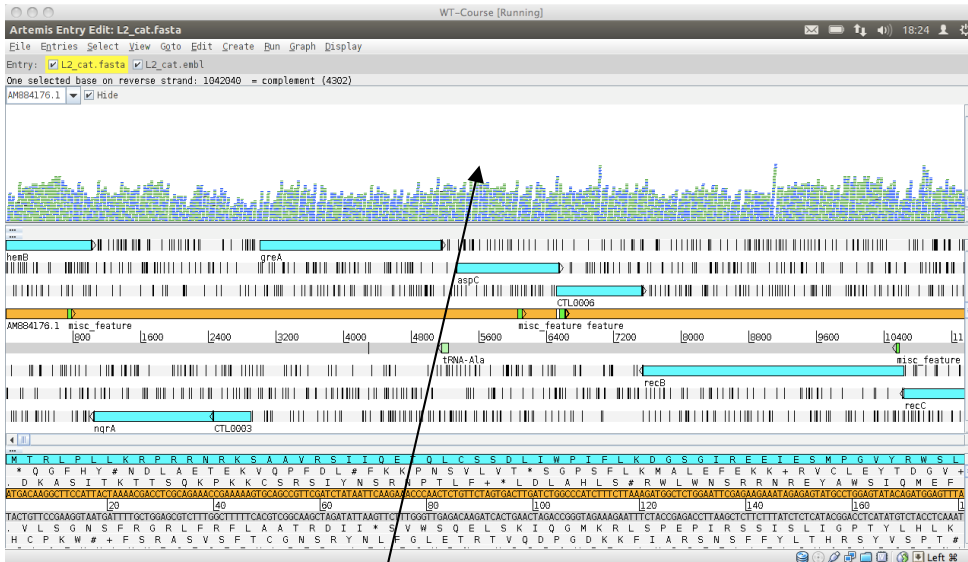


Reads with less than the mapping quality are not shown. Try 60.

HIDE the proper pairs. What happened?

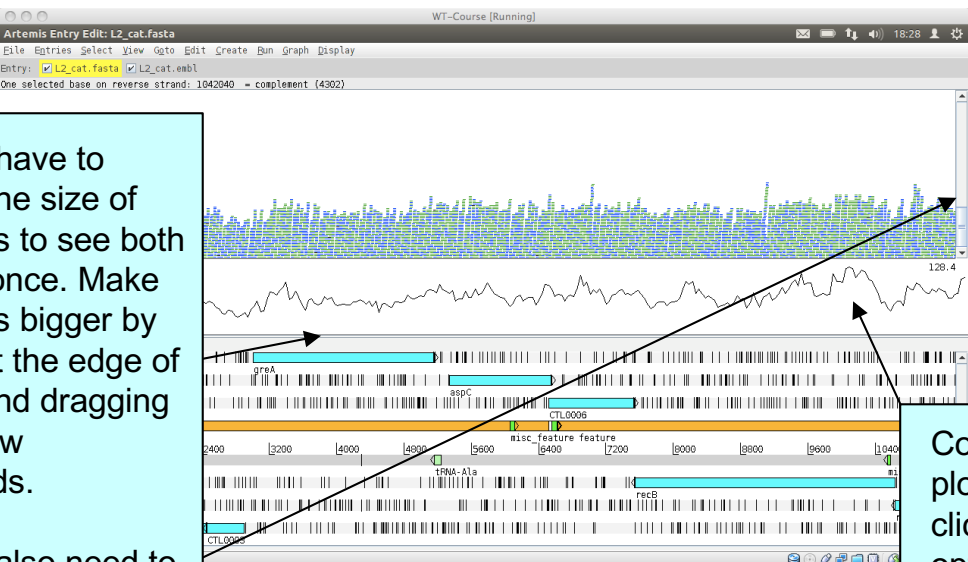
Filtering reads for repetitive regions or seeing properly-paired-reads only can be really helpful.

As mentioned before, to save space if there are duplicated reads only one is represented. But often one may want to know the actual read coverage on a particular region or see a graph of this coverage. You can do this by adding additional graphs as detailed below.



1

There are different views and graphs to display that you can choose from, for example: right click here and select 'Graph' then 'Coverage' from the menu. See below.



You may have to readjust the size of the panels to see both views at once. Make the panels bigger by clicking at the edge of a panel and dragging the window downwards. You may also need to use this slider to adjust the Stack View too.

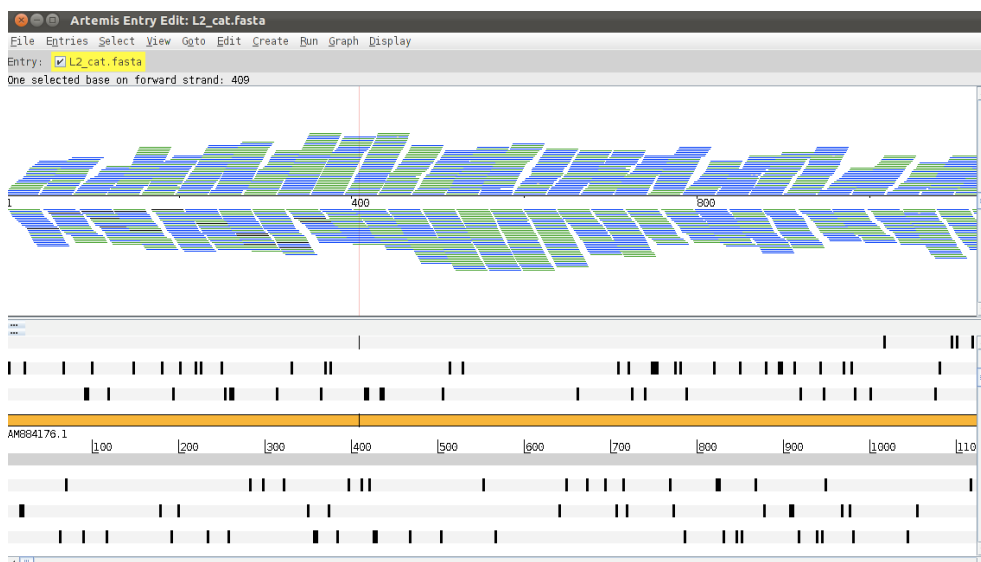
2

Coverage plot. Right click for more options.

There are several other ways to view your aligned read information. Each one may only be subtly different but they are very useful for specific tasks as hopefully you will see. To explore the alternative read views right-click in the BAM panel (1 below) and select the 'Views' menu option (2 below):

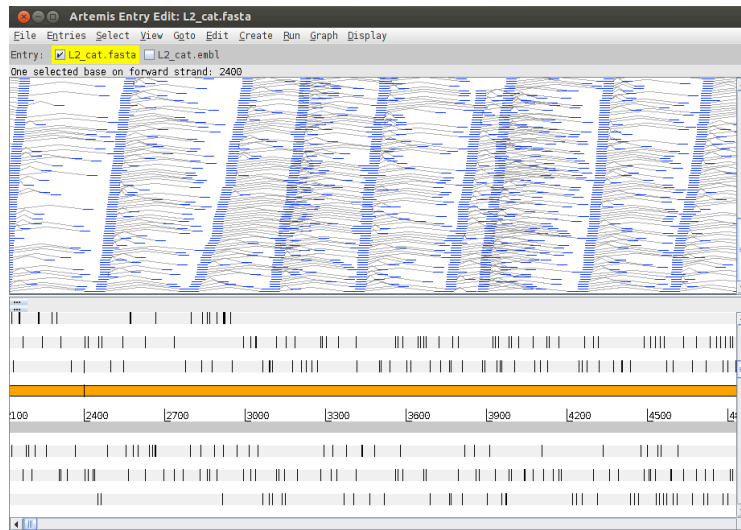


- We have already looked at '**Stack**' view.
- The **Coverage** view: just like adding the coverage plot above you can also convert the Stack view to a coverage view. This can be useful when multiple BAM files are loaded as a separate plot is shown for each. You can also look at the coverage for each strand individually by using the **Coverage by Strand** option. You can now also view the coverage as a **Heat Map**, with darker colours displaying higher coverage.
- The '**Strand Stack**' view (shown below), with the forward and reverse strand reads above and below the scale respectively. Useful for strand specific applications or for checking for strand-specific artifacts in your data. See picture below.



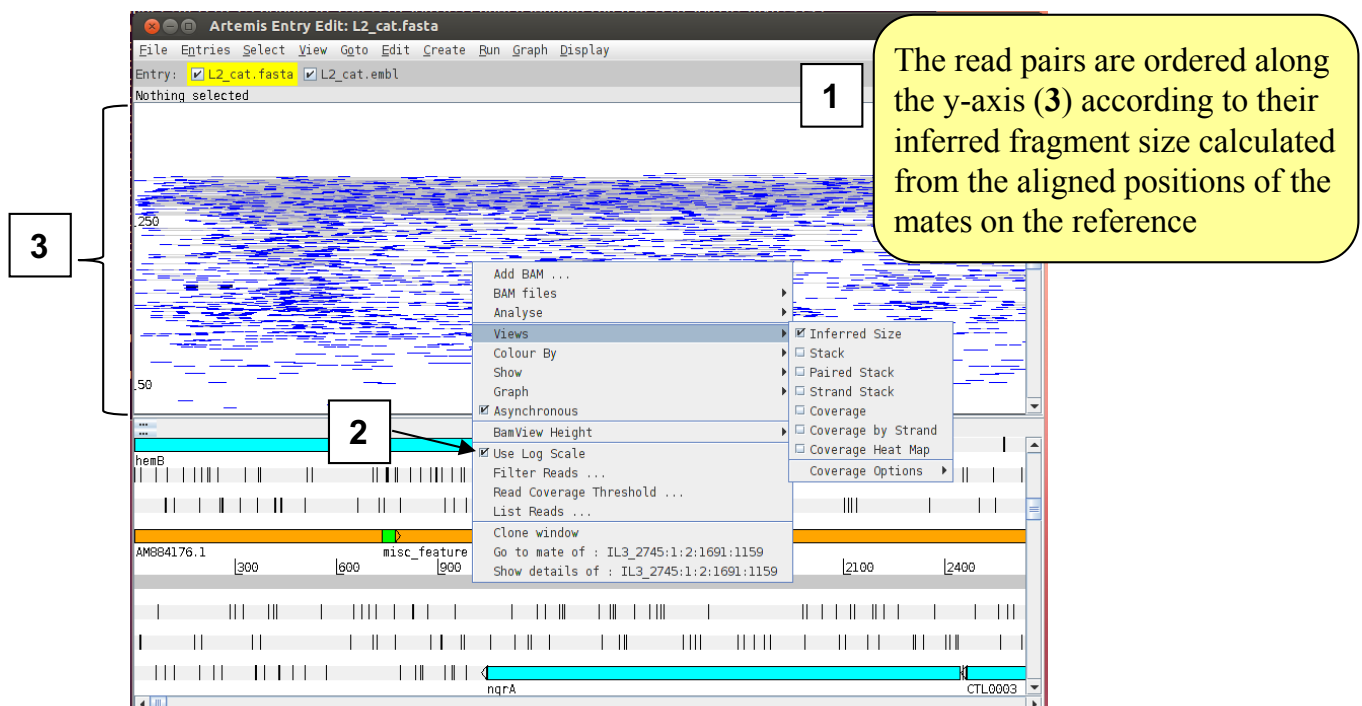
Alternative views continued:

d) The **'Paired Stack'** view (inverted reads are red) joins paired reads. This can be useful to look for rearrangements and to confirm that regions are close together in the reference and the genome from which the aligned reads originate.



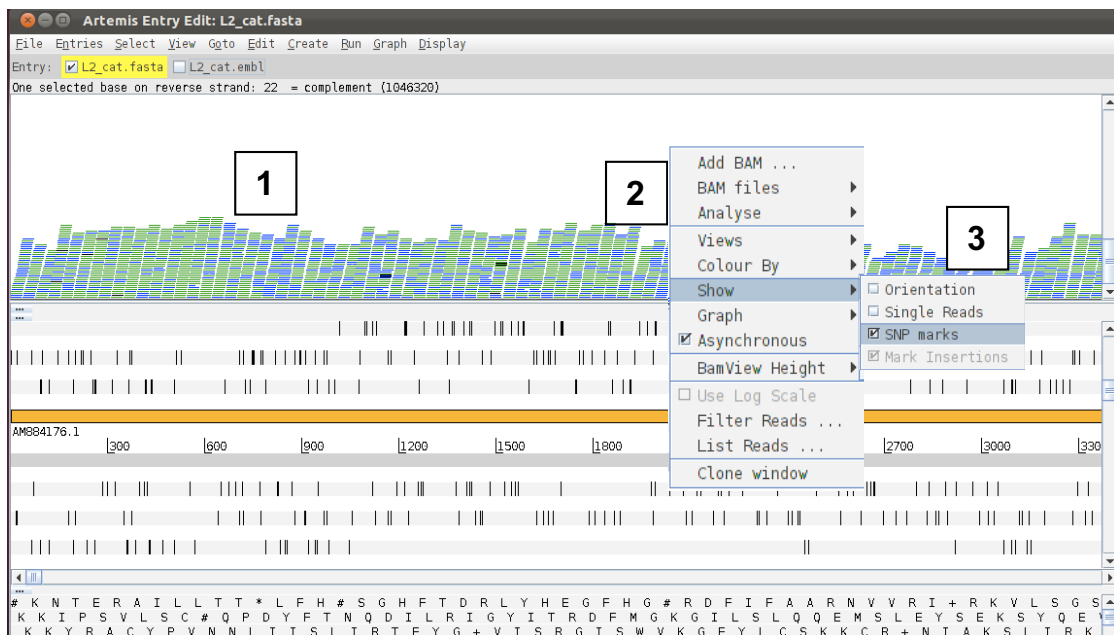
e) The **'Inferred Size'** is similar to the 'Paired Stack' view, but it orders the read pairs along the y-axis by their inferred insert size which is calculated from the aligned positions of the mates on the reference sequence (1). Optionally you can display the inferred insert sizes on a log scale (2). Note that Illumina libraries are usually made from size fractionated DNA fragments of about 250bp-500bp.

So this is not the actual library fragment size, although you would expect it to correlate closely, and be relatively constant, if your reference was highly conserved with the sequenced strain. The utility of this can seem a little obscure but its not and can be used to look for insertions and deletions as will be shown later in this Module.

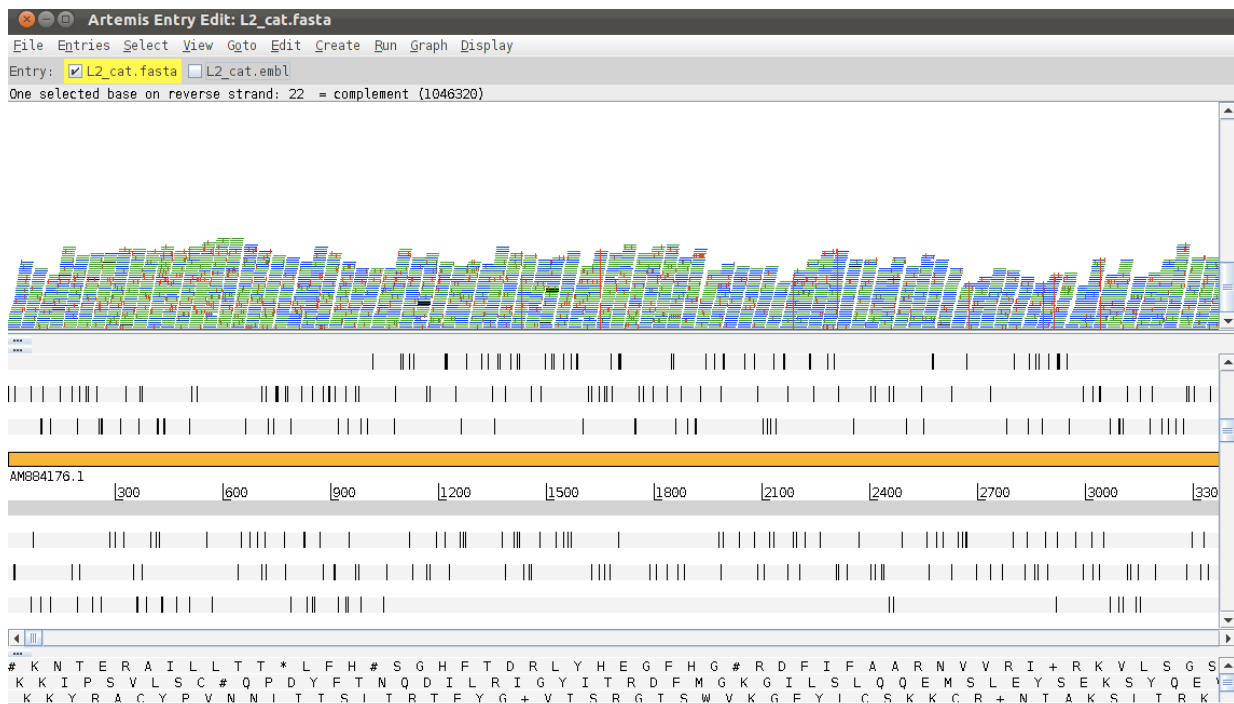


## Viewing SNPs

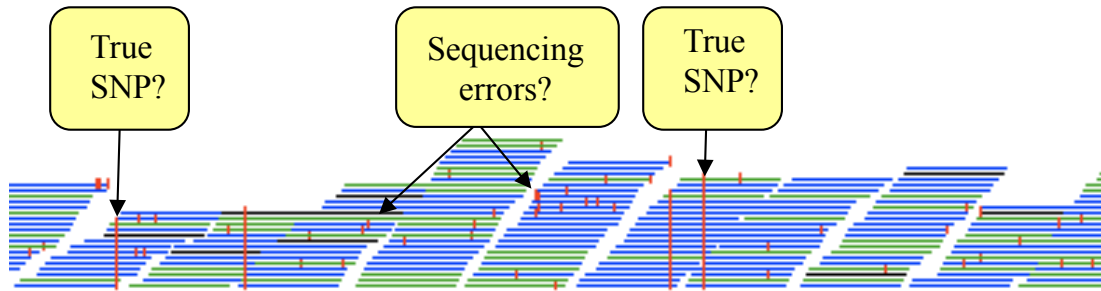
Start by returning your view back to 'Stack' view.



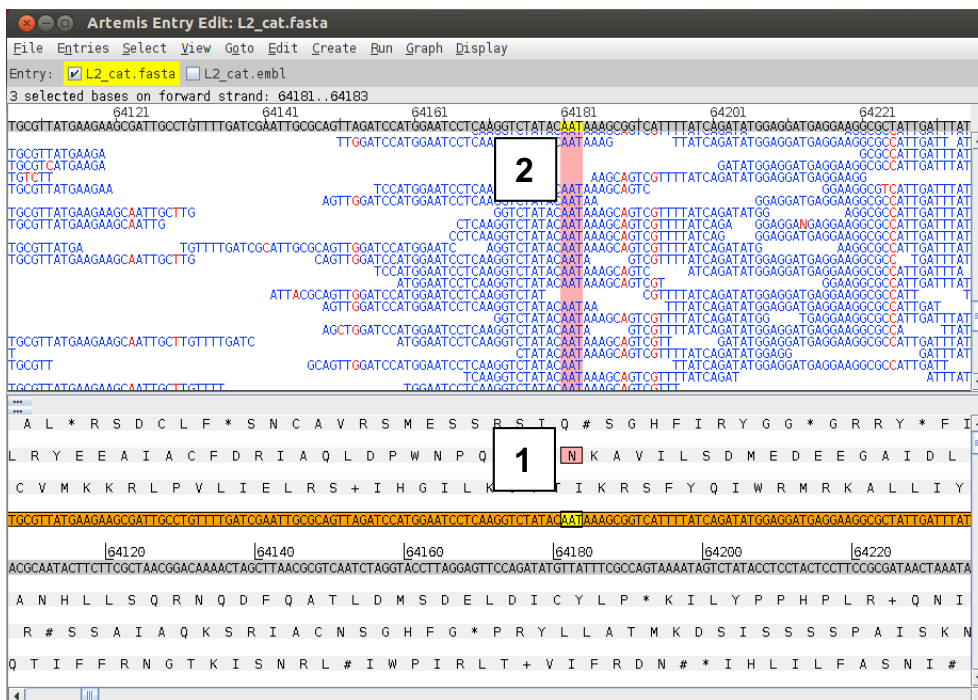
To view SNPs use your right mouse button to click in the BAM view window (the panel showing the coloured sequence reads; **1 see above**). Then in the popup menu click on **2 'Show'** and **3** and check the 'SNP marks' box. SNPs in your data in comparison to the reference sequence are shown as red marks on the individual reads as shown below.



In other words, the red marks appear on the stacked reads highlighting every base in a read that does not match the reference. When you zoom in you can see some SNPs that are present in all reads and appear as vertical red lines, whereas other SNPs are more sporadically distributed. The former are more likely to be true SNPs whereas the latter may be sequencing errors, although this would not always be true.



If you zoom in further, the sequence of the individual sequence reads and the actual SNPs become visible, with the reference sequence highlighted in grey at the top. If you click on amino acids or bases in the sequence view (1), they will be highlighted in the sequence reads (2).



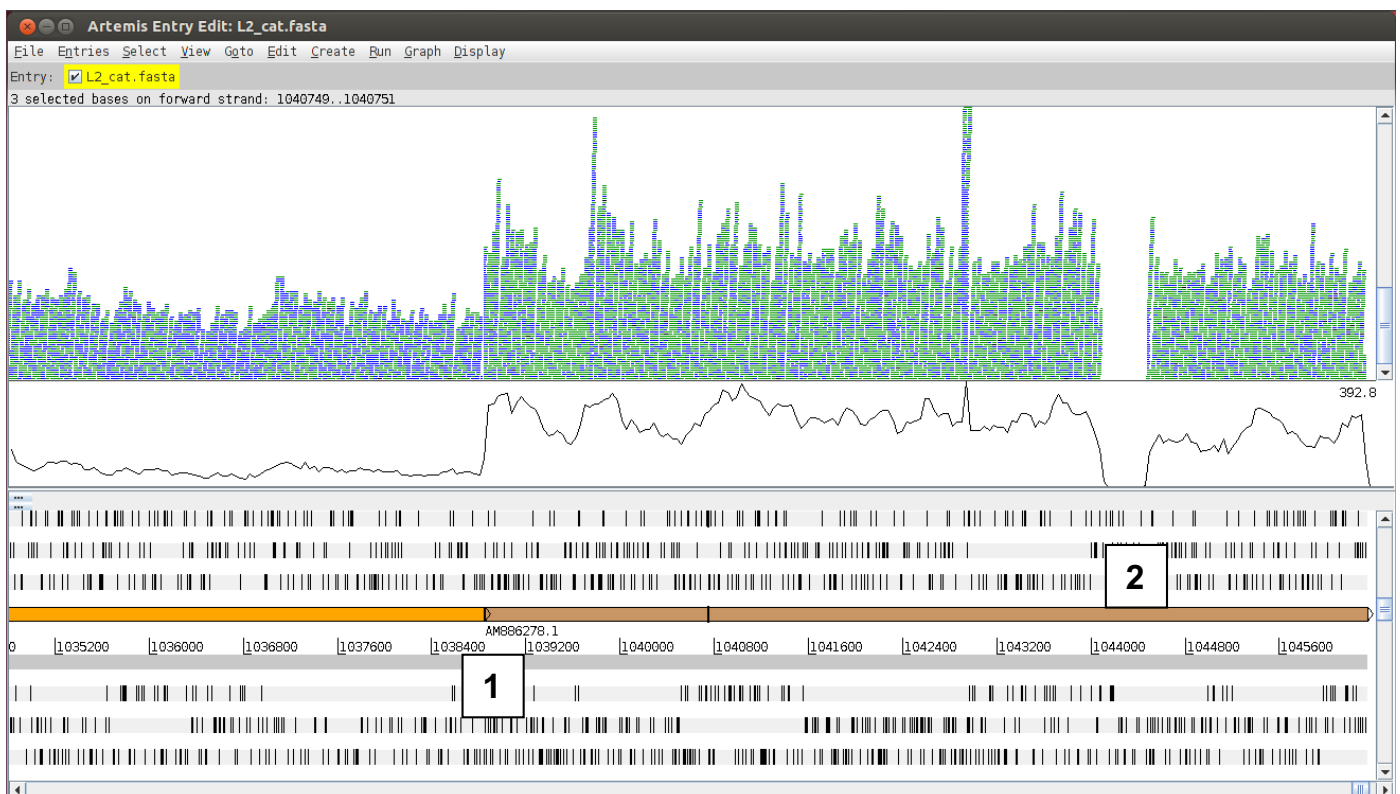
Many SNP examples are quite clear, however this is not always the case. What if the read depth is very low? If there are only two reads mapping, the reference is T and both reads are C is this enough evidence to say that the genomes are different? What if there are many reads mapping and out of e.g. 100 base calls at a particular position 50 are called as G and 50 are called as T: this could be due to a mixed infection/population that was sequenced, or this would be typical for a heterozygous locus in a diploid genome...



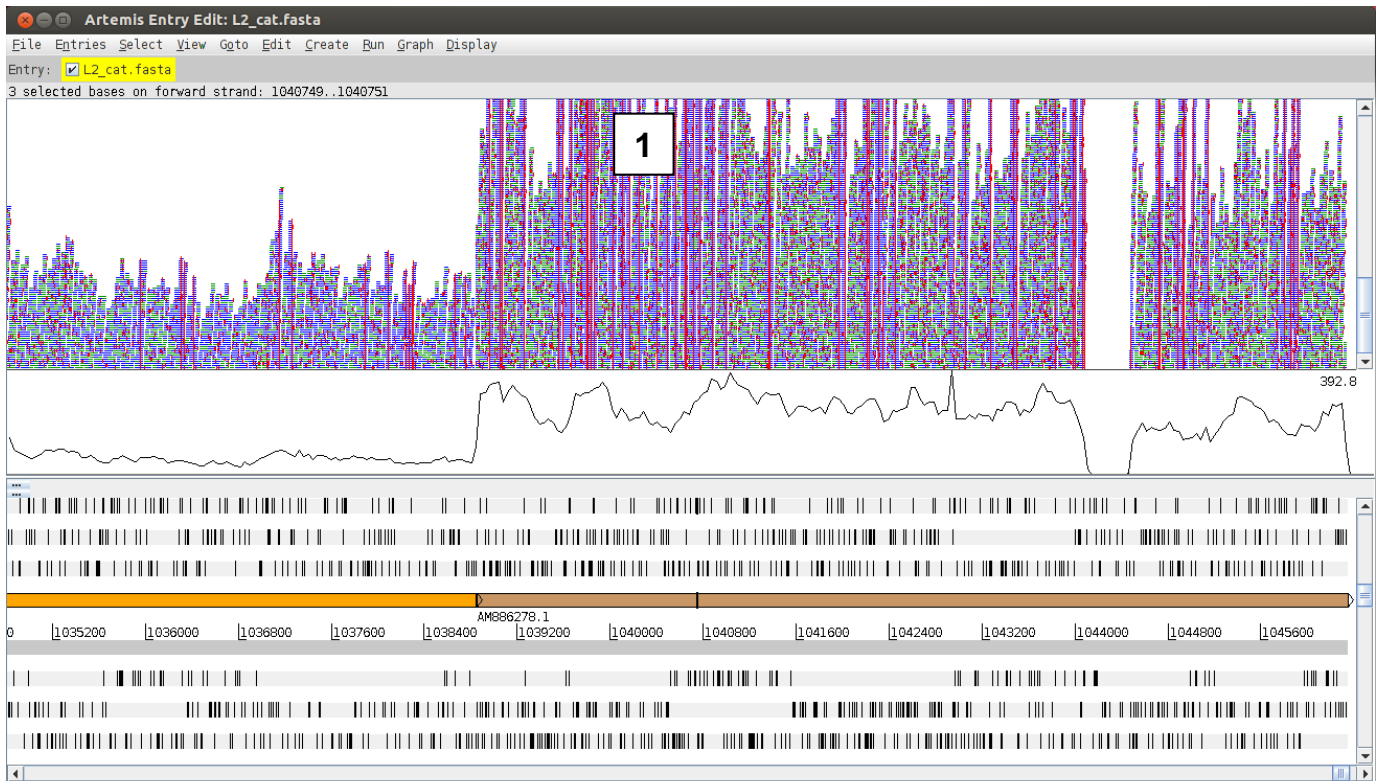
To give you a good biological example for when this type of information and analysis can be really informative and valuable, now do the following: using either the sliders, the GoTo menu or the 'Navigator', go to the end of the sequence or to base position 1043000. Adjust your view so you are in Stack view and have the depth of coverage graph showing. You might also need to adjust the Artemis window as well as the different panels.

If you adjust the zoom using the side sliders you should get a view similar to the one below. Notice two things: 1) the depth of coverage steps up at the beginning of the brown DNA line feature and 2) the coverage falls to zero within a region of this feature.

What could this mean?



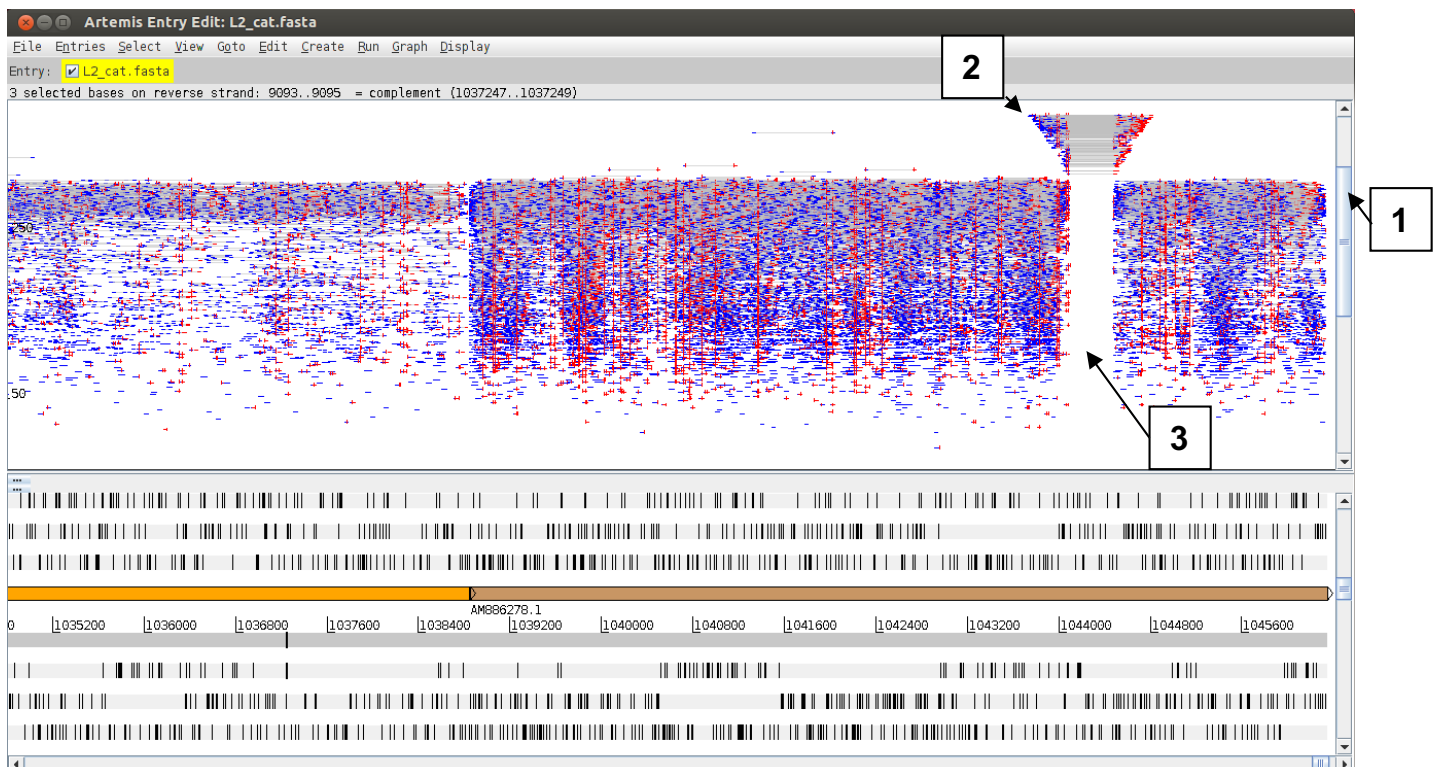
Note that the display changes when you switch on the display of SNPs (right click – Show SNP marks). This is due to a difference in display of duplicate reads. Reads having the same start and end position after mapping are considered duplicates and are displayed in green in the bam view. However, apart from the true SNPs, these duplicate reads are likely to differ in the sequencing errors, thus have to be displayed individually when the SNPs are displayed (1).



Coming back to the increase in coverage, the answer is that since part of the sequence you have been viewing is a plasmid (brown DNA feature) it is present in multiple copies per cell, whereas the chromosome is only present in one copy per cell (orange DNA feature). Therefore each part of the plasmid is sequenced more often than the rest of the genome leading to a higher read coverage in this area of the plot.

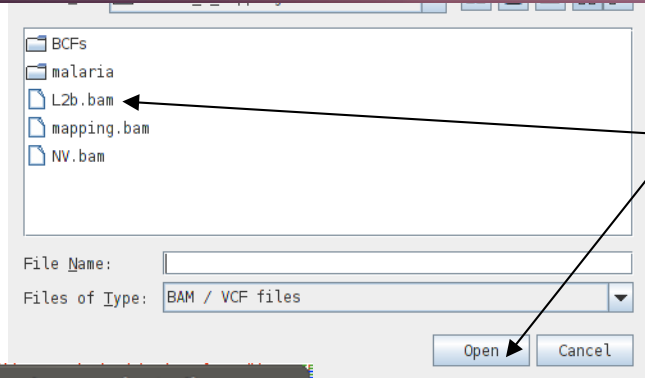
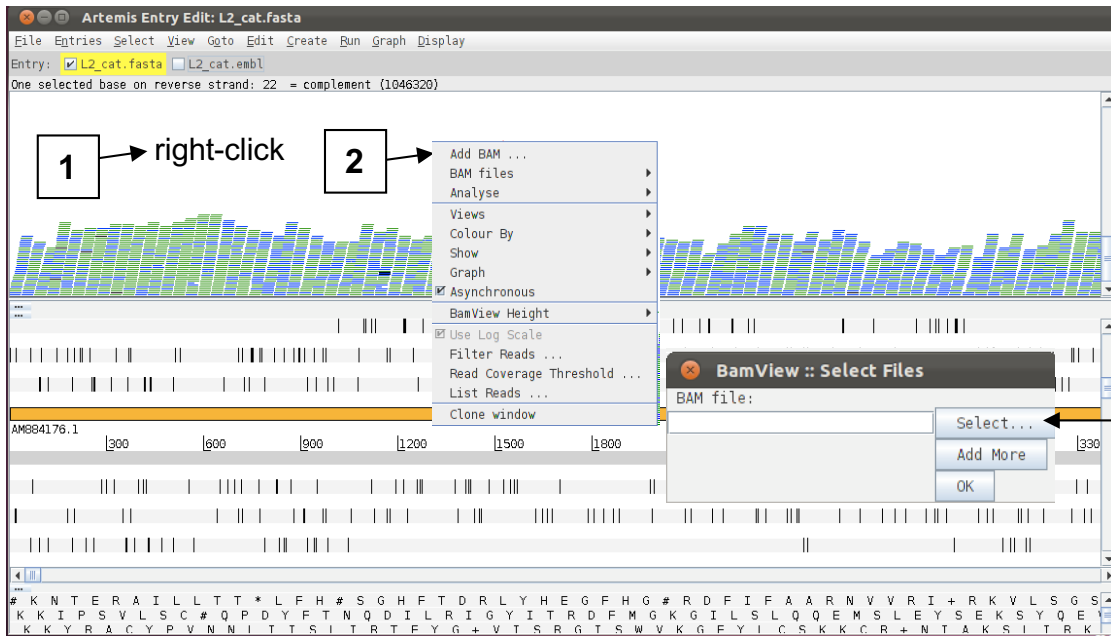
What about the region in the plasmid where no reads map?

This is where the Inferred Size view for the reads is useful. If you change the view as before to 'Inferred Size' and use the log scale you will see an image similar to the one below. You may have to adjust the view (1) to actually see the subset of reads that are shown above almost all other reads in this plot (2). The inferred insert size calculated from the alignment for this subset of reads is far bigger than the normal size range of other read pairs in this region (2) and there are no grey lines linking paired reads within the normal size range crossing this region (3). Together, this is indicative of a **deletion** in the DNA of the sequenced strain compared to the reference!

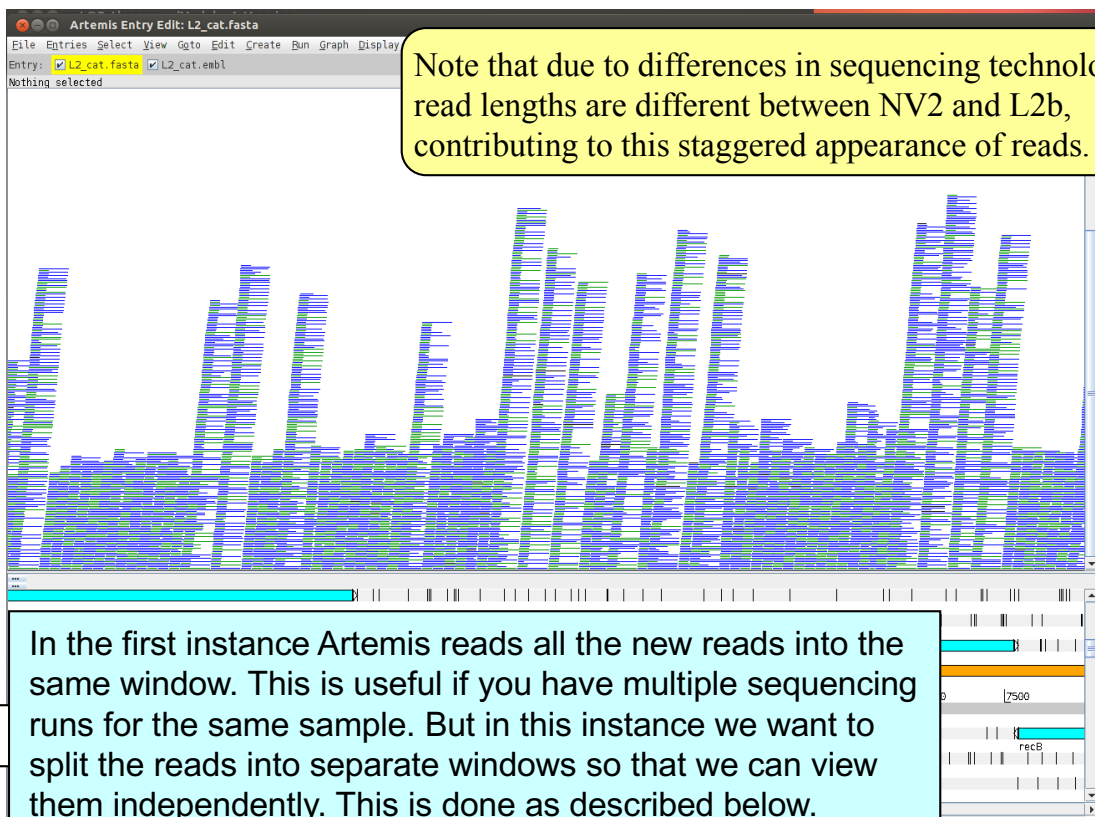
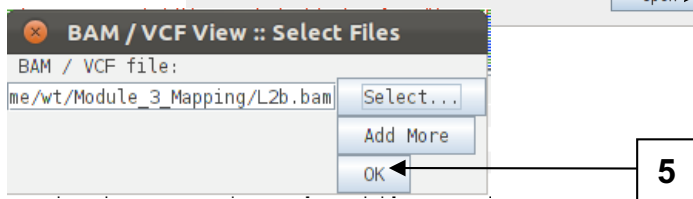


You can also view **multiple BAM files at the same time**. Remember that a BAM file is a processed set of aligned reads from (in this case) one bacterium aligned against a reference sequence. So in principle we can view multiple different bacterial isolates mapped against the same reference concurrently. The *C. trachomatis* isolate you are going to read in is *C. trachomatis* strain L2b. It is more closely related to the reference sequence that we have been using, hence the similar name.

We are not going to redo the mapping for a new organism, instead we have pre-processed the relevant FASTQ data for you. The file you will need is called **L2b.bam**. Follow the instructions below. Start by going back to a normal stacked read view and zooming in more detail.



4 → Within the Module 3 directory choose L2b.bam



Artemis Entry Edit: L2\_cat.fasta

File Entries Select View Goto Edit Create Run Graph Display

Entry:  L2\_cat.fasta  L2\_cat.embl

Nothing selected

Nothing selected

7 First, **clone** the BAM view window. Right-click over the BAM window and select 'Clone window'.

AM884176.1 misc\_feature 800 1600 2400 3200 4000 4800 5600 6400 7200 8000 8800 9600 10400 11200

Artemis Entry Edit: L2\_cat.fasta

File Entries Select View Goto Edit Create Run Graph Display

Entry:  L2\_cat.fasta  L2\_cat.embl

Nothing selected

Nothing selected

8 If you right-click over the top BAM window and select BAM files you can individually select the files as desired. This means you can display each BAM file in its own window by de-selecting one or the other file.

AM884176.1 misc\_feature 800 1600 2400 3200 4000 4800 5600 6400 7200 8000 8800 9600 10400 11200

Now go back to the plasmid region at the end of the genome sequence and have a look at the previously un-mapped region located around base position 1044200. You can see that the newly added BAM file (for L2b) shows no such deletion with reads covering this region (as shown below). Have a look at the inferred read sizes, too.

Artemis Entry Edit: L2\_cat.fasta

File Entries Select View Goto Edit Create Run Graph Display

Entry:  L2\_cat.fasta  L2\_cat.embl

Nothing selected

Nothing selected

AM884176.1  Hide Close

CTL0895

tRNA-Pro AM886278.1

repeat\_region repeat\_r1 repeat\_region r1 repeat\_region r1 repeat\_region r1

335200 1036000 1036800 1037600 1038400 1039200 1040000 1040800 1041600 1042400 1043200 1044000 1044800 1045600

pl2\_02



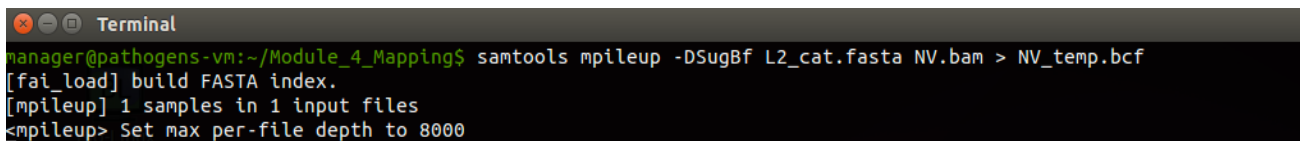
## Looking at SNPs in more detail

So far we have looked at SNP variation rather superficially. In reality you would need more information to understand the effect that the sequence change might have on for example coding capacity. For this we can view a different data type called **Variant Call Format (VCF)**. In analogy to the SAM/BAM file formats, VCF files are essentially plain text files while **BCF** files represent the binary, usually compressed versions of VCF files. VCF format was developed to represent variation data from the 1000 human genome project and is likely to be accepted as a standard format for this type of data.

We will now take our NV.bam file and generate a BCF file from it which we will view in Artemis.

To do so go back to the terminal window and type on the command line be patient and wait for it to finish and return to the command prompt before continuing:

```
samtools mpileup -DSugBf L2_cat.fasta NV.bam > NV_temp.bcf
```



```
Terminal
manager@pathogens-vm:~/Module_4_Mapping$ samtools mpileup -DSugBf L2_cat.fasta NV.bam > NV_temp.bcf
[fai_load] build FASTA index.
[mpileup] 1 samples in 1 input files
[mpileup] Set max per-file depth to 8000
```

There are two more steps required before we can view out SNPs in Artemis. First, do the actual SNP calling:

```
bcftools view -bcg NV_temp.bcf > NV.bcf
```

Second, as before we have to index the file before viewing in in Artemis:

```
bcftools index NV.bcf
```

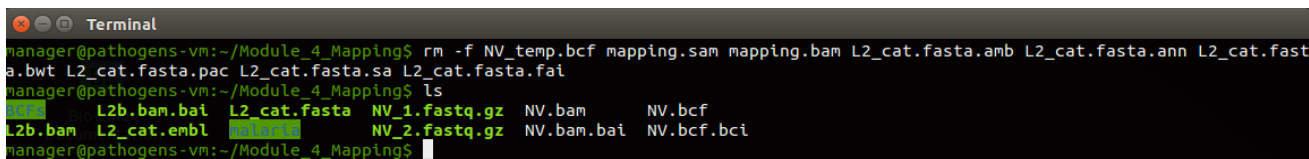
Now let's do a bit of house keeping because many of the files we have created are large and are no longer needed, before we view our SNP calls in the Artemis session that's still open. So please delete the following files:

```
NV_temp.bcf mapping.sam mapping.bam L2_cat.fasta.amb
L2_cat.fasta.ann L2_cat.fasta.bwt L2_cat.fasta.pac
L2_cat.fasta.sa L2_cat.fasta.fai
```

You can do this either in your terminal window with UNIX command rm (see below):

```
rm files
```

OR you can use the more conventional file manager if you prefer.



```
Terminal
manager@pathogens-vm:~/Module_4_Mapping$ rm -f NV_temp.bcf mapping.sam mapping.bam L2_cat.fasta.amb L2_cat.fasta.ann L2_cat.fasta.bwt L2_cat.fasta.pac L2_cat.fasta.sa L2_cat.fasta.fai
manager@pathogens-vm:~/Module_4_Mapping$ ls
L2b.bam.bai  L2_cat.fasta  NV_1.fastq.gz  NV.bam  NV.bcf
L2b.bam     L2_cat.enbl  NV_2.fastq.gz  NV.bam.bai  NV.bcf.bci
manager@pathogens-vm:~/Module_4_Mapping$
```

## File format: VCF / BCF (each line: one position in alignment)

Reference sequence name	Position	REF: base call in reference ALT: alternative base call in sequence data	Quality score of base call	Detailed information: DP=read depth DP4=REF,REF,ALT,ALT MQ=mapping quality	Genotype call info		
#CHROM	POS	ID	REF	ALT	QUAL	INFO	FORMAT
AM884176.1	24267	.	C	.	283	DP=159;AF1=0;AC1=0;DP4=75,84,0,0;MQ=60;FQ=-282	PL:DP:SP
0:159:0							
AM884176.1	24268	.	C	.	283	DP=159;AF1=0;AC1=0;DP4=73,84,0,0;MQ=60;FQ=-282	PL:DP:SP
0:157:0							
AM884176.1	24269	.	T	.	283	DP=156;AF1=0;AC1=0;DP4=75,81,0,0;MQ=60;FQ=-282	PL:DP:SP
0:156:0							
AM884176.1	24270	.	G	A	222	DP=157;VDB=0.1063;AF1=1;AC1=2;DP4=0,0,75,82;MQ=60;FQ=-282	GT:PL:DP:SP:GQ
1/1:255,255,0:157:0:99							

To look at a region with some interesting sequence variation, go again to the end of the sequence or to base position 1043000 using either the sliders, the GoTo menu or the 'Navigator'.  
Next read the BCF file that you have just created into Artemis by selecting menus and options as shown below.

1 Show File Manager ...

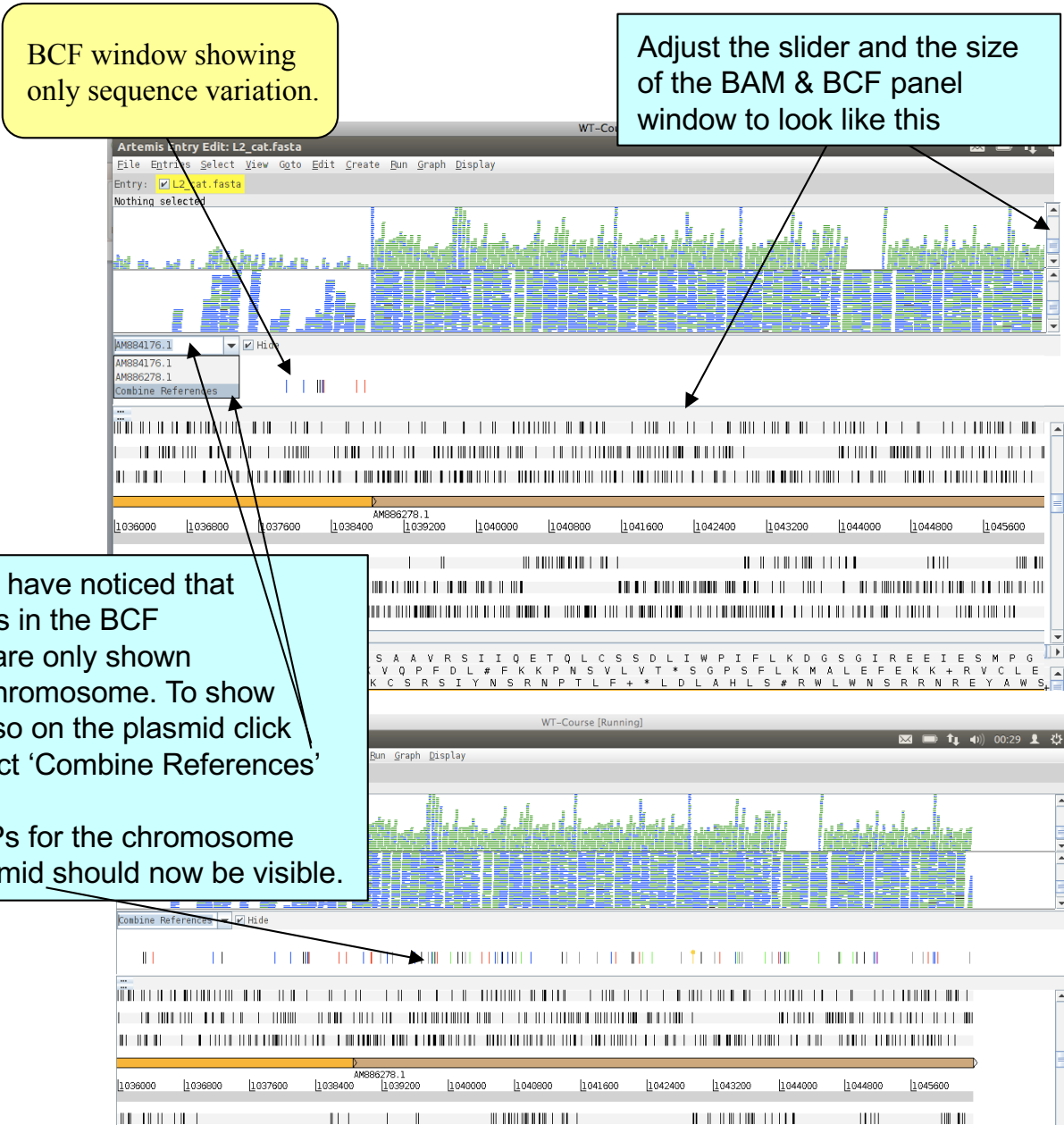
2 Read BAM / VCF ...

3 BamView :: Select Files

4 Select file NV.bcf

5 Open

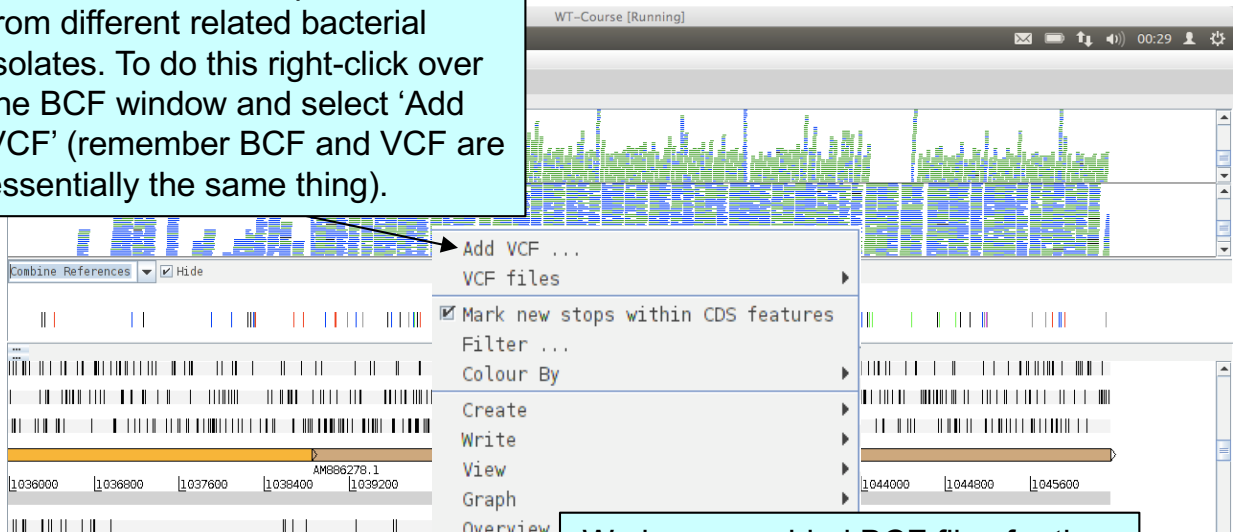




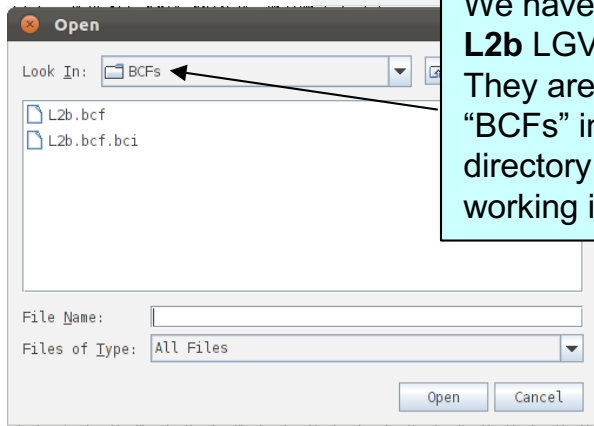
Below are the details of the three possible colour schemes for the variants in the BCF window panel (change the colour scheme via Right-click and Colour By). Note that this includes both SNPs and INDELS. Scroll along the sequence and see how many different kinds of variants you can find.

<b>1. Variant</b>	
<b>Variant A</b>	<b>Green</b>
<b>Variant G</b>	<b>Blue</b>
<b>Variant T</b>	<b>Black</b>
<b>Variant C</b>	<b>Red</b>
<b>Multiple Alleles</b>	<b>Orange, with circle at top</b>
<b>Introducing stop codon</b>	<b>Circle in the middle, colour of variant</b>
<b>Insertion</b>	<b>Magenta</b>
<b>Deletion</b>	<b>Grey</b>
<b>Non-variant</b>	<b>Light grey</b>
<b>2. Synonymous / Non-synonymous</b>	
<b>Synonymous SNP</b>	<b>Red</b>
<b>Non-synonymous SNP</b>	<b>Blue</b>
<b>3. Quality Score</b>	
<b>Variants are all on a red colour scale with those with a higher score being darker red</b>	

You can read in multiple BCF files from different related bacterial isolates. To do this right-click over the BCF window and select 'Add VCF' (remember BCF and VCF are essentially the same thing).

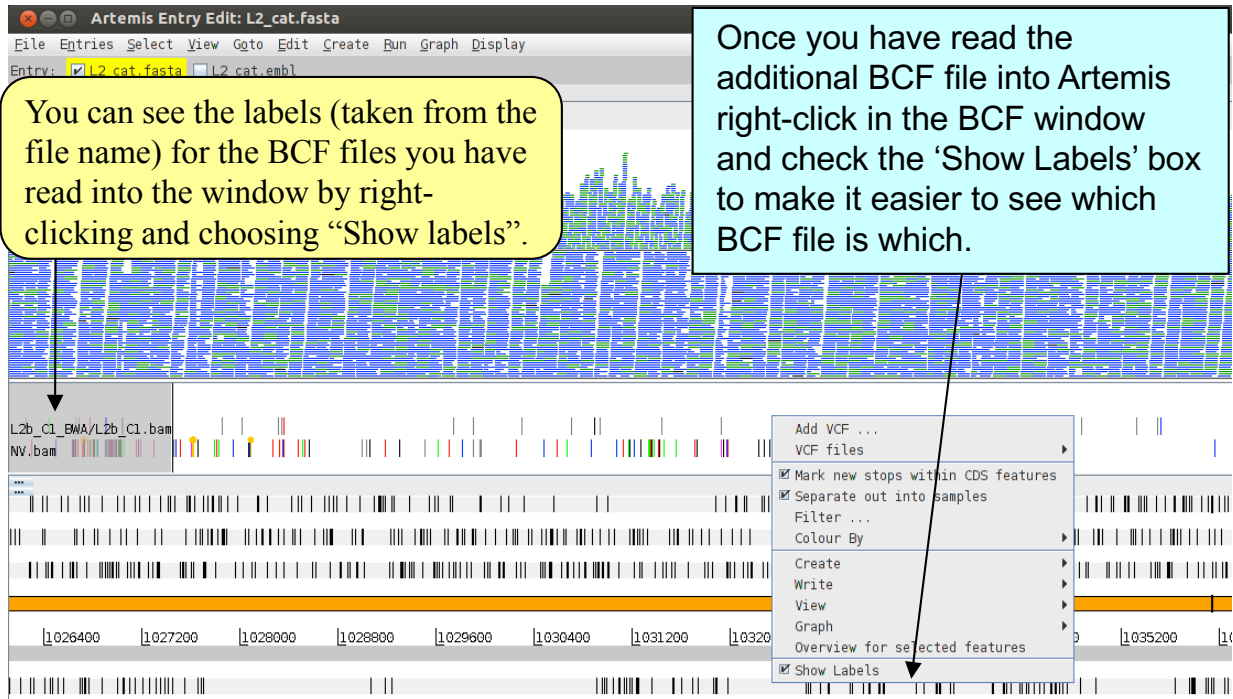


We have provided BCF files for the L2b LGV *C. trachomatis* strain. They are in the directory called "BCFs" in the Module\_2\_Mapping directory that you have been working in until now.



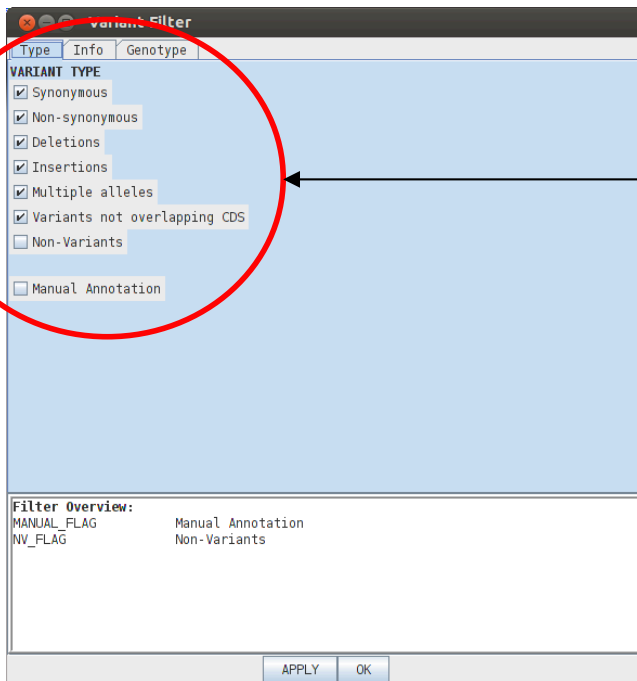
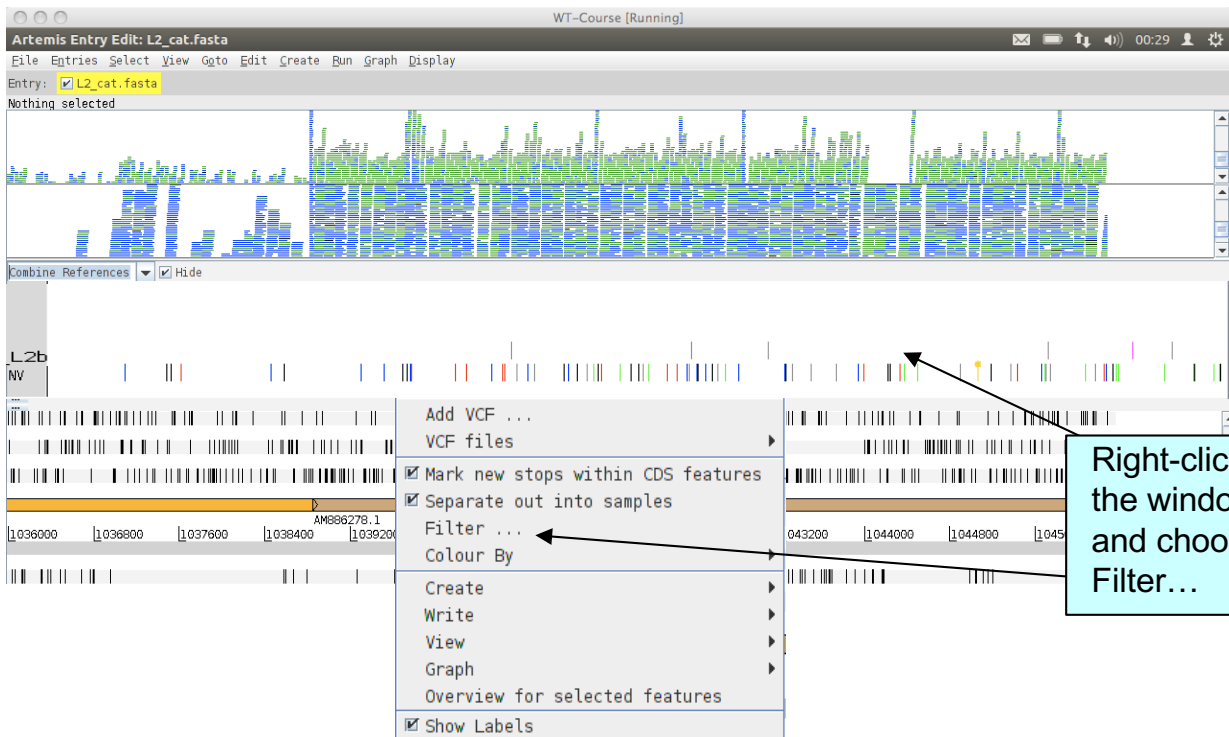
You can see the labels (taken from the file name) for the BCF files you have read into the window by right-clicking and choosing "Show labels".

Once you have read the additional BCF file into Artemis right-click in the BCF window and check the 'Show Labels' box to make it easier to see which BCF file is which.

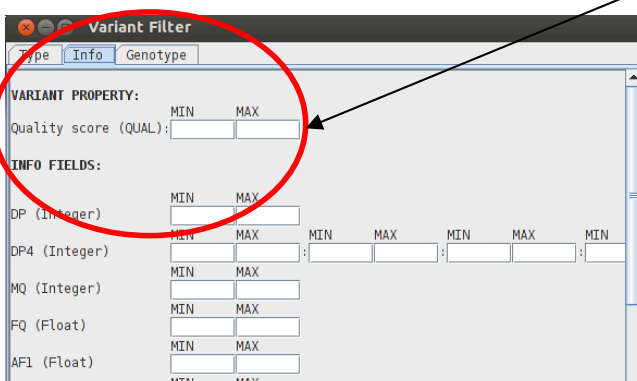


What you should notice is that L2b has far fewer SNPs and INDELs than NV compared to the reference. This is because L2b is an LGV strain of *Chlamydia* and NV is an STI strain. We will come back to these relationships later in the next Module.

As you may expect by now, Artemis also allows you to filter your VCF file.



Have a look through the variant filter window that pops up. You can select or unselect different SNP types or variants to modify your view. Non-variant sites are important because they differentiate sites where the data confirm that the sequence is the same as the reference from regions that appear not to contain SNPs simply because no reads map to them.



Like the BAM views you can also remove or include SNPs etc based on for example mapping score, depth of coverage or sequencing quality in the PROPERTY section listed under the INFO tab.

Useful cutoff values are e.g. DP of at least 10 and Qual of at least 30.

## 2. Exercise with data from *Plasmodium falciparum*

To give you a second example with exercises on how to use sequence read mapping, SNP calling and Artemis to identify relevant genomic variation, let's now turn to the data from *Plasmodium falciparum*, the eukaryotic pathogen that causes malaria in humans.

In the terminal, switch to the folder called 'malaria' using the Unix command 'cd':  
`cd malaria`

### Running a Bash script to do the work for us...

Mapping and aligning raw reads to a reference sequence is a common task in bioinformatics. To save time and to show you one example of how scripts can automate tasks we will use a **Bash script** to perform the following key tasks:

- map sequence reads from the malaria parasite strain IT to the reference sequence (3D7) using the BWA program
- call SNPs for the IT sequence data in comparison to the reference using the mpileup component of SAMtools

This shell script is very generic, and thus can be used over and over again to map different samples (also known as lanes) of sequence data.

To actually run the script, type the following on the command line:

```
./map_lanes.sh IT.Chr5_1.fastq.gz IT.Chr5_2.fastq.gz
Pf3D7_05.fasta BWA.IT.Chr5
```

Note that **BWA.IT.Chr5** still belongs to your command line! Please also note that this script will run for several minutes, so please be patient. Lots of information about the progress of the mapping will be printed to the screen, but its rare you'd ever need to look at it.

The commands performed by the script are listed on the next page. While the script is running, we can have a look at the commands, and the BASH structure. If you are not sure about certain commands, have a look back at the previous parts of this module or ask a course demonstrator or a class mate.

```
1
2 #!/bin/bash
3
4 #read in values from command line
5 fastq1=$1
6 fastq2=$2
7 ref=$3
8 output=$4
9
10 #index the reference file
11 bwa index $ref
12
13 #map the sequence data
14 bwa mem $ref $fastq1 $fastq2 > $output.sam
15
16 #create a quality filtered, sorted and indexed bam file
17 samtools view -q 15 -b -S $output.sam > $output.tmp.bam
18 samtools sort $output.tmp.bam $output
19 samtools index $output.bam
20
21 #generate a BCF file and index it
22 samtools mpileup -ugf $ref $output.bam > $output.tmp.bcf
23 bcftools view -bcvg $output.tmp.bcf > $output.bcf
24 bcftools index $output.bcf
25
26 #clean up your directory of temporary files
27 rm -f $output.tmp.bcf $output.sam $output.tmp.bam
28
29 #clean up your directory of unnecessary files
30 rm -f $ref.amb $ref.ann $ref.bwt $ref.pac $ref.sa $ref.fai
31
```

- **Line 1** tells the computer which program to use to execute or interpret this file, in this case it is the **bash** program.
- Empty lines have been inserted for clearer structure and are not interpreted. Lines starting with a **#** are comments and are not executed either.
- **Lines 4-7** read in the values passed to the script from the command line. These values are called command line arguments and will be discussed in more detail later.
- **Line 10** indexes the reference file.
- **Lines 13** aligns the fastq reads to the reference genome and outputs a sam file.
- **Line 16** filters the mapped reads and converts the .sam file into a .bam file.
- **Lines 17-18** sort and index the .bam file so that it can be viewed in Artemis.
- **Lines 21-23** generate a .bcf file and index it.
- **Lines 26 and 29** remove temporary and unnecessary files.

### Variables

In bash scripting, as in any scripting language, you use containers called variables to store data, change it, and access it later. New variables can be created like this:

```
name=value
```

In a bash script, you must do it exactly like this, with no spaces on either side of the equals sign, the variable name must contain only alphanumeric characters and underscores, and it cannot start with a numeric character. Accessing the values stored in a variable can be done like this:

```
$name
```

In the `map_lanes.sh` script we create four different variables and use them to store the values that are passed to the script from the command line.

```
fastq1=$1
fastq2=$2
ref=$3
output=$4
```

Later in the script we access the values stored in these variables. For example, we index the reference genome by passing the value that is stored in the `ref` variable to the `bwa index` command.

```
bwa index $ref
```

### Command Line Arguments

Since we want to use the `map_lanes.sh` script on different datasets, it takes some arguments on the command line telling it what to work on. These arguments are:

- Name of the input fastq files
- Name of the reference file to use
- A prefix to use when writing output files (e.g. `<prefix>.bam`).

Remember we have run the `map_lanes.sh` script with the following command line arguments

```
./map_lanes.sh IT.Chr5_1.fastq.gz IT.Chr5_2.fastq.gz
                Pf3D7_05.fasta BWA.IT.Chr5
```

A shell script can have any number of command line arguments which can be accessed in the script using the variables `$0`, `$1`, `$2`, `$3`, `$4`, `$5` etc.

- The variable `$0` is the script's name, when run with the command above this variable will contain the value `./map_lanes.sh`
- The variable `$1` is the first argument passed to the script, when run with the command above this variable will contain the value `"IT.Chr5_1.fastq.gz"`
- Similarly, the variable `$2` is the second argument and will contain the value `"IT.Chr5_2.fastq.gz"`
- `$3` is the third argument and will contain the value `"Pf3D7_05.fasta"`
- `$4` is the fourth argument and will contain the value `"BWA.IT.Chr5"`
- The total number of arguments is stored in `$#`.

When the `map_lanes.sh` script is finished running, type `ls` to see the contents of the directory. You should see a new file called `BWA.IT.Chr5.bam` which contains the results of mapping the files `IT.Chr5_1.fastq` and `IT.Chr5_2.fastq` to the `Pf3D7_05.fasta` reference sequence.

Why is the file called `BWA.IT.Chr5.bam`?



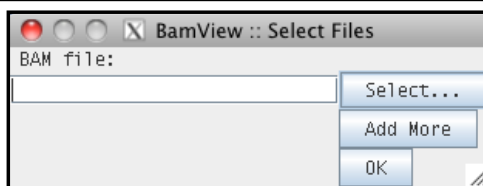
Now back to biology. It is thought that a duplication in the *mdr1* gene of *P. falciparum* is associated with drug resistance against the antimalarial mefloquine and that it may also modulate susceptibility to chloroquine, another antimalarial drug. For more information have a look in PubMed, e.g. at Borges et al. (2011) [PMID: 21709099] or at Mungthin et al. (2010) [PMID: 20449753]!

Please start up artemis using the following command:

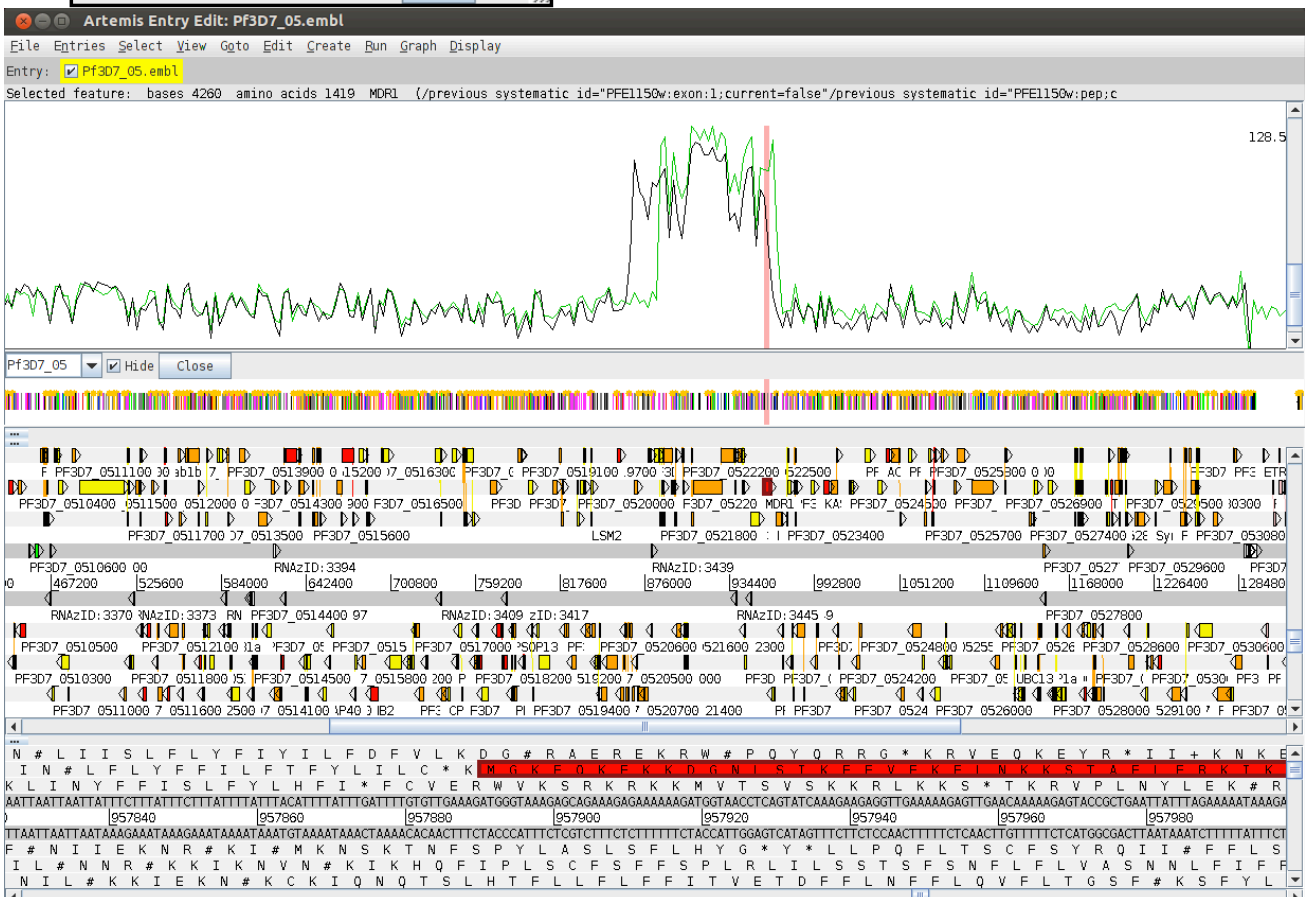
```
art -Dbam=BWA.IT.Chr5.bam,BWA.IT.Chr5.bcf Pf3D7_05.embl &
```

Once Artemis has started running on your screen **navigate to the *mdr1* gene locus** using e.g. the Navigator (Goto – Navigator... – Goto Feature With Gene Name). What can you say about the read coverage at this locus? (you may have to zoom out to get a good look at the whole region which is between 866,000 and 965,000bp).

So far you looked at the **IT strain**. What about the *mdr1* locus in the **Dd2 strain** of the malaria parasite? The Dd2 clone is known to be chloroquine resistant. Add its mapped reads (a file already prepared before the course) by right-clicking on the BAMview and choosing 'Add BAM...'. Select the file **DD2.Chr5.bam**



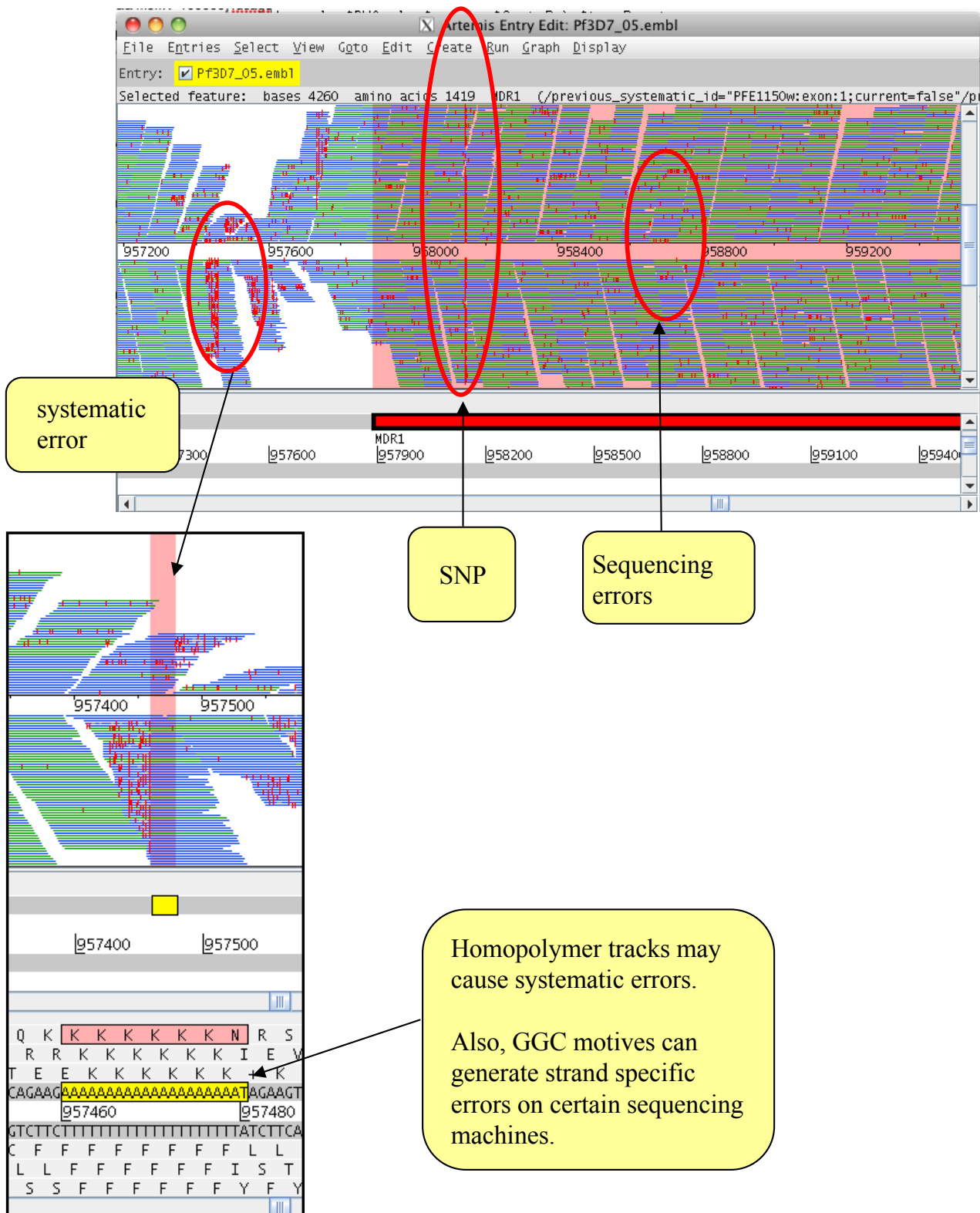
Select DD2.Chr5.bam



It looks like that both the IT and the Dd2 clone have a copy number variation (assuming the reference was assembled correctly). Is the duplication the same in both clones?

## SNPs

Let's have a look at SNPs now. To do so, zoom in on the *mdr1* gene and make sure you are in Strand Stack view and have Show SNP marks selected. As mentioned before, in addition to true SNPs some differences between the reference sequence and the mapped reads are due to sequencing errors. On average, 1 in every 100 bases in the reads is expected to be incorrect. In particular, some sequencing errors may be due to a systematic problem as illustrated below.



When you zoom in on position 958145 as far as you can go, you can see a SNP: here, both strain IT and Dd2 have a thymine where the reference (3D7) has an adenine.

What is the consequence of this SNP? Can we tell what effect it will have for the clones? Is this the mutation N86Y (or also simply referred to as Y86) that may modulate the degree of resistance to chloroquine? (See e.g. Mula et al. (2011) [PMID: 21810256])

Reference

Mapping reads

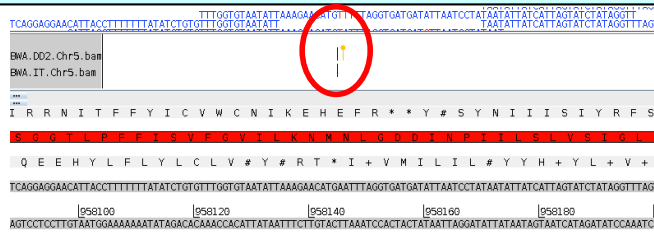
To answer the question mentioned just above, right-click on the gene annotation of the *mdr1* gene and then choose View – Amino Acids Of Selection: here you can see which amino acids are normally coded for around amino acid position 86. Note also that AAT codes for Asn (N) and TAT codes for Tyr (Y).

Now have a look at neighbouring position 958146. What could this be? Is it from IT or DD2? To answer this question clone the window and display the reads of only one or the other parasite strain in each window, just as we did earlier in this module.

This BAM window shows the IT reads

Is there another SNP in Dd2?

Now also open the BCF file for the Dd2 clone to the window by right-clicking on the VCF/BCF pane of the window, choosing Add VCF..., and selecting file "DD2.Chr5.bcf".

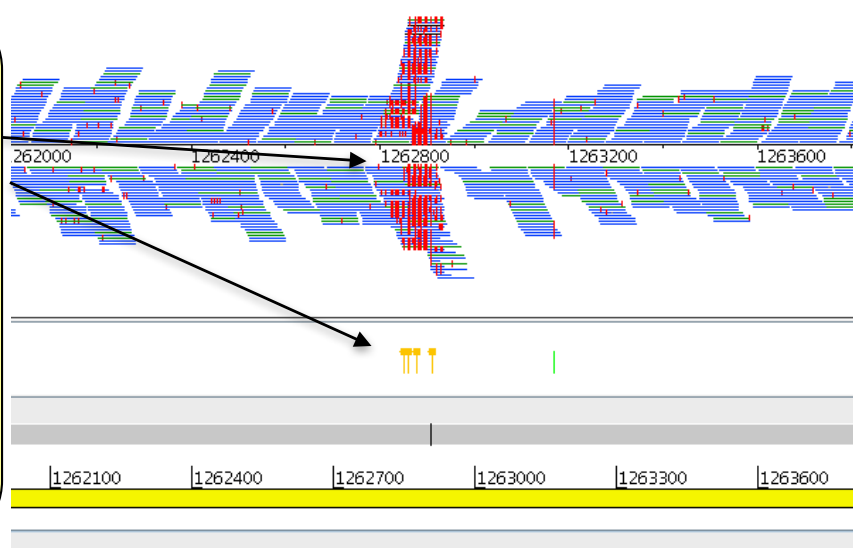


Note that the malaria parasite is a haploid organism. Yet the position above like many other in the genome have apparently more than one allele. What might be the explanation for this?

If you like, explore the genome a bit further, maybe you find some other surprising SNP calls and interesting genome variants!

Here is another example of apparently "heterozygous" SNPs in this haploid organism. Note also that the read coverage in this region is also unusually high.

A region like this would be best resolved by *de novo* sequence assembly.



**Extra exercise** for those who are interested in drug resistance in *P. falciparum*: Can you find the *pfcr* mutation associated with chloroquine resistance? It is on chromosome 7 at amino acid position 76 of the gene (K76T) – see e.g. Djimde et al (2001) [PMID: 11172152]. For this exercise you can start Artemis from the command line like this:

**art -Dbam=DD2.Chr7.bam,DD2.Chr7.bcf Pf3D7\_07.embl**

The screenshot shows the Artemis software interface with the following details:
 

- Window title: Artemis Entry Edit: Pf3D7\_07.embl
- Menu: File Entries Select View Goto Edit Create Run Graph Display
- Entry:  Pf3D7\_07.embl
- One selected base on forward strand: 404931
- Gene structure: Pf3D7\_07 with exons and introns.
- Coordinates: `02900 403200 403500 403800 404100 404400 404700 405000`

Note that AAA codes for Lys (K) while ACA codes for Thr (T).

**Extra exercises for bash scripting****Trouble-shooting – error checking**

Try running the `map_lanes.sh` script with the following command line arguments:

```
./map_lanes.sh IT.Chr5_1.fastq.gz IT.Chr5_2.fastq.gz
                Pf3D7.fasta IT.Chr5
```

Did the script run successfully? If not, why not?

Often, the difference between a good script and a poor script is assessed in terms of the robustness of the script. That is, the ability of the script to handle situations in which something goes wrong. In this case, does the `map_lanes.sh` script handle the situation where a file supplied by the user does not exist?

In this example we will look at improving the robustness of the `map_lanes.sh` script by adding some argument and error checking to the script. Using your preferred text editor open the file `map_lanes_validate_inputs.sh`. You should see the shell script shown on the next page.

Note that apart from error checking, this script also only performs the mapping, it does not generate the bcf files, and it does not clean up after itself!

This script performs some checks on the values passed to it from the command line and then performs a set of standard mapping tasks:

**Lines 1-7** tell the computer which program to use to execute this file and reads in the values passed to the script from the command line.

**Lines 10-13** checks that the correct number of command line arguments have been passed to the script. The lines say if the number of command line arguments passed to the script is NOT EQUAL TO 4, print a message to the screen telling the user what the correct usage is and exit the script.

**Lines 16-19** checks that all the files passed to the script exist. The lines say if the first fastq file does not exist OR the second fastq file does not exist OR the reference file does not exist, print an error message to the screen and exit the script.

**Lines 22-30** perform a set of standard mapping tasks.

**Please note:**

**\$#** is the number of command line arguments

**!=** means NOT EQUAL TO

**\$0** is the name of the script

**!** is the NOT operator

**-f** checks if a file exists

**||** means OR

```
1  #!/bin/bash
2
3  #read in values from command line
4  fastq1=$1
5  fastq2=$2
6  ref=$3
7  output=$4
8
9  #check the correct number of parameters have been passed to
  the script
10 if [ $# != 4 ]; then
11     echo "Usage: `basename $0` fastq1 fastq2 reference_file
  output_prefix"
12     exit
13 fi
14
15 #check the fastq and reference files passed to the script
16 exist
17 if [ ! -f $fastq1 ] || [ ! -f $fastq2 ] || [ ! -f $ref ]; then
18     echo "Error: One of the input files does not exist"
19     exit
20 fi
21
22 #index the reference file
23 bwa index $ref
24
25 #map the sequence data
26 bwa mem $ref $fastq1 $fastq2 > $output.sam
27
28 #create read quality filtered sorted and indexed bam file
29 samtools view -q 15 -b -S $output.sam > $output.tmp.bam
30 samtools sort $output.tmp.bam $output
  samtools index $output.bam
```

If you want to you could modify the script to check to see if the output file already exists. If it does exist, print a warning message and exit from the script.



### Decision Statements

Sometimes you will want to perform different tasks depending on whether a condition is true or false. In bash this can be achieved with the keyword, `if`. The `if` statement consists of a condition that is evaluated, and a block of code that is run if the condition evaluates to true.

```
if [ CONDITION ]; then
# instructions to follow if condition is true
fi
```

### Loops

In bioinformatics we often have to perform the same action/analysis multiple times. For example, its quite common to multiplex a 96 well plate of samples into a single Illumina lane, so to analyze your data you'll need to run the same commands on all 96 sets of sequencing data. Rather than

running a script over and over again, you can use a loop. It will keep running a set of commands until a condition is met, for example, loop over all files in a directory and run the commands on each file.

In bash this can be achieved with the keyword `FOR`. The `FOR` statement consists of a list and a variable name, then a block of commands to run. In the example below, the `ls` command is run to get a list of files in the current directory. Each file is then taken in turn and is assigned to

the variable `i`. The block of code is then run, and `$i` contains the name of the file. Here we just print out the filename, but you can use any command.

```
FOR i in $( ls ); DO
echo $i
DONE
```

**End of module...**

**ANY QUESTIONS?**

**Please feel free to ask at any time!**

Please close down Artemis, ready to start the next module.



# Module 5

# Phylogenetics

## Introduction

Phylogeny has important applications in many fields of genome biology. For example, when annotating a gene in a new genome it is useful for identifying previously-annotated genes in other genomes that share a common ancestry. It is also becoming increasingly common to use phylogeny to trace the evolution and spread of bacterial diseases, and even as an epidemiological tool to help identify disease outbreaks in a clinical setting. Further analysis of genome sequences to examine recombination, molecular adaptation and the evolution of gene function, all benefit from phylogeny.

Phylogenetics is essentially about similarity, and looking at patterns of similarity between taxa to infer their relationships. Although the methods may initially appear quite daunting, the basic ideas behind phylogenetics are quite simple. We want to identify the tree that best fits our data assuming that the data evolve under a simple model. When the data are DNA, we can use our knowledge of biology to improve our chances of finding the correct tree by defining evolutionary models that make biological sense. For example, we know that transition mutations occur more frequently than transversions, so we can make our model favour trees in which this is the case with our data.

In this module we will use phylogenetics to explore the strengths and weaknesses of techniques that may be used for typing *Chlamydia trachomatis*. We will reconstruct phylogenetic trees in a Maximum Likelihood framework, using the PhyML program, since this allows the use of complex (more realistic) models of nucleotide (or amino acid) substitution to be applied, and it produces a statistical measure of tree quality (‘likelihood’) that can be directly compared between hypotheses. By the end of this module you should have some understanding of:

1. Multiple sequence alignment using ClustalX and/or muscle.
2. Maximum likelihood phylogenetic estimation using a nucleotide model.
3. Evaluation of support for relationships in a tree using non-parametric bootstrap replicates.
4. Interpretation of phylogenetic trees.

## Exercise: Assessing typing tools for *C. trachomatis*

In this module we will use phylogenetic techniques to assess the utility of some of the molecular typing tools available for *C. trachomatis*.

The exercise will begin by using assembled sequences. Historically this is the type of data that would have been available for molecular phylogenetic analyses. We will look at the *ompA* gene, the most commonly used *Chlamydia* typing locus.

In the second part of the exercise, we will see how Artemis can be used to create sequence alignments from the *C. trachomatis* bcf files used in the mapping module. We will extract the sequences of genes used in a *Chlamydia* multilocus-typing scheme and compare the results with those from the *ompA* analysis.

Finally we will create a tree based on whole-genome SNP data and see which of the other two typing schemes it supports.

### i) *ompA* phylogeny from gene sequences

Later in the week you will learn how short-read sequence reads, such as those produced by the Illumina platform, can be assembled into draft genomes. Assembly of short read data is similar to the methods historically used to create high quality reference genomes from capillary (Sanger) sequence data. Mapping and assembly both have their advantages and disadvantages, which should become apparent during this course.

For our first phylogeny we will make a tree using the sequences of the *ompA* gene from 16 strains of *C. trachomatis*. These sequences are provided for you in a single file called *ompA\_assembled.mfa* in the Module 5 directory.

Historically, the most commonly used tool for typing *C. trachomatis* isolates was serotyping using the MOMP (major outer membrane protein), which is encoded by the *ompA* gene. There are two biovars of *C. trachomatis*: 1) the trachoma biovar includes ocular and urogenital strains, which cause the majority of trachoma and STIs, and are characterised by localised infections of the epithelial surface of the conjunctiva or genital mucosa; 2) the lymphogranuloma venereum (LGV) biovar includes strains which are distinguished by their ability to spread systemically thorough the lymphatic system, causing genital ulceration and bubonic disease. Based on MOMP serotyping, *C. trachomatis* has been subdivided into between 15 and 19 serotypes: the trachoma biovar includes ocular serotypes A to C and urogenital serotypes D to K, while the LGV biovar includes serotypes L1, L2 (including L2a, b and c) and L3.

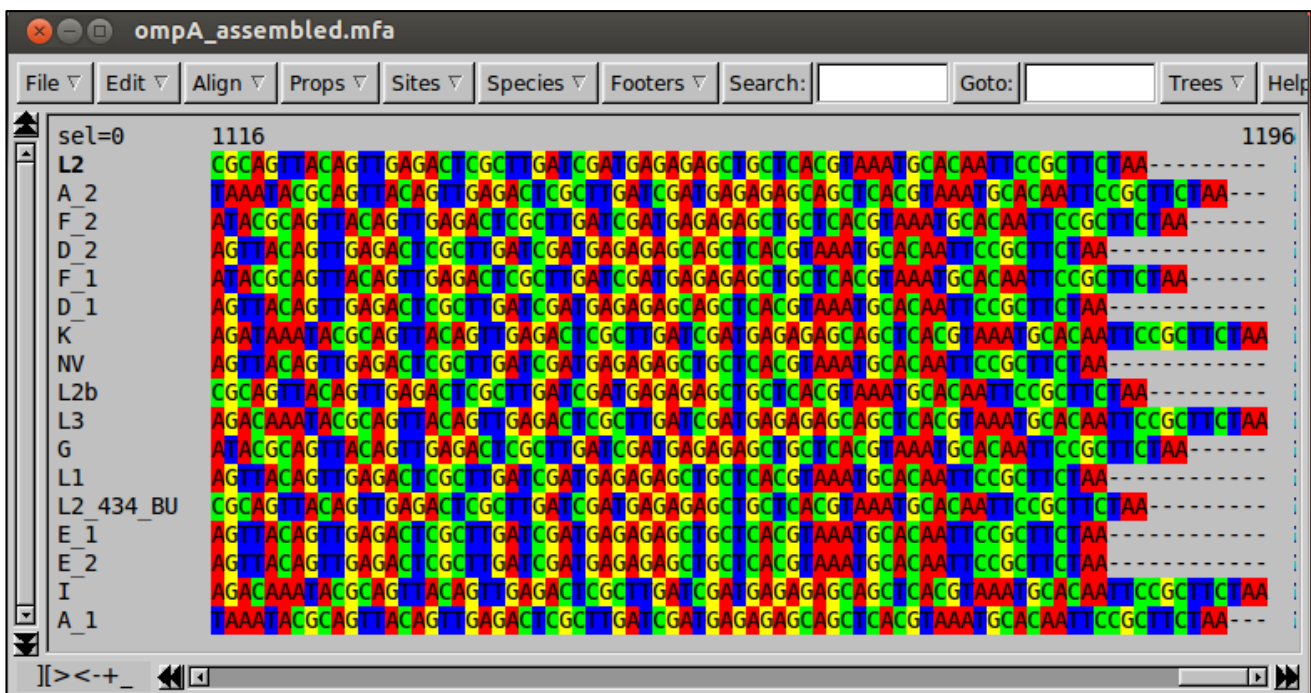
## Viewing the alignment in Seaview

To view and edit alignments and produce phylogenies we will use a program called **Seaview**. Seaview is a graphical user interface (GUI) that combines a number of the most popular alignment and phylogeny programs. We have chosen to use Seaview in this module as it is freely-available for all platforms, simple to use, and contains all of the tools necessary for automatic and manual multiple sequence alignment and producing phylogenies. However, the primary aim of this module is to demonstrate the principles of alignment and phylogenetic reconstruction that apply to any of the wide range of programs that you may encounter.

First you should navigate to Module 5 directory.

Start Seaview by typing the command '**seaview**' in the terminal window, and load the alignment file `ompA_assembled.mfa` by selecting '**Open**' from the '**File**' menu.

The DNA sequences of the *ompA* gene from each of the *C. trachomatis* strains will be shown in the Seaview window. Note that the sequences are not all of the same length, so although the beginning aligns well, if you scroll along you will find that they start to go out of alignment.



If you have time, have a look at the mapping of the *ompA* gene in some of your samples. The diverse nature of the *ompA* region, with high SNP density and a number of indels, makes mapping of the region difficult and variant-calling from mapped data prone to errors. It is for this reason that we have provided assembled versions of the gene for this part of the module. The methods used to produce these assemblies were exactly as you will see in the assembly module later in the course.

## Multiple sequence alignment

Before any phylogenetic analysis, we must make sure that the columns in our data represent homologous bases. With gene or protein sequence data, this usually means aligning the nucleotide or amino acid sequences using a multiple alignment program. In this part of the exercise you will align the *ompA* sequences. Length differences complicate multiple sequence alignment because these require the insertion of gaps into an alignment to ensure that homologous sites remain aligned. Alignment should be checked by eye wherever possible, because no program is perfect; but for large numbers of sequences and sites, de novo manual alignment is impractical and we must rely on the algorithms of sequence alignment tools.

Seaview allows alignment using two programs, **clustal** and **muscle**. Generally muscle is faster, and the protein alignments are of similar quality to clustal. In both cases, sequences are aligned by assigning costs to particular base changes and gap insertions and then minimising the overall cost of the alignment. The parameters can be altered to optimise the process, but in practice the default options will usually suffice. It is usually better to align genes after translating them into amino acids, so we will do that here.

On the 'Props' menu choose 'View as proteins'.

```

ompA_assembled.mfa
File Edit Align Props Sites Species Footers Search: Goto: Trees Help
sel=0 320 400
L2 LNPTIAGAGDVKASAEGLGDTMQIVSLQLNKMKSRRKSCGIAVGTTIVDADKYAVTVETRLIDERAHVNAQFRF*---
A_2 TTLNPTIAGKGTVVSSAENELADTMQIVSLQLNKMKSRRKSCGIAVGTTIVDADKYAVTVETRLIDERAHVNAQFRF*-
F_2 LNPTIAGCGSVAGANTEGQISDTMQIVSLQLNKMKSRRKSCGIAVGTTIVDADKYAVTVETRLIDERAHVNAQFRF*-
D_2 NPTIAGAGDVKTGAEGQLGDTMQIVSLQLNKMKSRRKSCGIAVGTTIVDADKYAVTVETRLIDERAHVNAQFRF*---
F_1 LNPTIAGCGSVAGANTEGQISDTMQIVSLQLNKMKSRRKSCGIAVGTTIVDADKYAVTVETRLIDERAHVNAQFRF*-
D_1 NPTIAGAGDVKTGAEGQLGDTMQIVSLQLNKMKSRRKSCGIAVGTTIVDADKYAVTVETRLIDERAHVNAQFRF*---
K TTLNPTIAGKGA VVSSGSDNELADTMQIVSLQLNKLKSRKSCGIAVGTTIVDADKYAVTVETRLIDERAHVNAQFRF*
NV NPTIAGAGDVKASAEGLGDTMQIVSLQLNKMKSRRKSCGIAVGTTIVDADKYAVTVETRLIDERAHVNAQFRF*---
L2b LNPTIAGAGDVKASAEGLGDTMQIVSLQLNKMKSRRKSCGIAVGTTIVDADKYAVTVETRLIDERAHVNAQFRF*---
L3 TTLNPTIAGKGSVVASGSENELADTMQIVSLQLNKMKSRRKSCGIAVGTTIVDADKYAVTVETRLIDERAHVNAQFRF*
G LNPTIAGCGSVVAANTEGQISDTMQIVSLQLNKMKSRRKSCGIAVGTTIVDADKYAVTVETRLIDERAHVNAQFRF*-
L1 NPTIAGAGEVKANAEGQLGDTMQIVSLQLNKMKSRRKSCGIAVGTTIVDADKYAVTVETRLIDERAHVNAQFRF*---
L2_434_BU LNPTIAGAGDVKASAEGLGDTMQIVSLQLNKMKSRRKSCGIAVGTTIVDADKYAVTVETRLIDERAHVNAQFRF*-
E_1 NPTIAGAGDVKASAEGLGDTMQIVSLQLNKMKSRRKSCGIAVGTTIVDADKYAVTVETRLIDERAHVNAQFRF*---
E_2 NPTIAGAGDVKASAEGLGDTMQIVSLQLNKMKSRRKSCGIAVGTTIVDADKYAVTVETRLIDERAHVNAQFRF*---
I TTLNPTIAGKGTVVASGSDNDLADTMQIVSLQLNKMKSRRKSCGIAVGTTIVDADKYAVTVETRLIDERAHVNAQFRF*
A_1 TTLNPTIAGKGTVVSSAENELADTMQIVSLQLNKMKSRRKSCGIAVGTTIVDADKYAVTVETRLIDERAHVNAQFRF*-
  
```

To start the alignment, select 'Align' then 'Align all'.

When the alignment process is complete, Seaview will have inserted gaps into the sequences so that homologous sites (or at least homologous according to the alignment program) are lined up in columns.

The screenshot shows the Seaview interface with the file `ompA_assembled.mfa` open. The alignment is displayed in a grid format. The top bar shows menu options: File, Edit, Align, Props, Sites, Species, Footers, Search, Goto, Trees, Help. The main window shows a list of sequences on the left (L2, A\_2, F\_2, D\_2, F\_1, D\_1, K, NV, L, L, G, L1, L2\_434\_BU, E\_1, E\_2, I, A\_1) and their corresponding amino acid sequences. The sequences are color-coded by amino acid type. A yellow callout box points to a gap in the sequence and says "Inserted gaps". Another yellow callout box points to a column and says "Each column is now a hypothesis of homology".

If you inspect the alignment, it should be clearer how the sequences differ from one another. Can you see which sequences are most closely related?

If an alignment has been problematic, requiring many gaps, it is advisable to inspect the it by eye and edit where necessary. In Seaview you can add gaps with the space bar, and remove them with the backspace (for more detailed instructions see the Seaview 'Help'). If you are not convinced by any region of the alignment it is probably better to remove it, as erroneous alignment provides misleading information for phylogeny reconstruction.

If you turn off protein view you can see that the nucleotides are also now aligned.

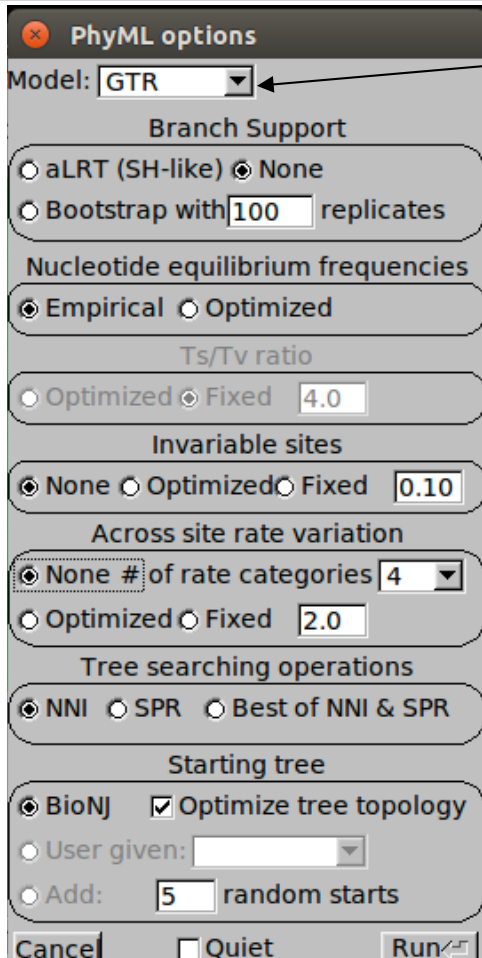
The screenshot shows the Seaview interface with the file `ompA_assembled.mfa` open. The alignment is displayed in a grid format. The top bar shows menu options: File, Edit, Align, Props, Sites, Species, Footers, Search, Goto, Trees, Help. The main window shows a list of sequences on the left (L2, A\_2, F\_2, D\_2, F\_1, D\_1, K, NV, L2b, L3, G, L1, L2\_434\_BU, E\_1, E\_2, I, A\_1) and their corresponding nucleotide sequences. The sequences are color-coded by nucleotide type. The alignment is still visible in a grid format.



# Phylogeny estimation using PhyML

To estimate the phylogeny, we will use a program called **PhyML**, which is included in Seaview. PhyML uses maximum likelihood (ML) to estimate the tree. We will use ML because it is more accurate than simpler methods as it specifies an explicit evolutionary model to account for sources of homoplasy, but at the same time it is relatively fast. Furthermore, it uses an optimality criterion (likelihood), which means it is possible to compare different trees directly.

Make sure you are in nucleotide view, then choose 'Trees' then 'PhyML'



Choose the 'GTR' model.

In the 'Branch support' box select 'None'. We will look at branch support later

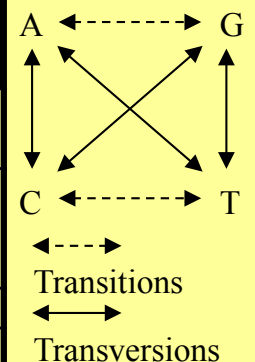
If you choose a model which includes different transition and transversion substitution rates you can set the ratio here or ask to optimise it from the data

In the 'Across site rate variation' box select 'None'. We will look at rate variation later

The last two boxes are for specifying how the program will search the tree, and how it will choose a tree to start from. We will use the defaults

PhyML includes a number of nucleotide substitution models. The strength of using a Maximum Likelihood method is that an explicit model of nucleotide substitution is applied, which can be more biologically realistic than some other methods. Various models are available from the very simple (and unrealistic) to the quite complex:

Model	# free Params	Base frequencies	Substitution rates
JC69	0	All equal	All equal
F81	3	All free	
K2P	1	All equal	Transitions and transversions different
HKY	4	All free	
F86	4		
TN93	5		Two types of transition and transversion
GTR	8		All free



Increasing complexity

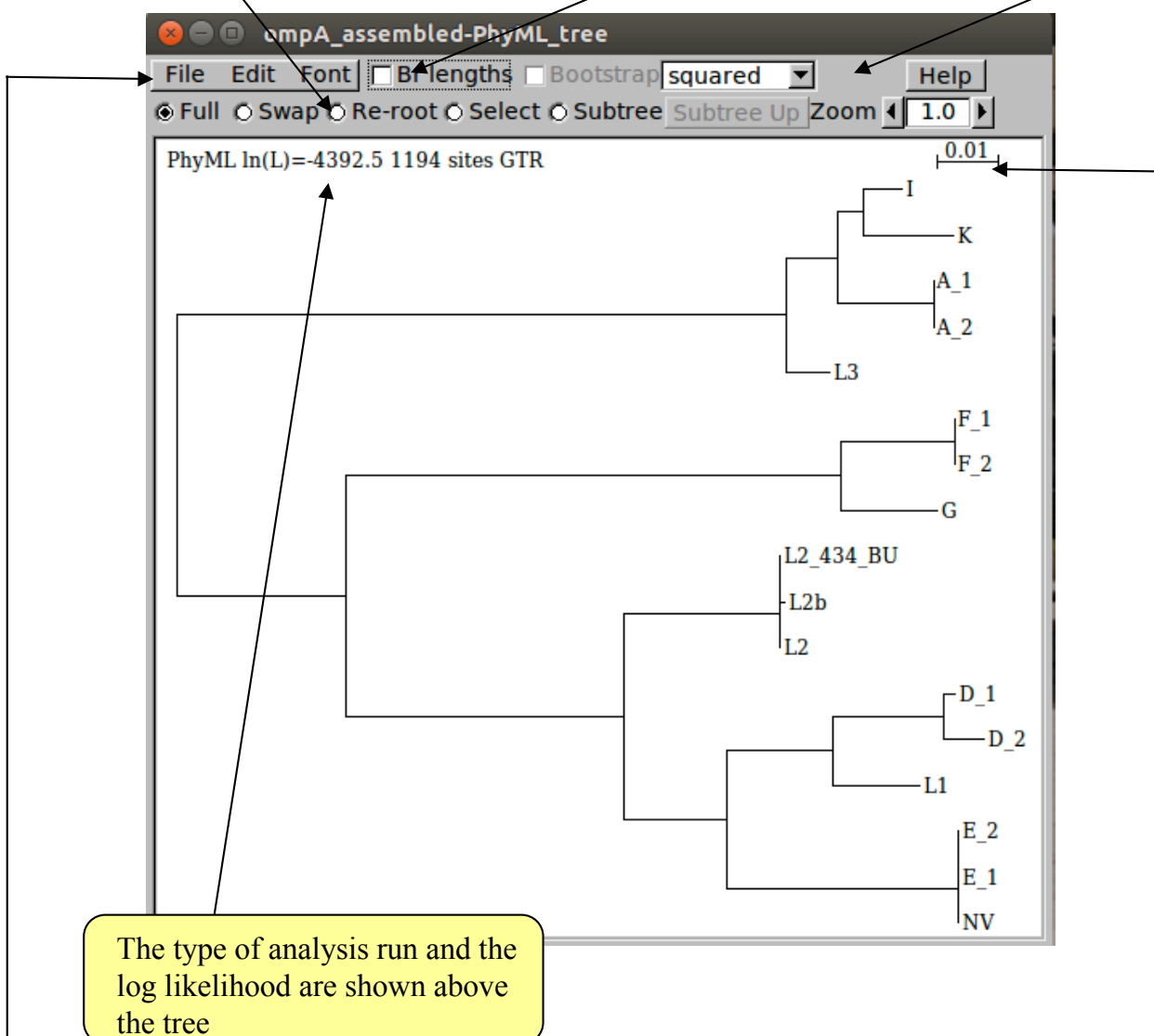
## Interpretation of phylogeny

Once the run has finished, click 'OK'. The tree created by PhyML not only includes the **topology** of tree (i.e., the relationships between sequences) but also the **branch lengths** (i.e., the amount of change occurring in each lineage). Therefore, the tree is drawn as a **phylogram**, in which the length of branches is proportional to the amount of evolutionary change (shown below).

You can alter the placement of the **root** by selecting this option and clicking on a node in the phylogeny.

You can tick '**Br lengths**' to add branch length values to the tree.

You can change between a rooted and unrooted tree



You can save your tree in Newick format that can be opened in other tree viewers

The scale bar represents the number of inferred substitutions per site

If you are running behind at this point, skip now to part ii of the exercise. You can come back to this part when you have time. Otherwise, note down the likelihood of the tree. We will use it later.



## Phylogeny estimation with across site rate heterogeneity

Almost all nucleotide sequences in nature display **across site rate heterogeneity**. This means that not all sites within the sequence evolve at the same rate; rather some parts of a gene evolve faster than others, an active site of an enzyme for example, generally changes more slowly because it is functionally important. When comparing several sequences to estimate a phylogeny, we should account for rate heterogeneity to avoid errors, i.e., fast evolving, but otherwise unrelated, sequences clustering together.

PhyML options

Model: **GTR**

Branch Support

aLRT (SH-like)  None

Bootstrap with  replicates

Nucleotide equilibrium frequencies

Empirical  Optimized

Ts/Tv ratio

Optimized  Fixed

Invariable sites

None  Optimized  Fixed

Across site rate variation

None # of rate categories

Optimized  Fixed

Tree searching operations

NNI  SPR  Best of NNI & SPR

Starting tree

BioNJ  Optimize tree topology

User given:

Add:  random starts

Quiet

Now we will estimate the phylogeny again with additional parameters in the substitution model, which will correct for rate heterogeneity among sites.

In the 'Across site rate variation' box make sure 'Optimized' is selected.

This option tells the program that the sites may evolve at different rates. Although the method is complex, you can think of it as if the program has a number of bins (rate categories) with different rates of change from slow to fast. During the optimisation process the characters are sorted into the bin that fits their profile best.

You can specify how many rate categories the program should use. However, doubling the number of categories doubles the run time. Usually 4 to 8 suffice.

Make sure '# of rate categories' is set to 4.

**Proportion of invariable sites:** you can also include a parameter to optimise the proportion of sites in the alignment which cannot change. Most maximum likelihood models assume that all sites in an alignment can change. However, there may be sites that are essential for the correct functioning of the protein, for which substitutions cannot occur without disruption the protein function. Such sites would break the assumptions of the model. The invariable sites parameter was designed to adjust for this. However, it is often unnecessary if you have defined a model with correction for rate heterogeneity.

Select 'Optimized' from the 'Invariant sites' box to allow the proportion of invariant sites to change. After selecting these parameters, ensure that you have selected the same choice of substitution model as you did in the previous analysis. Now press 'Run' to begin the estimation process as before. Examine the tree; have the additional parameters had any effect? Again note down the likelihood of the tree. We will use it in the next section.

## Comparing models with the likelihood ratio test

The **likelihood ratio test (LRT)** can be used to statistically test the difference in fit of two nested evolutionary models to the data.

'Nested' means that the more complex model must include all of the parameters of the simpler model. For comparison of non-nested models, more complex methods are available - see the notes on model selection at the end of this module.

Increasing model parameters can only improve the fit of the data to the trees, so more complex models will always produce higher likelihood values (= smaller negative log likelihood values). However, to justify the addition of these additional parameters we want to know if they have provided a *significant* improvement to the fit to the data.

To perform a LRT we must first calculate the likelihood ratio (LR) of our two models:

$$LR = 2 \times (\text{neglogL1} - \text{neglogL2})$$

Where: neglogL1 is the negative log likelihood of the simpler model, and neglogL2 is the negative log likelihood of the more complex model.

The LRT statistic approximately follows a chi-square distribution, so we can evaluate the significance of our LR using chi-square significance tables (or calculate p-values using statistical calculators)

However, as with any chi-square significance test, we need to know the degrees of freedom (df). In a LRT the df are the difference in number of free parameters between the two models.

The number of free parameters for the models in Seaview are listed in the **table of models on page 6 of this module**. e.g. the GTR model has 8 free parameters. Both the gamma parameter of among site rate variation and the parameter for estimating the proportion of invariant sites add a single free parameter each.

To help you perform your LRT, we have provided a pre-formatted spreadsheet for calculating significance from your negative log likelihoods and the number of free parameters of the models you used.

Start **LibreOffice Calc** using the icon on the left of the screen and **open LR\_test.xls** from the phylogeny directory of Module 5. Type the negative log likelihood values you recorded for your two trees into the appropriate boxes. Type the number of free parameters for your two models into the appropriate boxes. It should look something like the image below.

	simpler model	more complex model	LR/df diff
neglogL	4392.5	4234.1	316.8
df	8	10	2
		P-value	1.6134E-069

Do the additions of the gamma and invariant sites parameters significantly improve the fit of the model to the data?

## Phylogeny estimation with bootstrapping

**Bootstrapping** is a statistical technique for adding confidence intervals around an estimate, in this case, a tree topology. Non-parametric bootstrapping involves repeated analysis of the data set through “**resampling with replacement**”.

Imagine putting each site into a bag. Replicate data sets are created by randomly drawing sites from the bag until a new dataset the same size as the original has been created. Importantly, after a site has been drawn, it is replaced back into the bag. This means some sites may be present more than once in the resampled dataset whilst others may not be present at all.

Trees are then built for each replicate data set. Robust relationships, i.e. those that are repeatable, will occur in a large proportion of randomised data sets.

Estimate a bootstrapped phylogeny for the *ompA* data set by creating a new phylogeny as before, with the addition of 10 bootstrap replicates. Click on ‘**Bootstrap**’ in the ‘**Branch Support**’ box, and enter ‘**10**’ in the replicates box. Processing of the 10 replicates may take a few minutes, so you could move on while this is running. In practice you would want to run more replicates (100 or 1000), but we are only using 10 for speed.

**Branch Support**

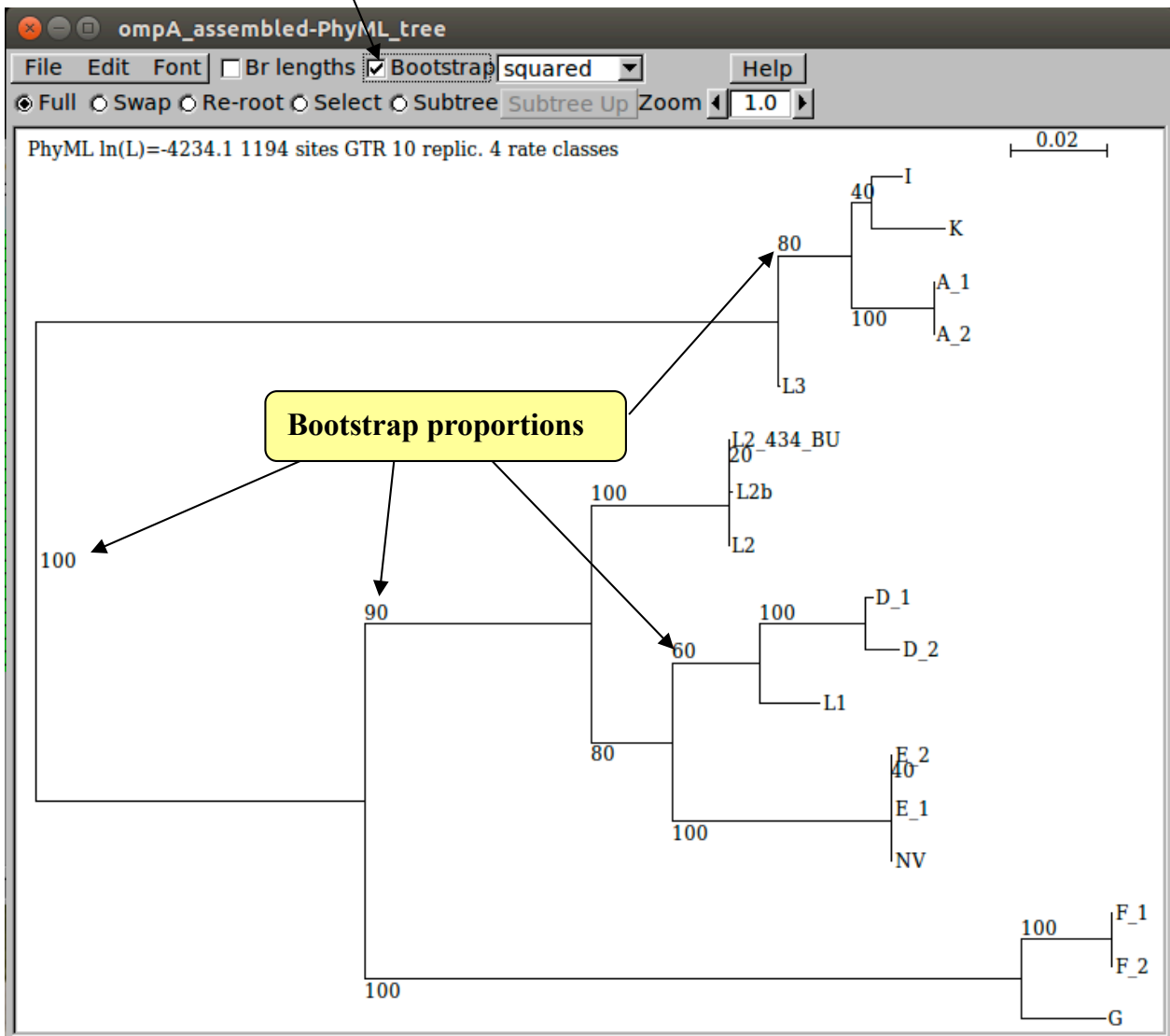
aLRT (SH-like)    None  
 Bootstrap with  replicates

**WARNING!:** bootstrap proportions are measures of robustness, or repeatability. A high bootstrap value indicates that a given node tends to occur in every analysis. **This does not guarantee that the node is correct.** For example, if the substitution model is inaccurate, it could produce the wrong answer in every estimation.

You will notice that PhyML includes another branch support method called aLRT. aLRT assesses whether each branch on the tree provides a significant likelihood improvement over the same tree with that branch collapsed. We will not use this method here or discuss it further. It is not equivalent to bootstrap, but is much faster and may be useful for assessing branch support on large phylogenies.

Once the search is complete, you can show the bootstrap values on the tree by ticking this box.

Each node in the tree now has an associated value out of 100, its bootstrap. Can you identify any nodes that are not robust? Unfortunately there is no generally accepted threshold for significant bootstrap robustness, so you must use your judgement.



From the trees that you have produced, which MOMP type would you suggest the new variant (NV) strain belongs to?

Do the *ompA* trees agree with the separation of *C. trachomatis* into trachoma (serotypes A to K) and LGV (L serotypes) biovars?

## ii) MLST gene phylogeny using Artemis

A second typing method used for many bacterial species is multilocus sequence typing (MLST). MLST involves the sequencing of fragments of a number (usually 6 or 7) of housekeeping genes spread around the genome. In true MLST, each different allele for each locus is assigned a number. Each unique allelic profile defines a sequence type (ST).

A number of MLST schemes have been devised for *C. trachomatis*, but we will use the scheme of Dean *et al.* (Emerg Infect Dis. 2009 Sep;15(9):1385-94.), which comprises the following seven loci.

Gene name	Locus tag in L2_cat.embl
<i>glyA</i>	CTL0691
<i>mdhC</i>	CTL0630
<i>pdhA</i>	CTL0497
<i>yhbG</i>	CTL0022
<i>pykF</i>	CTL0586
<i>lysS</i>	CTL0150
<i>leuS</i>	CTL0461

If **Artemis** is not open, start it now and open the L2\_cat.fasta reference. Read in the annotation (L2\_cat.embl) by 'Read Entry Into'. Open the NV.bcf file by selecting 'Read BAM / VCF' from the 'File' menu.

To show the bcf of the plasmid as well as the chromosome, choose **Combine references**

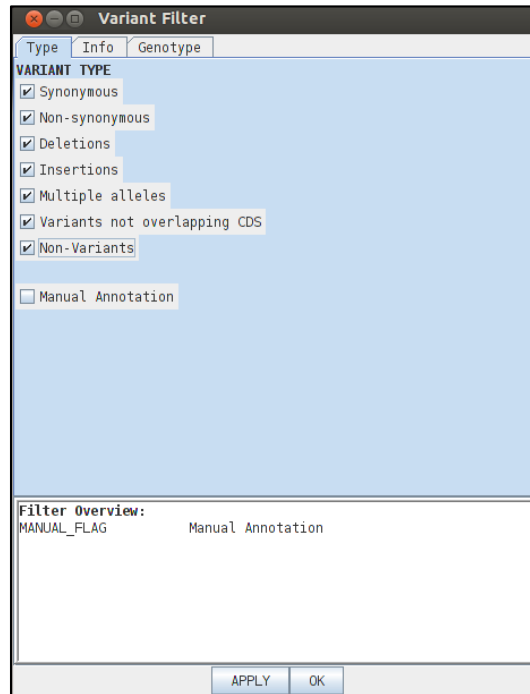
You can reduce the bcfview panel size by clicking on the border and dragging it

To create more space you can hide the second reference panel by clicking on the ... here

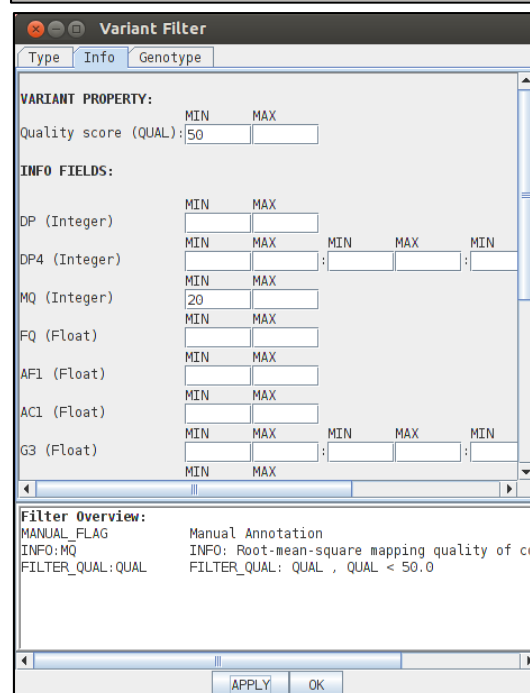
You can turn off the forward and reverse frame lines by right clicking on the reference panel

As you saw in the mapping module, bcf files contain support values for each variant. Before writing out the alignment of the MLST genes it is crucial to filter the variants so that only strongly supported variants are included in the alignment. Right click on the bcfview panel and select the Filter option. You can choose your own filters, but we would suggest something similar to the following:

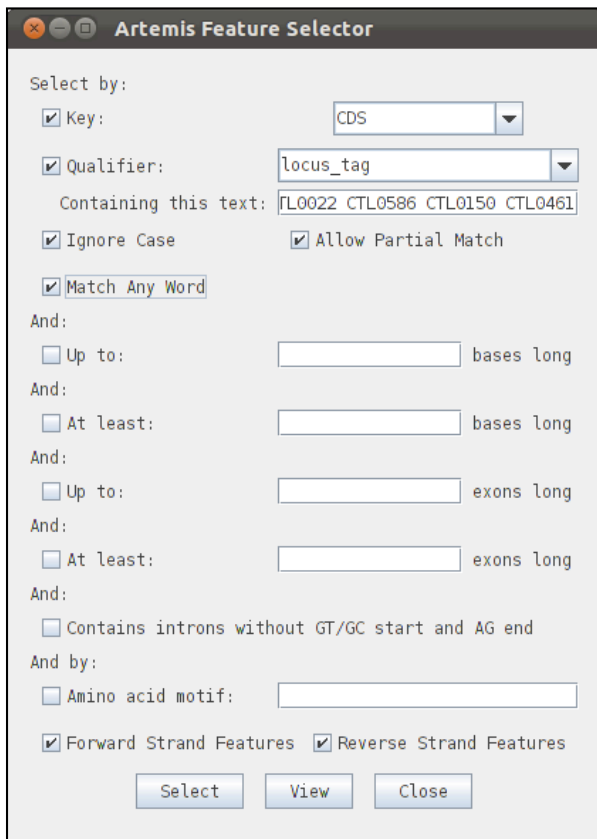
Click these buttons to turn on and off types of variant.



Enter values in these boxes to filter values below a minimum value or above a maximum value



Importantly, note that we have also selected the **'non-variants'** tick box. This option tells Artemis to also show sites which match the reference, rather than just variant sites, provided, of course, that they pass the filters. This information is necessary when saving an alignment, as it allows differentiation between regions that show no variation because they are the same as the reference from those that show no variation due to a lack of mapping. To apply the filters click on the **'Apply'** button. Note what happens to the bcfview in the MLST gene regions. Once you are happy with your filters, close the Variant filter box.

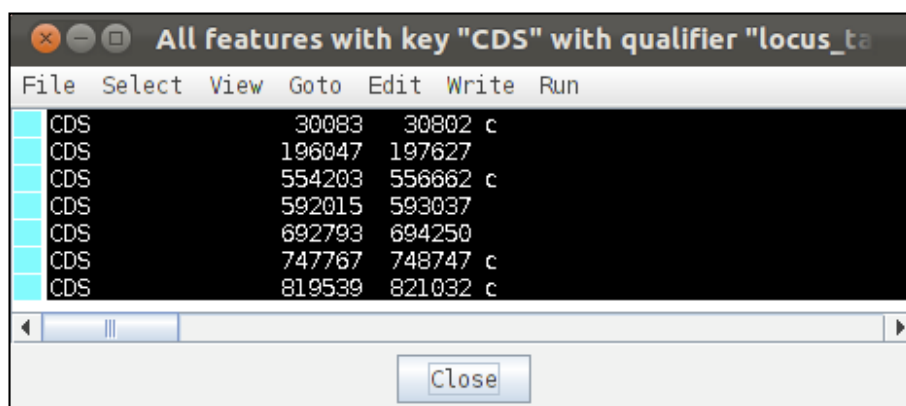


If you have time, have a look at some of the MLST genes in Artemis. How does their diversity within the *C. trachomatis* strains compare to that of *ompA*? Can you think of a possible reason that the diversity may be different?

Rather than use fragments of these loci in a true MLST typing scheme, we will extract the sequences of the seven genes and run a phylogenetic analysis on the concatenated sequence.

There are many ways in which you could identify and select the seven MLST genes in Artemis, but one convenient method is using the **Feature Selector** in the **Select** menu. We want to search for CDSs with the locus tags in the MLST scheme, so in the **Key** dropdown box select **CDS**, and in the **Qualifier** box select or type **locus\_tag**. You then now type the locus tags of the seven loci into the **Containing this text** box, and make sure to tick the **Match Any Word**.

To find the genes click on the select button. To check the results of your search, click on the view button. This will list the features that have been selected by your search.



Check that you have selected the correct seven genes. Next we will write out their sequences into a fasta file.



Feature Type	Start	End	Orientation	Description
CDS	30083	30902	c	
misc_feature	30350	30394	c	PS00211 ABC transporters family signature.
misc_feature	30674	30697	c	PS00017 ATP/GTP-binding site motif A (P-loop).
CDS	30811	31299	c	
misc_feature	31219	31287	c	1 probable transmembrane helix predicted for CTL0023 by TMHMM2.0 at aa 5-27
sig_peptide	31225	31299	c	Signal peptide predicted for CTL0023 by SignalP 2.0 HMM (Signal peptide probability 0.96)
misc_feature	31240	31272	c	PS00019 Prokaryotic membrane lipoprotein lipid attachment site.
CDS	31296	32105	c	
tRNA	32412	32484		tRNA Arg anticodon TCT, Cove score 86.53
CDS	32591	32884		
CDS	32884	33201		
CDS	33263	34290		
CDS	34390	34628		
CDS	34765	36237	c	
CDS	36252	38069	c	
CDS	38449	39456		
misc_feature	38500	38922		HMMFam hit to PF05201, Glutamyl-tRNAglu reductase, N-terminal domain, score 3.9e-54
CDS	40168	40569		

Choose this option to write the sequence of one or more feature in the (e.g. CDSs) correct orientation

Choose this option to write the sequence of any selected region of the genome

We are now ready to write out the alignment of the NV MLST genes. To do this, make sure the correct genes are selected in the embl file and right click on the bcfview panel. From the menu choose **Write** -> **Fasta of selected features**. You will be asked for a filename for the alignment file. Call it NV\_MLST.fasta. Note the three options on the right hand side of the save dialogue box.

The **Combine feature sequences** option is useful if you have selected more than one feature in the reference when choosing to write the bcf sequences. With this option selected, the sequences of each feature will be concatenated together in one file. If you deselect this option you will save one fasta per feature selected. We need to make sure this option is selected

The **Use N for filtered out sites** tells Artemis that when a site fails the chosen filters, that base should be written as an N (unknown) in the alignment. If you deselect this box any site that fails any of the filters will be saved in the alignment as the reference base. Why might it be a bad idea?

The **Use N for sites without non-variant** option is useful when there are non-variant sites confirming the reference sequence as this will then write out 'N' for each of the non-confirmed sites. If you have multiple bcf files open, a fourth option will appear. The **Single fasta** option tells Artemis to save the sequences of multiple bcf files into a single fasta file. If you deselect this option, one fasta file will be saved for each individual bcf. This does not apply here.

We have only output the MLST gene sequences for one *Chlamydia* isolate. If we had opened more than one bcf file in Artemis, we could have output the sequences for all of those isolates in one go to create an alignment of MLST gene sequences. However, running Artemis with many bcf files open can be slow, especially on the USB stick. An alternative is to output the sequence for each strain into a separate file in exactly the same way as we have for NV. You can then concatenate the separate files into one alignment file using the ‘**cat**’ command. Remember, though that the reference will be included in each file created in Artemis.

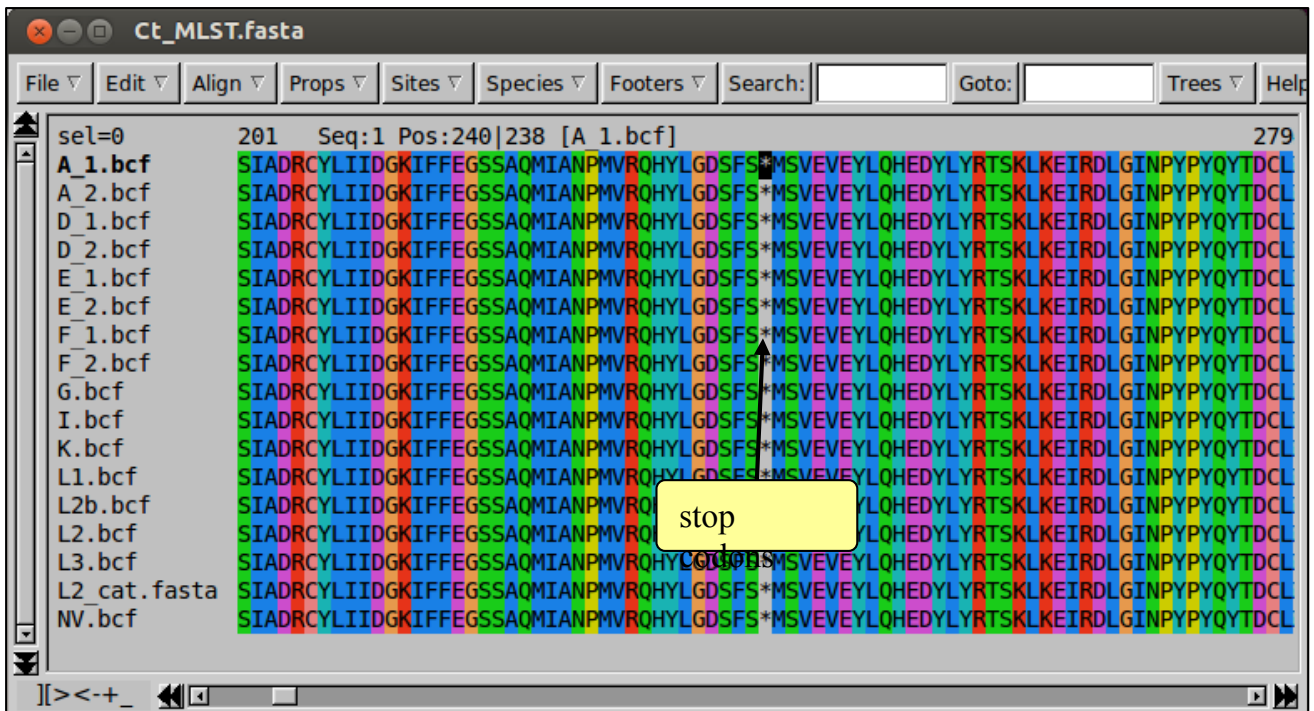
For speed we have provided you with an alignment of the MLST genes from the 15 other *C. trachomatis* isolates you included in the *ompA* tree. These were produced in exactly the same way as the NV\_MLST.fasta file you just created. We will use **cat** to add the reference and NV sequences and make a file containing all 17 isolates by typing:

```
cat Others_MLST.fasta NV_MLST.fasta > Ct_MLST.fasta
```

Open Ct\_MLST.fasta in Seaview.

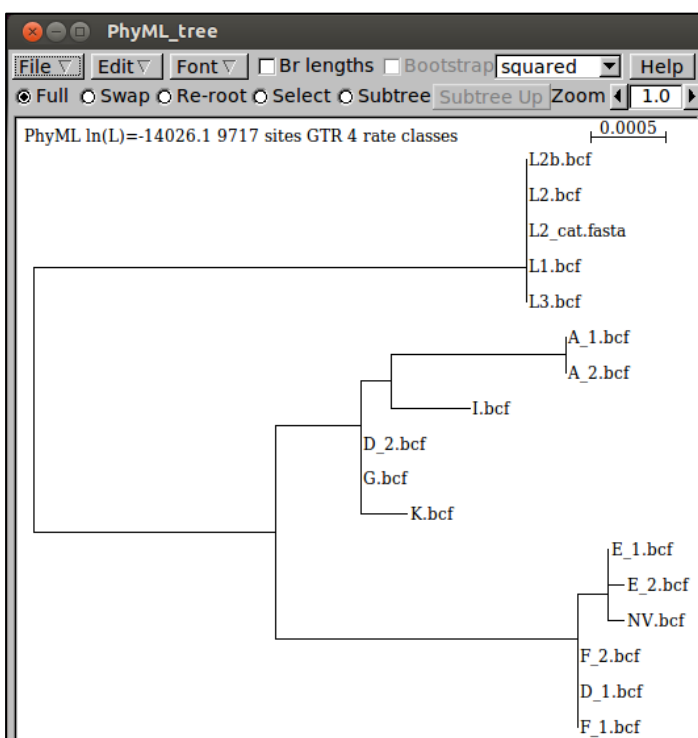
Notice that there is only one sequence for each strain. This is because Artemis has concatenated the sequences of the seven loci into one long sequence.

To check all seven genes are in the alignment you can use Seaview to translate the nucleotides into their corresponding amino acids. To do this, select 'Props' and then 'View as proteins'



To check you have seven genes in your alignment you can count the number of stop codons, which are represented as asterisks (\*) in the protein view. To do this you can either scroll along the sequence or use the search box.

Construct a maximum likelihood tree of the MLST alignment as nucleotides.



How does the tree compare with the *ompA* tree you made earlier?  
 Does the MLST gene tree support the splitting of *C. trachomatis* into trachoma and LGV biovars?  
 Do the strains cluster by serotype in the MLST tree?  
 What biological processes could account for these discrepancies?  
 Do you think *ompA* is a good gene for typing *C. trachomatis* isolates?

### iii) Phylogeny from whole genome SNPs

In the past few years, with whole genome data being produced for large numbers of bacterial isolates, it has become possible to use variation from whole genomes for phylogenetic reconstruction.

Although it is possible to extract whole genome variation from Artemis in the same way as we did for the MLST genes, this is not what Artemis was designed to do, making the process slow. However, there are ways we can extract SNPs from bcf variation files at the command line. Here we will use the `vcfutils.pl` script that is included with samtools and the `fastq` script.

At the prompt type the following command:

```
bcftools view -cg NV.bcf "AM884176.1" | vcfutils.pl vcf2fq -d 5 - > NV_WGS.fastq
```

Although this command line looks complex, it is just things you've seen before put together.

The command is in two parts separated by a pipe '|', which simply tells the command line to take the output from the first command and use it as the input for the second command. The first command here is to use **bcftools view** to change the compressed bcf file into a text file that is readable, which is then used as the input for `vcfutils.pl`. The **-cg** tells `bcftools view` to call SNPs. After the bcf file name you will notice we have included '**AM884176.1**'. This is the name of the chromosome in the reference file (named using its embl accession number), and tells `bcftools` to only output variants on this contig.

The second command uses **vcfutils.pl vcf2fq** to filter the base calls and output the sequence in fastq format. **-d** is a minimum depth cutoff, similar to that you have seen in Artemis. The final '-' in the command line tells it that the input is coming from the pipe rather than a file. Finally the result is redirected into a file called **NV\_WGS.fastq**.

Next we need to convert the fastq format file to fasta:

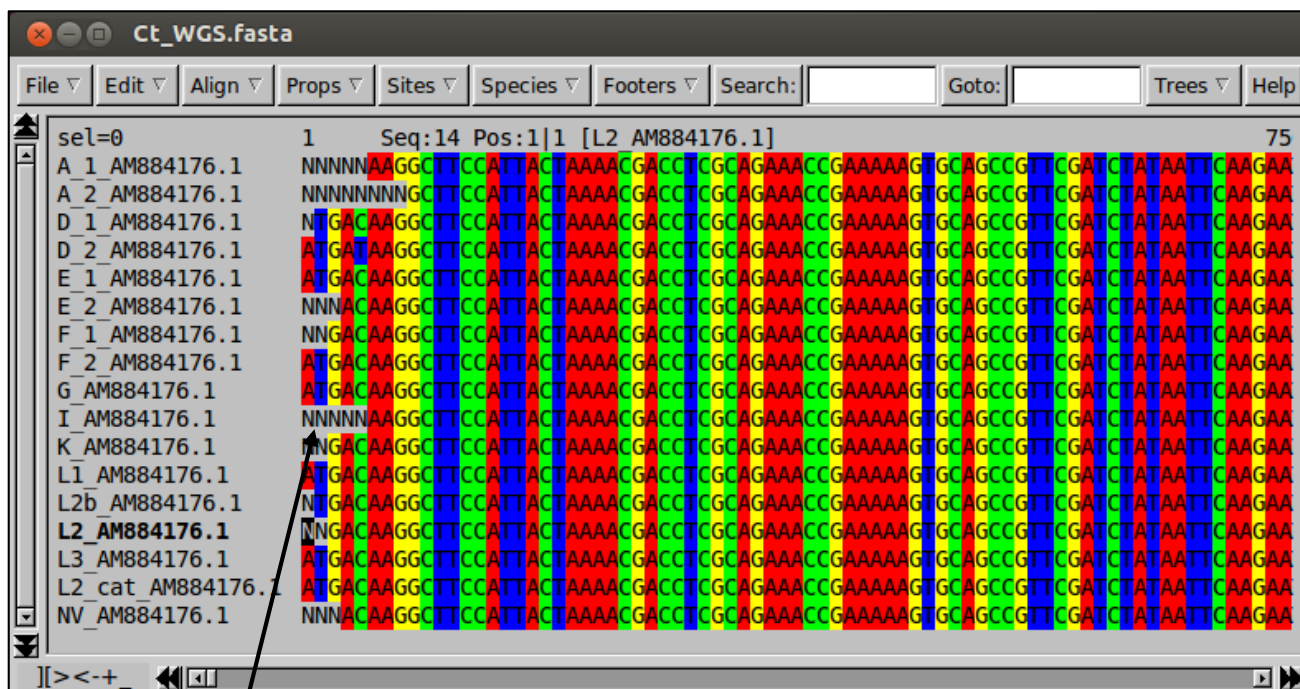
```
seqtk seq -l 60 -A NV_WGS.fastq > NV_WGS.fasta
```

The sequence produced is not the complete genome of the NV isolate, as mapping only allows variants to be called in regions present in the reference genome, as where the amount of variation is not too great. Instead, the sequence produced is a pseudosequence of the bases mapped against each base of the reference genome.

We have created pseudosequences for the other 15 isolates, and included these, along with the reference chromosome sequence in a file called **Others\_WGS.fasta**

Using **cat**, concatenate the **Others\_WGS.fasta** and **NV\_WGS.fasta** files into a single alignment called **Ct\_WGS.fasta**

Try to open the alignment in **Seaview**. You will find it is much larger than the other datasets you have used.



The Ns in the alignment represent bases that cannot be called from the mapping. This may be because there is no mapping in a region due to a true deletion, or because the mapping of that base fails to meet one of the filters imposed.

Although it is possible to run a tree on such a large dataset because phylogenetic methods reduce the complexity by only analysing identical site patterns once, this may still be very slow. Instead, we will extract only those sites which contain variation and run a tree on those using a C script called **snp\_sites**.

At the command line, run this command:

```
snp-sites -o Ct_WGS_SNPs.fasta Ct_WGS.fasta
```

Open the SNP alignment in **Seaview** and make a tree as before. Do not include the invariant sites parameter, as this would not make sense – we have just removed all invariant sites from the dataset.

How does the tree of whole genome SNPs compare to the other trees you have made?

What could cause analyses of different parts of the genome to produce different phylogenies?

What does this tell you about *ompA*, MLST and whole genome SNPs for typing and surveillance of *C. trachomatis*?



## Other useful Seaview features

### File Menu:

- **Save as...** allows you to save your alignment in many formats, including fasta, phylip (used by many phylogenetics programs), nexus (used by PAUP\* and MrBayes), and many more.
- **Save selection.** Allows you to save a region of your alignment masked with a set or an alignment of only the selected taxa.
- **Concatenate.** Allows you to join multiple alignments together either based on the names of the taxa or the order of the taxa in the alignment

### Edit Menu:

- In this menu there are options to **delete**, **add**, **edit**, **reverse** and **reverse complement** sequences in your alignment
- You can view a **dot plot** of any selected pair of sequences

### Props Menu:

- This menu contains options about how the sequences are shown. If you select the **Allow seq. edition** option, you can manually add or delete bases from the sequences

### Sites Menu:

- This menu allows you to create a mask (set) under the alignment, which you can use to select a set of sites.
- When sites (also taxa) are selected, any tree run will only include the selected sites and taxa.
- You can save your selection using the **save selection** option in the **file** menu

### Search and Goto:

- Search allows you to find a sequence of nucleotides or amino acids in a sequence.
- Goto allows you to specify a base or amino acid number to move to in the alignment

## FigTree: a more versatile tree viewer

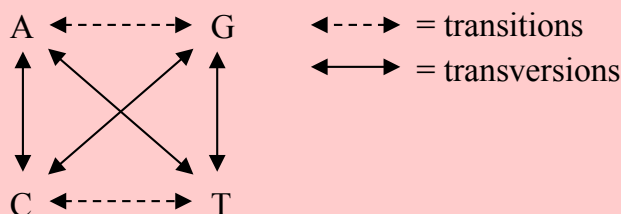
Also included on your disk is a tree viewing program called FigTree. FigTree can open Newick format trees, as output by Seaview (see page 8). FigTree is more versatile than the tree viewer in Seaview, allowing you to colour branches and taxa, redraw the tree in a number of ways, collapse branches and output the results in a large number of graphics formats including eps and pdf. It is particularly useful for preparing figures for manuscripts. If you have time you may find it useful to try opening and editing your tree in FigTree. FigTree can be opened by typing '**figtree**' into the terminal.

## Protein models

In this exercise you have become familiar with phylogenetic analysis of nucleotide data. Analysing sequences on the protein level is very similar, but there are some differences in the substitution models used.

When analysing nucleotide sequences our models optimise substitution rates from the data. For a JC69 model this means optimising a single rate for all changes, while for a GTR 6 substitution rates need to be optimised.

The 6 nucleotide substitutions:



When analysing amino acid data we have 20 “standard” amino acids, which would require 190 substitution rates to be estimated. In most circumstances our data does not contain enough information to estimate all of these parameters, and even if it did, the calculation time would become prohibitive.

Instead we usually use a pre-made matrix of substitution rates calculated from large collections of alignments of proteins of known function. PhyML provides a range of options:

Model	Alignment source
LG	Nuclear globular proteins
WAG	
Dayhoff	
Blosum62	
MtREV	Vertebrate mitochondrial proteins
RtREV	Viral reverse transcriptase proteins
CpREV	Chloroplast proteins
DCMut	Extensions to Dayhoff's PAM matrix
VT	Nuclear globular proteins
MtMam	Mammalian mitochondrial proteins
MtART	Arthropod mitochondrial proteins
HIVw	HIV-1 viral genes
HIVb	

The most commonly used model over the last few years has been the WAG, as it is generally applicable and in most cases gave the best results. However, the new LG model seems to be a further improvement.

The gamma correction for among site rate variation, and invariant sites correction is exactly the same as with nucleotide data



## Model selection

One of the most difficult and important decisions in phylogenetic analyses is which model to choose. Overly simple models are not biologically realistic, and have been shown to produce wrong trees in certain circumstances. For example, it is well known that parsimony and overly simple ML models often place long, unrelated branches together because they underestimate the number of multiple substitutions at sites. This phenomenon, known as long branch attraction, is one of the major causes of error in phylogeny.

Given the problems associated with overly simple models, it is tempting to always use the most complex models available, which estimate more parameters from the data. Often you will find these are the best models for you to use.

AN INCREASE IN THE NUMBER OF MODEL PARAMETERS CAN ONLY INCREASE THE FIT OF THE MODEL TO THE DATA AND THEREFORE CAN ONLY IMPROVE THE LIKELIHOOD.

However, increasing the number of parameters reduces the amount of data we have to estimate the correct value for these parameters.

One way to approach choosing a model is to run a mini analysis with a range of models and compare the results. There are a number of tests that allow us to compare the likelihoods of models with different numbers of parameters. These include the likelihood ratio test, which you have used in this module, and the AIC.

You can run these tests using the programs jMODELTEST for nucleotides (a reduced version called Findmodel is available online) and PROTTEST for proteins:

jMODELTEST:

<http://darwin.uvigo.es/software/jMODELTEST.html>

Findmodel:

<http://www.hiv.lanl.gov/content/sequence/findmodel/findmodel.html>

Prottest:

<http://darwin.uvigo.es/software/prottest.html>

**A final warning note:** never select a model with one of these methods without thinking about the biology. If you're working on *Plasmodium*, would it be sensible to use a protein model produced from HIV sequences?

## Alignment: some things to remember

- It is worth spending time to make a good alignment. If your alignment is wrong, your phylogeny is also likely to be wrong
- Progressive alignment is a mathematical process that is completely independent of biological reality
- Can be a very good estimate
- Can be an impossibly poor estimate
- Requires user input and skill
- Treat cautiously
- Can (usually) be improved by eye
- Often helps to have colour-coding
- Depending on the use, the you should be able to make a judgement on those regions that are reliable or not
- For phylogeny reconstruction, only use those positions whose hypothesis of positional homology is strong

FINALLY...

IT IS NOT USUALLY SENSIBLE TO ALIGN PROTEINS AT THE NUCLEOTIDE LEVEL:

- The result might be highly-implausible and might not reflect what is known about biological processes.
- It is much more sensible to translate the sequences to their corresponding amino acid sequences, align these protein sequences and then put the gaps in the DNA sequences according to where they are found in the amino acid alignment.

# Module 6

## Helminth *de novo* genome assembly

### Overview and aims

The aim of this practical class is to introduce you to some of the concepts involved in the assembly of a eukaryotic genome. The workflow that you will be using is not extensive, nor comprehensive, and like many bioinformatic tasks, there are many tools that do a similar job. However, this workflow should give you an overview of how to perform a genome assembly, and identify some of the ways to assess (and maybe improve) the quality of your genome assembly.

The data you will be working with in this tutorial comes from a species of parasitic blood fluke named *Schistosoma mansoni*. This parasite causes a disease called schistosomiasis that affects approximately 200 million people who reside in Africa, the Middle East, the Caribbean, Brazil, Venezuela and Suriname. The lifecycle of the parasite is shown in Figure 1, which illustrates two main life history stages: (1) the maturation into adulthood and sexual reproduction in the mammalian host (here a human), and (2) clonal reproduction and transmissible stage in an intermediate host (typically a snail), and in the lakes and streams in which the snail resides. The DNA for sequencing was derived from a maintained laboratory line of *S. mansoni* at the Wellcome Sanger Institute, in which the mammalian host is a mouse in the maintenance of the life cycle

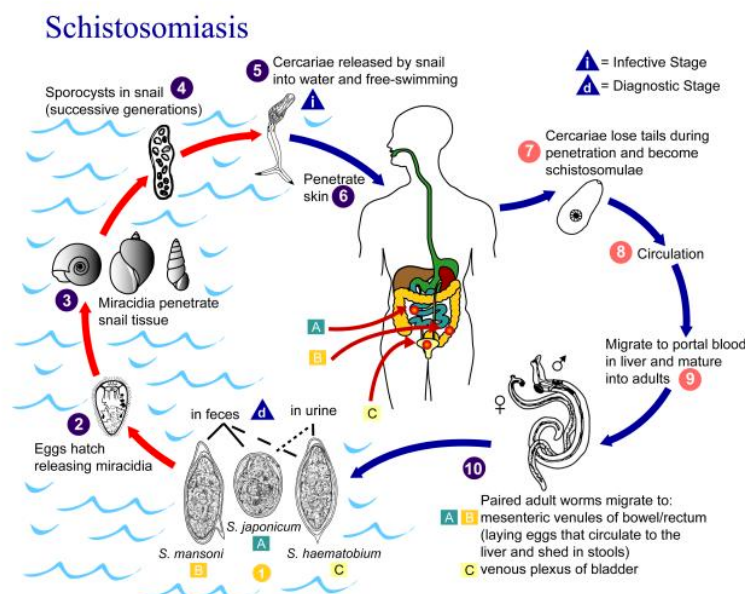


Figure 1. *Schistosoma mansoni* lifecycle.

The data you will be using was generated by the Parasite Genomics group at the Wellcome Sanger Institute; a draft genome sequence was initially published in 2009 (Berriman et al. 2009 <https://doi:10.1038/nature08160>), followed by an improved version in 2013 (Protasio et al. <https://doi.org/10.1371/journal.pntd.0001455>); however, it has subsequently been the focus of further improvement, particularly using long read Pacbio data and genetic mapping, and now is largely complete in chromosome-scale scaffolds (7 autosomes + Z/W sex chromosomes) that total approximately 380 Mb in length.

Genome assembly of a 380 Mb genome is a relatively big task and is suited to a computer cluster environment, and not personal computers. To make things manageable in terms of computer power and run time, we have selected data that corresponds to a single *S. mansoni* autosome, designated chromosome IV, which is approximately 47 Mb in length. While only a fraction of the *S. mansoni* genome, a single chromosome is comparatively *huge* relatively to many prokaryotic genomes, and still comes with the complexity of an eukaryotic genome that is not often present in a prokaryote.

To assemble the 47 Mb chromosome IV, we will use the following workflow and demonstrate following concepts:

Step 1: Checking raw sequencing data before assembly

- Tools used: FastQC, MultiQC, Kraken

Step 2: Estimating your genome size from raw sequence data

- Tools used: Jellyfish, GenomeScope

Step 3: Performing a genome assembly using either Illumina short read or Pacbio long read data

- Tools used: Canu, Spades, Miniasm

Step 4: Comparison of your assemblies against a known reference sequence

- Tools used: Nucmer, Assemblytics

Step 5: Further exploration of your genome assemblies

- Tools used: Bandage, Nucmer, Genome Ribbon

## Tips to get you started

- read the text! They contain lots of hints that should help you to answer some of the questions
- Grey boxes contain instructions for running commands

```
# run FastQC for read 1 and read 2
$ fastqc SM_V7_chr4_illumina_R1.fq
$ fastqc SM_V7_chr4_illumina_R2.fq
```

A line stating with a “#” and is blue is an instruction – it does not need to be typed

A line stating with a “\$” is a command and needs to be typed into the command line to run. Each line that begins with a \$ represents a new command

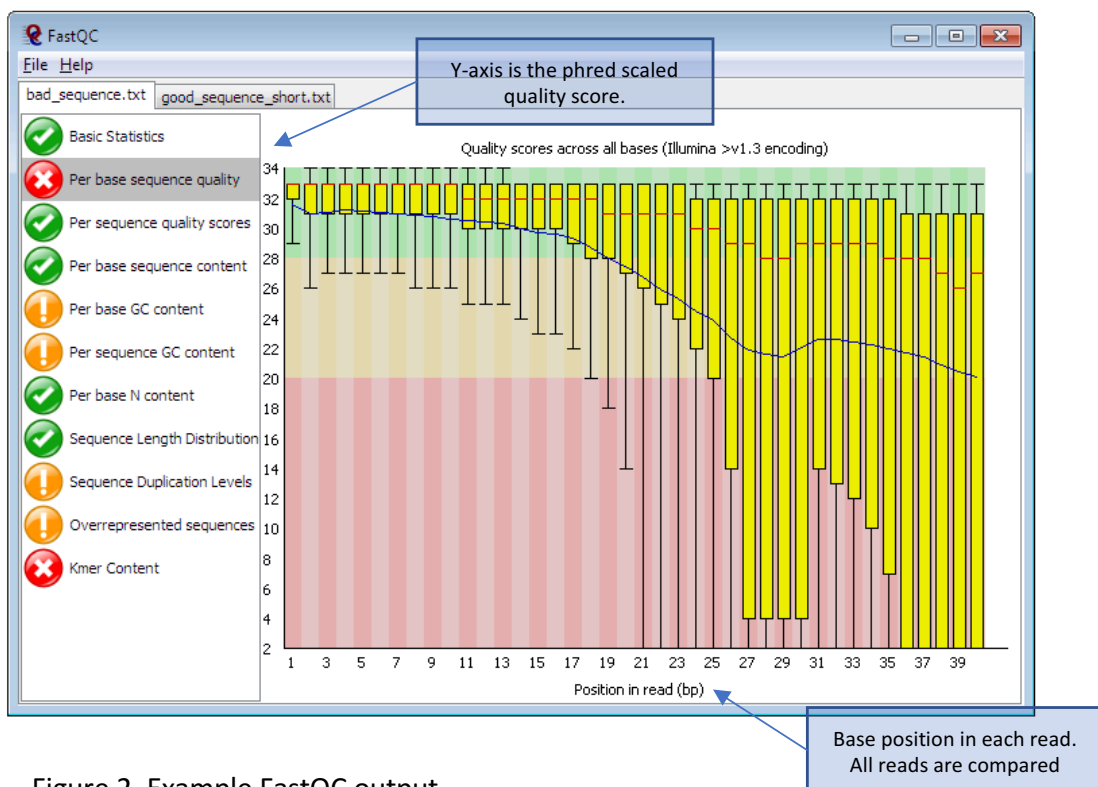
Commands in **BLACK** need to be run. Commands in **RED** do not need to be run – they have been run for you to save time.

```
# your first command – move to the working directory to get started!
$ cd /home/manager/Module_6_helminth_denovo_assembly
```

## Step 1: Checking raw sequencing data before assembly

The first exercise of any genomics project is to turn your sample of interest into sequencing data. There are many steps involved, including sample collection (and storage), DNA extraction (and storage), sequencing library preparation, and then finally submitting and having your DNA library sequenced on one or more of a number of different sequencing platforms. Not surprisingly then is that the success of each step will influence how well your sample will be sequenced and will impact on the quality of the data generated. Exploring and understanding the characteristics of the raw data before any assembly is performed should give you some confidence in whether your data is sufficient to undertake a genome assembly, and may provide some insight into how an assembly will proceed.

We will use two tools to assess different aspects of the raw data. The first tool is called **FastQC** (<https://www.bioinformatics.babraham.ac.uk/projects/fastqc/>). **FastQC** takes raw fastq reads and provides simple graphs and tables to quickly assess the quality of the data. It also highlights where they may be problems in different aspects of your data (NOTE: it is parameterised on human data, i.e., GC content, and so may report as “failing” based on assessing your data [if not human] because it does not look like human – be aware that not all “fails” are bad). The features of the raw data that are being assessed is presented in the left-hand panel of Figure 2.



The main panel of this figure shows an example of the comparison of the distribution of per base quality (Phred score, on the y-axis) per base position in the read (x-axis). Phred scores above 30 are typically considered to be good quality for an Illumina read. In this case, it shows higher quality bases toward the start of the read (in the green section), followed by a decrease in quality along the read, in which the quality drops into the yellow (Phred < 30) and then into the red (Phred < 20). Some examples of “good” and “bad” quality data is found in the “Example Reports” section of the FastQC website.

The main features of the data to look out for will be:

- Per base sequence quality:* Good overall indicator of data quality. Should remain mostly in the green, however, will drop in quality over the length of the read. Longer reads will show great drop, and R2 will show greater drop compared to R1.
- Per base sequence content:* The base frequency of each nucleotide should reflect the GC content, with  $f(A)=f(T)$  and  $f(C)=f(G)$ . The expectation is that these lines should remain horizontal and even throughout the read. Are they even?
- Per base sequence GC:* Abnormal GC distribution is a good indicator of contamination. Does the GC profile fit the expected GC content for your species of interest? Is it a smooth distribution, or are there spikes?
- Per base N content:* Is there an excess of positions in the reads for which a “N” base was called? Excess Ns can often indicate an issue with the sequencing.
- Sequence duplication levels:* Is there excessive duplication? Duplication may suggest artefacts generated during library preparation / PCR amplification
- Adapter content:* What is the proportion of known Illumina adapters that are present in the data?

FastQC reports would typically be generated for each read set. However, if you have a lot of datasets, it can be tedious to look at each one individually. We are going to use a visualisation tool called **MultiQC** (<http://multiqc.info/>), which can automatically detect the **FastQC** data once generated and arrange the data into single plots for all read sets. **MultiQC** is not only good for visualising **FastQC** output; it supports the QC of over 60 bioinformatic tools, including mapping, SNP calling, transcriptomic analyses etc. It is a great way of summarising lots of datasets in one place.



The second tool to assess the quality of your raw data is **Kraken** (<https://ccb.jhu.edu/software/kraken/>). **Kraken** is a fast way of assigning and approximating the abundance of known species based on short DNA sequences called kmers. A kmer is simply a short length of nucleotide sequence of a given length. For example, in a DNA sequence:

DNA sequence: ATGCGTCATGC

Kmer = 4 : ATGC, TGCG, GCGT, CGTC, GTCA, TCAT, CATG, ATGC

Kmer = 4 (n): ATGC (2), TGCG (1), GCGT (1), CGTC (1), GTCA (1), TCAT (1), CATG (1)

I.e. ATGC was seen twice, while the rest were seen only once.

**Kraken** works by aligning kmers from your DNA sequence against known kmer frequency data for different species in a kraken database. It will therefore only assign species that it knows, else, it calls the sequence “unclassified”. Most **kraken** databases contain comprehensive bacterial and viral species lists, however, it may also contain human and mouse profiles. **Kraken** databases can be customized to include any species with DNA sequence available. Therefore, if you are investigating one of the species in the kraken database, running kraken will give you a good estimate of the amount of reads specifically from that species. If your species is not in the database, then you would expect most if not all reads to fall into the “unclassified” category. Either way, this approach can serve as an effective screen for contaminants in your sequencing reads.

### Tasks:

- run fastqc
- visualise output of FastQC using MultiQC and check sequence quality
- view the Kraken report to determine if there are any contaminants

```
# go to the working directory
$ cd /home/manager/Module_6_helminth_denovo_assembly/step_1

# run FastQC for read 1 and read 2
$ fastqc SM_V7_chr4_illumina_R1.fq
$ fastqc SM_V7_chr4_illumina_R2.fq

# Once FastQC has finished running, run MultiQC and visualise
output in web browser
$ multiqc .
$ firefox multiqc_report.html &

# Once you have finished exploring FastQC/MultiQC, open the
kraken report to determine the proportion of the read data
that is “unclassified”.
$ head -n 100 kraken.report
```

You should only need to top 100 lines or so of this much larger file, which we can access using “head -n 100”

## The kraken report

- The output of kraken-report is tab-delimited, with one line per taxon. The fields of the output, from left-to-right, are as follows:
  - Percentage of reads covered by the clade rooted at this taxon
  - Number of reads covered by the clade rooted at this taxon
  - Number of reads assigned directly to this taxon
  - A rank code, indicating (U)nclassified, (D)omain, (K)ingdom, (P)hylum, (C)lass, (O)rder, (F)amily, (G)enus, or (S)pecies. All other ranks are simply '-'.
    - NCBI taxonomy ID
    - indented scientific name

## Questions you should be asking:

### - FastQC / MultiQC output

- What are the similarities / differences between read 1 and read 2?
- Are the base quality and nucleotide frequency distributions relatively level, or are they uneven?

### - Kraken

- What proportion of the reads are “unclassified”, and therefore potentially *S. mansoni* reads?
- What looks to be the main contaminant? Why might this be so?

## Step 2: Estimating your genome size from raw sequence data

In this tutorial, we are in the unique position to already know what the length of the chromosome sequence we are trying to assemble. However, if sequencing a new species for the first time, we may not know what the genome size is. Knowledge of the genome size can be an important piece of information in its own right, however, it can also be useful to help parameterise some stages of the genome assembly.

We can estimate the genome size based a calculation of the kmer coverage of our reads. We introduced kmers in the last section – they are simply a string of nucleotides of a given length. The relationship between kmer coverage and genome size is described by:

$$C_{kmer} = \frac{L - k}{L} * \frac{N_{reads} * L}{G}$$

Where  $C_{kmer}$  is the average kmer coverage,  $N_{reads}$  is the number of reads,  $L$  is the average read length,  $k$  is the length of the kmer, and  $G$  is the genome size (Vurture et al 2017; <https://doi.org/10.1093/bioinformatics/btx153>; supplementary data). It is not important to know this equation, however, we illustrate it to demonstrate that kmer coverage can be informative about genome size.

There are a number of different tools available to count kmers (<https://omictools.com/k-mer-counters-category>) and to calculate the genome size. Today, we are going to count kmers using **Jellyfish** (<http://www.genome.umd.edu/jellyfish.html>), and use the output to calculate the genome size using a online web tool called **GenomeScope** (<http://qb.cshl.edu/genomescope/info.php>).

You can explore some examples of kmer spectra and genome size estimates on the GenomeScope website. Figure 2 presents an example of a *Drosophila* dataset (quick access here: <http://genomescope.org/analysis.php?code=example5>); the difference between the two plots is the scale on the axes, with the first plot zoomed in, and the second plot zoomed further out. In both plots, the blue data represents the actual kmer frequency data generated by Jellyfish. The dark black line represents a model of the kmer spectra, used to characterise the number of peaks, which are indicated by the black dashed line. The orange line represents very rare kmers (low coverage), which are likely associated with sequencing errors and are ignored. This data is used to estimate the genome size, taking into account the heterozygosity and error of the sequencing reads.

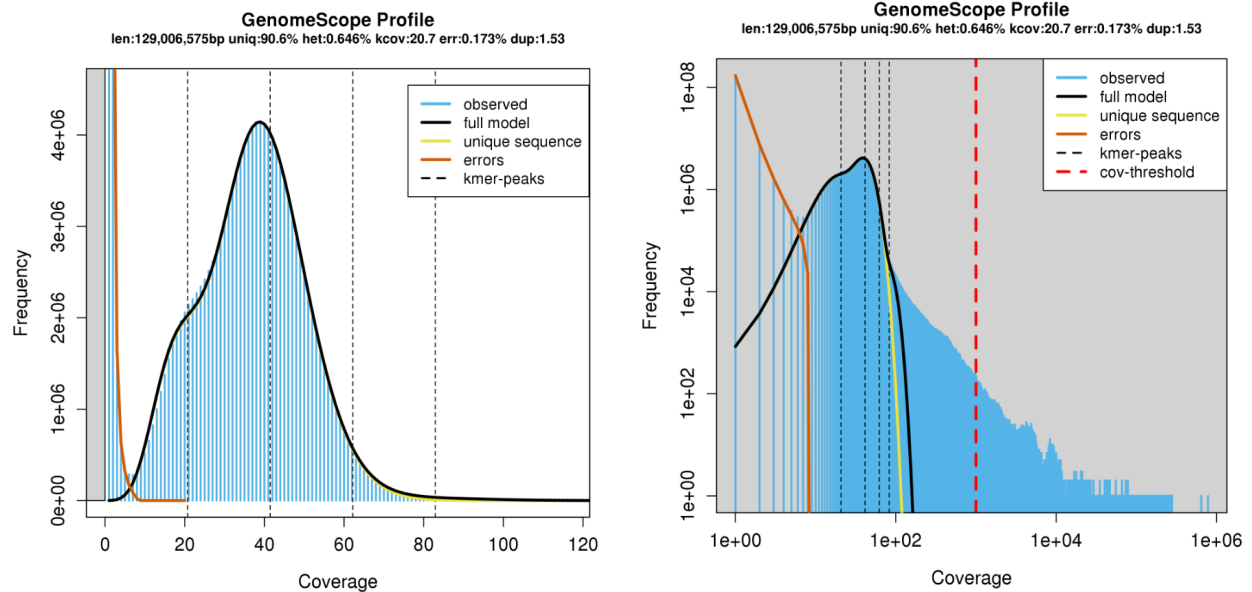


Figure 3. Example GenomeScope output. Kmer coverage is presented on the x-axis, and kmer frequency on the y-axis.

- Tasks
  - Run jellyfish on your raw sequencing data
  - Upload your kmer count data to GenomeScope and estimate the genome size

```
# go to the working directory
$ cd /home/manager/Module_6_helminth_denovo_assembly/step_2

# run Jellyfish commands. The first step will take a few
minutes
$ jellyfish count -C -m 21 -s 1000000000 \
  -t 4 ../step_1/*.fq -o my_reads.jf
$ jellyfish histo -t 4 my_reads.jf \
  > my_reads.histo

# Once Jellyfish commands have been run and you have the
"reads.histo" file, open the webpage:
http://qb.cshl.edu/genomescope/
# Upload reads.histo to GenomeScope
```

The "\ " allow a long command to be split over multiple lines. They don't need to be included if you write the whole command on a single line

GenomeScope Home Info Examples ▾

# GenomeScope

Estimate genome heterozygosity, repeat content, and size from sequencing reads using a kmer-based statistical approach.

Run GenomeScope

Click or drop .histo file here to upload

Drag and drop your "reads.histo" file here

Description my sample

Kmer length 21

Read length 100

Max kmer coverage 1000

Submit

It is not necessary to change any other parameters. Just submit!

Instructions

Upload results from running Jellyfish. Example: inputk21.hist

Instructions for running Jellyfish:

1. Download and install jellyfish from: <http://www.genome.umd.edu/jellyfish.html#Release>
2. Count kmers using jellyfish:
 

```
$ jellyfish count -C -m 21 -s 1000000000 -t 10 *.fastq -o reads.jf
```

Note you should adjust the memory (-s) and threads (-t) parameter according to your server. This example will use 10 threads and 1GB of RAM. The kmer length (-m) may need to be scaled if you have low coverage or a high error rate. You should always use "canonical kmers" (-C)

3. Export the kmer count histogram
 

```
$ jellyfish histo -t 10 reads.jf > reads.histo
```

Again the thread count (-t) should be scaled according to your server.

4. Upload reads.histo to GenomeScope

Note: High copy-number DNA such as chloroplasts can confuse the model. Set a max kmer coverage to avoid this. Default is -1 meaning no filter.

Figure 4. GenomeScope webpage. <http://qb.cshl.edu/genomescope/>

- NOTE: if you would like to use your own data, check the read length and modify the input above accordingly.

### Questions you should be asking:

- what is my predicted genome / chromosome size?
- how does it compare to the expected size?
- what does changing the kmer length do?

## Step 3: Performing a genome assembly using either Illumina short read or Pacbio long read data

Now that you have performed some QC on your raw data and estimated your genome size, it is now time to perform a genome assembly. There are a huge number of tools dedicated to genome assembly; OMICS tools describes 163 dedicated for *de novo* genome assembly (<https://omictools.com/genome-assembly-category>), however, there are likely others. Furthermore, there are likely to be at least as many tools that value-add to a genome assembly, including but not limited to scaffolders, circularisers, gap closers etc. The choice of assembler and subsequent add-ons is dependent on the type of data available, type of organism, i.e., haploid, diploid etc, genome size, and complexity of the task among other variables.

The aim of this practical is not to assess these tools or promote any particular tool(s) in any meaningful way, but to compare and contrast two technologies commonly used in genome assembly: Illumina short-read and Pacbio long read.

Illumina short read sequencing has been the workhorse of genome assembly and resequencing studies for the last few years, and continues to be the main technology for high throughput genome sequencing. This is because it is possible to sequence millions to billions of short reads at the same time. A genome assembly using Illumina short reads begins by fragmenting DNA into ~300-500 bp lengths (less than 1000 bp), after which universal sequencing adapters are ligated to each end to generate a **sequencing library**. These adapters enable a site for a sequencing primer to bind, the attachment of the library read to the sequencer, and may contain barcoding indices to allow sample multiplexing. Sequencing is typically performed using a **paired-end** chemistry, which means that two reads are generated per library fragment, one from the beginning of the fragment, ie. read 1, and one from the end, ie. read 2. Depending on the chemistry and sequencer used, these paired-reads will each be ~100-250 bp in length; therefore, some read 1 and read 2 pairs will overlap, whereas others will be separated by a gap, dependent on the library fragment and sequencing read lengths. After sequencing, paired-end reads (which maintain their relationship and orientation via information coded in their name in the fastq output files) are assembled, resulting in **contigs** – contiguous stretches of assembled sequence that do not contain gaps - and **scaffolds** – which are assembled sequence that do have gaps, typically generated by the spanning of two contigs by read pairs that do not overlap and lack nucleotide coverage in the gap. The contiguity is therefore dependent on the ability to find unique overlaps between read pairs; features of the genome, including but not limited to repetitive and/or low complexity regions, or even inherent genetic diversity in the sequences, cause uncertainty in the assembly and often prevents further extension of a contig or scaffold. To overcome some of these difficulties, library preparation approaches to produce **mate-pair** or **jumping libraries** may be performed, which increase the gap distance between the paired-end reads, ie., 3-kb, 8-kb, 20-kb, and in turn, may span the difficult to assembly region; this results in an increase in the scaffold- but not contig length overall.

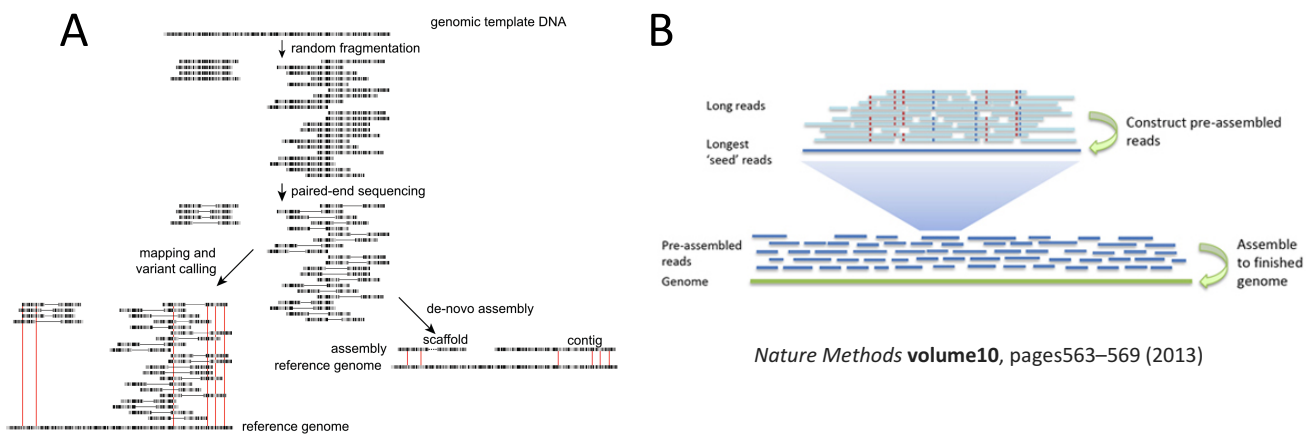


Figure 5. Overview of Illumina short read (A) and Pacbio long read assembly approaches (B).

While Illumina library preparation aims to sequence from fragments of DNA that are only a few hundred base pairs long, Pacbio sequencing aims to sequence DNA fragments that are tens of kilobases in length, i.e. 10s-100s of times longer than Illumina reads. The key advantage of this approach is that many short, complicated genome regions that would have broken an Illumina assembly are spanned by Pacbio long reads, and therefore can be assembled accurately. Moreover, the longer read lengths increase the probability of identifying unique overlaps between reads. Both features enable significantly longer contig lengths from a Pacbio assembly when compared to an Illumina assembly alone.

One feature of all sequencing technologies is that sequence quality declines over the read length - you should have observed this in your FastQC analysis of raw Illumina reads (*Step 1: Checking raw sequencing data before assembly*). Pacbio reads are not only much longer than Illumina reads, but that when sequenced, the raw reads produced are derived from a single molecule of DNA. This differs from Illumina reads, in which a "raw" (but really, a consensus sequence) is generated from a cluster of reads representing the original library fragment. For these two reasons, Pacbio reads are more error-prone than Illumina reads. To overcome this, two initial informatic "correction" steps are undertaken prior to assembly (**Figure 5B**). Raw DNA is fragmented and size selected to achieve fragment lengths in the 10s of kilobases, before the addition of barbell adaptors, which provide sequencing primer binding sites. Sequencing is performed by the polymerase attaching to the barbell adaptor, and processing around the circle to produce a raw read, which contains the library insert sequence flanked by the adapter sequences in an array. In the first correction step, the raw read is trimmed to remove adapters, and the library inserts are aligned to produce a consensus sequence. In the second correction step, the longest of the first round consensus sequences (~30-40% of the total reads) are used as a template to map the remaining shorter, more accurate reads; taking the consensus of the mapped reads, in turn, corrects the longer reads. In this way the more error prone long reads increase in quality, which is ideal from an assembly point of view. Only these long, twice corrected reads are used for the genome assembly.

The process of error correction does take a substantial amount of time and compute resources. It has recently been demonstrated that the second error correction step can be sacrificed to significantly increase assembly speed and the cost of assembly base-level



accuracy, i.e., it is uncorrected, and so the assembly error rate is similar to the read error rate. We will perform a raw Pacbio assembly using **Minimap** and **Miniasm** to compare with our other two assemblies.

- Tasks
  - Run the Miniasm command to generate your first Pacbio assembly of Chromosome IV
  - The Canu and Spades assemblies have been provided for you – it would take too long to run these here – however, we have provided the commands for your reference
  - Determine the assembly statistics of each genome assembly

```
# go to the working directory
$ cd /home/manager/Module_6_helminth_denovo_assembly/step_3

# run the Miniasm assembly
$ minimap2 -x ava-pb -t4 SM_V7_chr4_subreads.fa \
    SM_V7_chr4_subreads.fa > SM_V7_chr4.minimap.paf
$ miniasm -f SM_V7_chr4_subreads.fa \
    SM_V7_chr4.minimap.paf > SM_V7_chr4.miniasm.gfa
$ cat SM_V7_chr4.miniasm.gfa |
    awk '$1=="S" { print ">"$2"\n"$3} ' \
    > MINIASM_SM_V7_chr4.contigs.fasta
# run time: step1 ~ 20 mins, 20 Gb RAM, 4 threads, steps2 and
3 are quick (< 1 min)

# run the Canu assembly
$ canu genomeSize=43M -pacbio-raw SM_V7_chr4_subreads.fa \
    -d PB_SM_V7_chr4 -p PB_SM_V7_chr4 \
    java=/software/jdk1.8.0_74/bin/java
# run time: ~ 6h, 30 Gb RAM, 4 threads

# run the Spades assembly
$ dipspades.py -o SPADES_SM_chr4 \
    -1 SM_V7_chr4_illumina_R1.fq \
    -2 SM_V7_chr4_illumina_R2.fq --threads 4
# run time: ~ 50h, 6 Gb RAM, 4 threads
```

Once you have your assemblies, you will probably want to know how well they have come together. We will do this in two ways, first by generating and comparing basic statistics about the assemblies, and secondly from a comparative genomics perspective by visualising how well each assembly compares to the known reference, and to each other (next section: Step 4).

Table 1 below outlines the data we will generate about each assembly. Each is relatively self-explanatory, however, you may not have been introduced to N50 and N50(n). These statistics are a measure of how contiguous a genome assembly is. Imagine if your assembly is sorted by sequence length, ie., longest to shortest; your N50 is defined as the sequence length at which 50% of the entire assembly is contained in contigs or scaffolds equal to or larger than this contig. It is essentially the midpoint of the assembly. The N50(n) is simply the contig number in which the N50 base is found. More contiguous assemblies will have a higher N50 (and lower N50(n)), whereas more fragmented assemblies will show the opposite trend. Note that you can artificially increase N50 by randomly joining sequences together, and therefore, misassembly or overassembly can inflate N50 values. It is important to not completely rely on N50 as absolute truth and to perform other assembly validations if possible.

```
# calculate the assembly statistics for all three assemblies,
and complete Table 1 below.
```

```
$ assembly-stats PB_SM_V7_chr4.contigs.fasta
$ assembly-stats MINIASM_SM_V7_chr4.contigs.fasta
$ assembly-stats SPADES_SM_V7_chr4.consensus_contigs.fasta
```

Table 1. Comparison of assembly stats

	Pacbio (Canu)	Pacbio (Miniasm)	Illumina (Spades/dispades)
Assembly size			
Number of sequences			
Longest sequence			
Average size			
N50			
N50 (n)			

### Questions you should be asking:

- how do my assemblies compare to the expected size of chromosome IV?
- what is the impact of long reads versus short reads on assembly contiguity?
- how did the uncorrected (Minimap/miniasm) assembly compare to the corrected Canu assembly?

## Step 4: Comparison of your assemblies against the known reference sequence

Now that we have three independent genome assemblies, we would like to see how they compare to the reference chromosome IV sequence. This is only possible because we already have a reference sequence, however, if you have a closely related species with a more contiguous reference, it might be worth trying. If you do not have a good reference to compare against, you could simply compare different versions of the *de novo* assembly to see how they compare (we would like you to do this if you have time).

There are a number of ways to compare genomes. We will be using **nucmer** to do the DNA vs DNA sequence comparison, and the web application **Assemblytics** (<http://assemblytics.com/>) to visualise the comparison. **Assemblytics** is a nice way to visualise this comparison, as it not only allows a “zoomed” out view of how the genomes compare (via the Interactive dot plot), but it also provides base-level and small structural variant statistics. These can be informative particular when comparing different sequencing technologies, ie., Illumina versus Pacbio, and may reveal inherent biases in each.

- Tasks
  - Run nucmer of each of the three comparisons, ie. Ref vs PB, ref vs miniasm, ref vs illumina
  - Explore each of the interactive dotplots
  - Compare the base level statistics for each comparison (these are the colour plots)

```
# go to the working directory
$ cd /home/manager/Module_6_helminth_denovo_assembly/step_4

# run nucmer to generate the comparison between the reference
and each genome assembly. We have provided one example, but we
would like you to run all three assemblies against the
reference.

$ nucmer -maxmatch -l 100 -c 500 SM_V7_chr4.fa \
  ../step_3/PB_SM_V7_chr4.contigs.fasta -prefix chr4_v_PB

$ gzip chr4_v_PB.delta

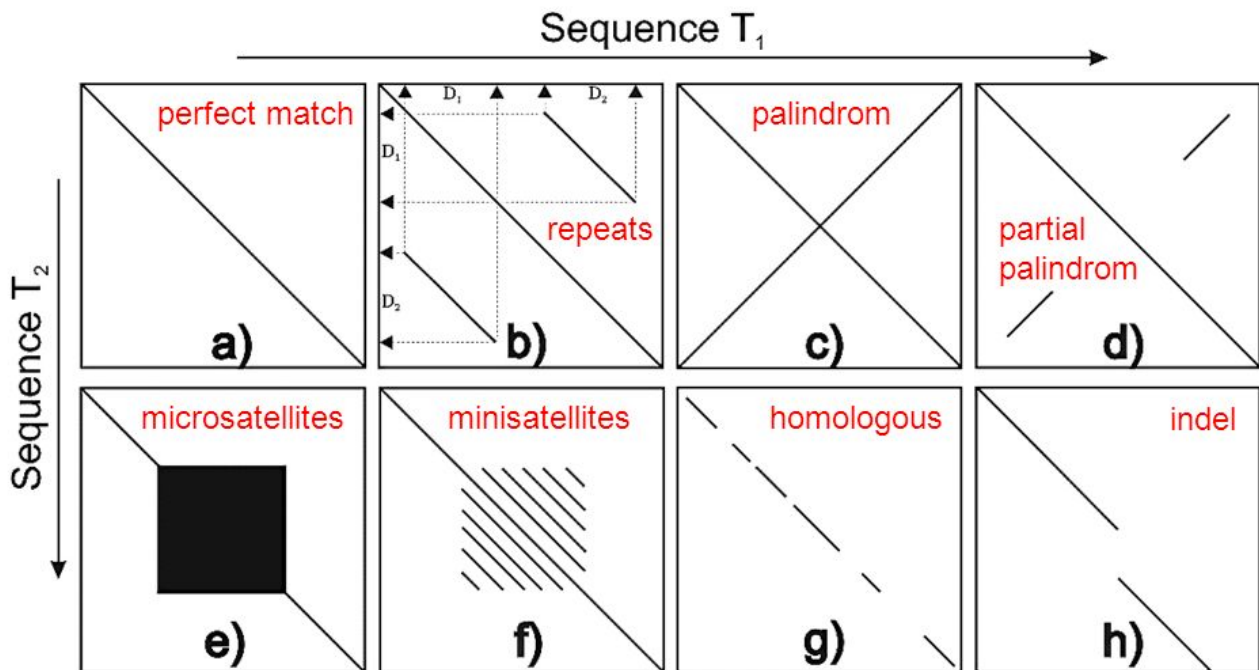
# open the webpage: http://assemblytics.com/
# upload the OUT.delta.gz using the instructions provided
# note that that upload might take a minute or two to analyse
# the raw data and provide the data output / plots
```

**Figure 6. Assemblytics homepage and upload instructions**

Note: once your analysis completed successfully on the website, it will generate a http link that you can use to visualise your data even after you have closed your browser down. This is nice, as it means you can easily share this analysis via email of the link. Make a note of each http link for each comparison so you can compare each comparison.

**Figure 7. Assemblytics output – saving the http link for future reference**

To help you visualise and interpret the interactive dot plot, we have provided some examples of pairwise DNA sequence comparisons that are commonly observed (Figure 7). Ideally, we are looking for a perfect match (a), however, there are many feature of a genome that either complicate, and often break, assemblies, including repeats (b), palindromes (c), and low complexity repeats such as microsatellites (e) to name a few. See what features are present in your assemblies, and if there are features associated with the ends of contigs that might be associated with breaking your assemblies.



**Figure 8. Schematic of dot plot examples.** Originally from [goo.gl/P4QTFd](http://goo.gl/P4QTFd); adapted from <http://slideplayer.com/slide/10357320/>

### Questions you should be asking:

- how does each assembly compare against the reference?
- particularly in the ref vs PB dot plot comparison, what sequence features are found and sequence ends, and why might they be there?
- are there base level characteristics found in one assembly but not the other? Is there anything specific to the Pacbio assembly but not Illumina assembly, and vice versa?
- what sequence features define the uncorrected Miniasm in particular?

## Step 5: Further exploration of your genome assemblies

Now that we have compared and contrasted our initial genome assemblies, we would now want to think about ways of improving them to make them more contiguous – each of our assemblies is still some way off being a single sequence, i.e., a single chromosome. One way would be to generate additional, complementary data that might be used to scaffold the existing contigs together to create much longer sequences. Approaches include generating mate-pair libraries, or alternate long range sequencing technologies such as Nanopore, optical mapping, or HiC to name a few. However, this is obviously outside the scope of this tutorial.

Our assemblies are currently represented in a FASTA file; each individual sequence is presented separate from each other, and there is no information that links each sequence to each other. However, in generating the assembly, the assembler catalogs overlaps between sequences with the aim of joining / extending existing sequences; if there is a single overlap, a join is made, however, if there are two or more overlaps between which the assembler cannot confidently make a decision, it will not make the join and report multiple sequences. These multiple paths between sequences might be due to genetic variants, haplotypes, repeats etc. Importantly, some assemblers record these multiple paths in a structure known as a **genome graph**. These genome graphs are composed of:

- nodes – these are the individual sequences presented in the FASTA
- edges – these link two nodes together
- paths – describes the linking of nodes via edges to form a longer sequence

We will use the tool **Bandage** (<https://rrwick.github.io/Bandage/>) to visualise the genome graphs produced by miniasm and spades assemblers, and demonstrate how to extract extended sequences from these graphs to extend your genome assemblies. We will compare your new sequence against the reference using the web tool, **Genome Ribbon** (<http://genomeribbon.com/>), which is similar to ACT, but is more suited for larger genomes.

### Tasks

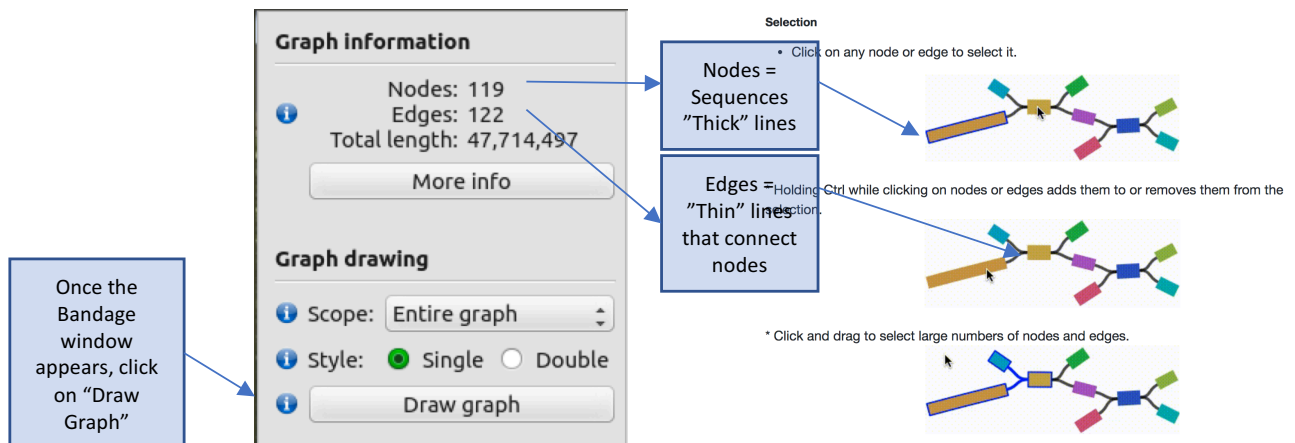
- visualise and compare the Pacbio miniasm and Illumina Spades genome graphs
- using the Pacbio miniasm graph, construct a path through the graph, making a new sequence
- compare your new sequence against the reference

```
# go to the working directory
$ cd /home/manager/Module_6_helminth_denovo_assembly/step_5

# load Pacbio miniasm genome graph into Bandage. This file was
made during the miniasm assembly

$ bandage load SM_V7_chr4.miniasm.gfa

# use Bandage to explore the graph
```



The genome graph is sorted by size, with the largest sequence(s) in the top left corner, which get progressively smaller down the page. The colours represent difference sequences in the genome, which are called "nodes" in the graph. These are joined in some cases by thin black lines called "edges", which where possible, describe the relationship between sequences. Graphs therefore provide an additional level of detail over the genome sequence alone; each node is represented as an independent sequence in a fasta file, however, in a genome graph, alternate paths that connect nodes can be visualised. These alternate paths typically break assemblies, as the assembler cannot reliably choose a single path to extend the assembly.

Explore the genome graph, zooming into some of the groups of sequences. Some consist of a single node, i.e., a single contig sequence, with no relationships to other sequences, whereas other are more complex, in which larger nodes may be connected by two or more alternate nodes.

```
# Once finished, load the Illumina (SPADES) graph (made during
the Illumina assembly) into Bandage, and compare.

$ bandage load SM_V7_chr4.spades.gfa

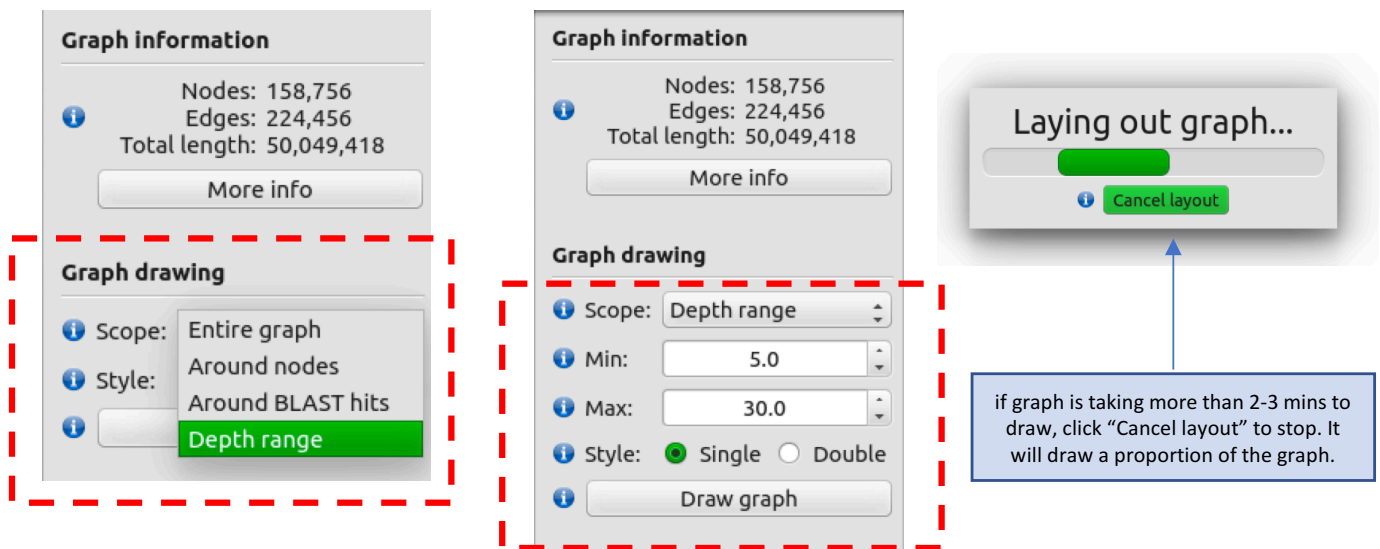
# Note that this file will take longer to load than the
previous one.
```



Once Bandage loads, we are going to limit the amount of data displayed to enable faster viewing.

1. Under the “Graph drawing” subheading, select “Depth Range” in the “Scope” drop-down.
2. Set the “min” to 5 and “max” to 30
3. Click on “Draw Graph”

NOTE: if the graph has not appeared after 2-3 mins, click on “Cancel layout”, after which the graph should appear shortly.



The Illumina genome graph will look *quite* different to the Pacbio miniasm graph.

**Zoom out** completely to give you a sense of the scale of the graph.

- If you recall from the “assembly-stats” output in step 3, the Illumina assembly was in many more pieces than the Pacbio assemblies. The graph reflects this by the large number of unique nodes present.

Move to the top left hand corner containing the largest collection of sequences, and take a closer look by **zooming in**.

- The graph also demonstrates the reason for the fragmentation in the Illumina assembly; the relationships between nodes is often much more complex with many more paths present, due to non-unique edges between sequences.
- You should also see in this this graph (if you look closely) that there are many paths that terminate suddenly. If you look closer still at the direction of the edges connecting the nodes, some look to turn around, resulting in a duplication of the sequence. This might be due to repetitive or haplotypic sequences in the assembly.



Lets now compare your new sequence back against the reference to see how you have done.

1. Use *nucmer* to compare the reference sequence and your new path\_sequence

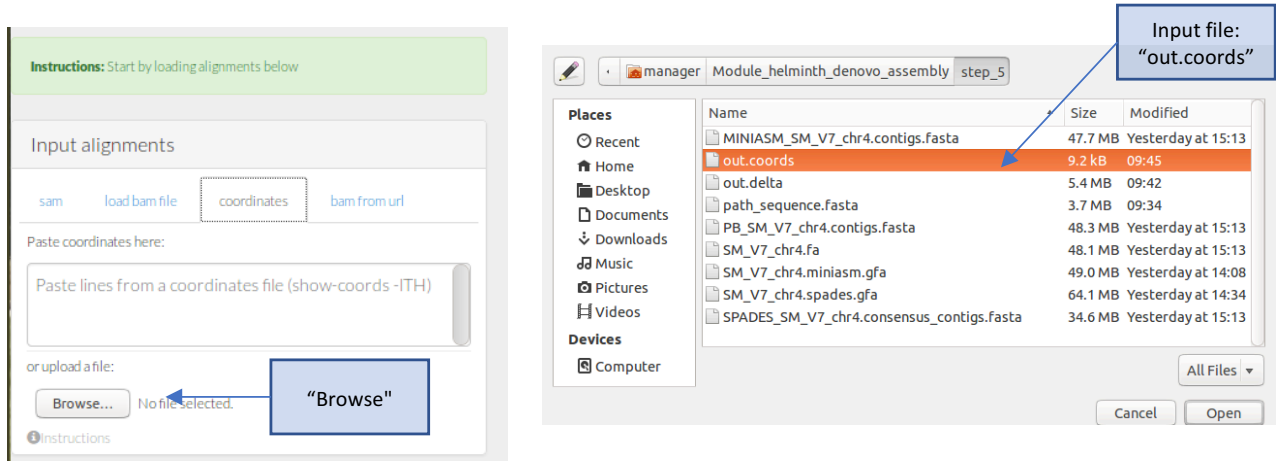
```
# Compare your new sequence with the reference using nucmer
and show-coords

$ nucmer -maxmatch SM_V7_chr4.fa path_sequence.fasta

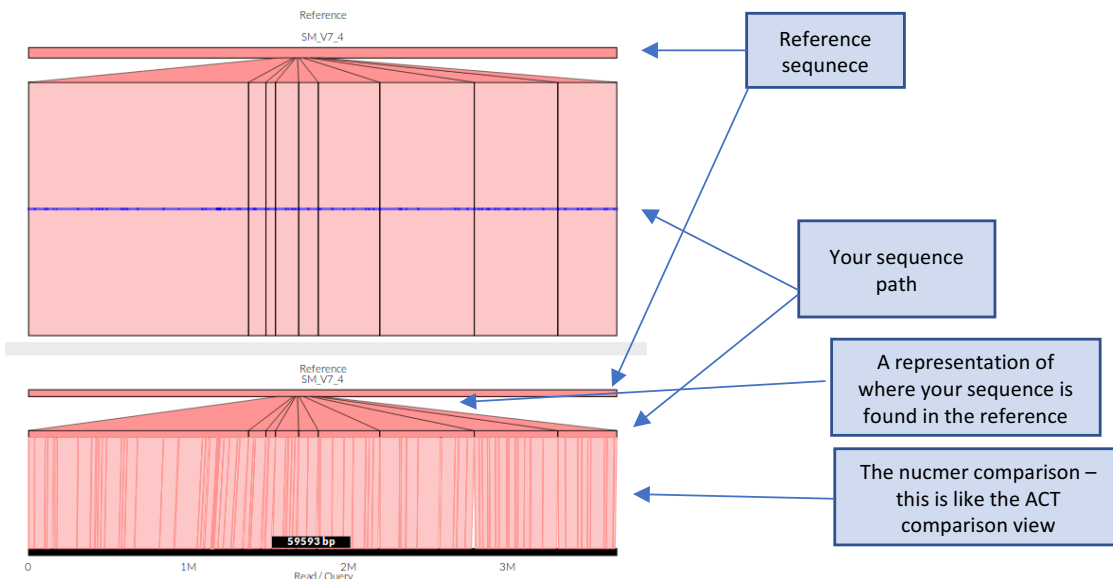
$ show-coords -lTH -L10000 out.delta > out.coords

# once completed, load Genome Ribbon (genomeribbon.com) in a
web browser.
```

2. Load your data into Genome Ribbon. Scroll down the page until you see the “Input alignments” window, select the tab “coordinates”, and then click “Browse”
3. A finder window will appear – select your “out.coords” file and click “Open”



4. The comparison between your sequence and the reference should now appear.



**Questions you should be asking:**

- what are the main differences between the Pacbio and Illumina genome graphs?
- what is the length of your new sequence?
- how did your new sequence compare to the reference? Was it syntenic?

## Summary

This module aimed to introduce you to some of the concepts involved in eukaryotic genome assembly, from the QC of your raw data, through to assembly, validation and improvement. In reality, eukaryotic genomic assembly is a challenging task, often requiring multiple datasets and tools, each with their own strengths and weaknesses. It is important to understand or at least be aware of these differences to maximise the completeness of the assembly. Hopefully it is clear from the examples that long read technologies such as Pacbio significantly improve the contiguity of assemblies over Illumina-only assemblies.

# References / Links

- Assemblytics
  - Web: <http://assemblytics.com/>
  - Paper: <https://doi.org/10.1093/bioinformatics/btw369>
- Bandage
  - Web: <https://rrwick.github.io/Bandage/>
  - Paper: <https://academic.oup.com/bioinformatics/article/31/20/3350/196114>
- Canu
  - Web: <https://canu.readthedocs.io/en/latest/>
  - Paper: <https://genome.cshlp.org/content/27/5/722.full.pdf+html>
- FastQC
  - Web: <https://www.bioinformatics.babraham.ac.uk/projects/fastqc/>
- Genome Ribbon
  - Web: <http://genomeribbon.com/>
  - Paper: <https://www.biorxiv.org/content/early/2016/10/20/082123>
- Genome Scope:
  - Github: <https://github.com/schatzlab/genomescope>
  - Paper: <https://doi.org/10.1093/bioinformatics/btx153>
- Illumina
  - Web: <https://emea.illumina.com>
- Jellyfish
  - Web: <http://www.genome.umd.edu/jellyfish.html>
  - Paper: <https://doi.org/10.1093/bioinformatics/btr011>
- Kraken
  - Web: <https://ccb.jhu.edu/software/kraken/>
  - Paper: <https://doi.org/10.1186/gb-2014-15-3-r46>
- Minimap2 / Miniasm
  - Github: <https://github.com/lh3/minimap2>
  - Paper: <https://academic.oup.com/bioinformatics/article/32/14/2103/1742895>
- MultiQC
  - Web: <http://multiqc.info/>
  - Paper: <https://doi.org/10.1093/bioinformatics/btw354>
- Nucmer
  - Web: <http://mummer.sourceforge.net/>
  - Paper: <http://mummer.sourceforge.net/MUMmer3.pdf>
- Pacbio
  - Web: <https://www.pacb.com>
- Spades / dispades
  - Web: <http://cab.spbu.ru/software/spades/>
  - Paper: <https://dx.doi.org/10.1089%2Fcmb.2012.0021>

# Module 7

# Genome Annotation and Differential Expression

## Introduction to Genome Annotation

One of the key goals of producing a draft genome is to define the genes encoded in it. This is the first step in answering many important questions. How many genes does this organism have? What metabolic pathways are present? Are there novel gene families encoded in the genome? Subsequent uses of the genome, such as proteomics and transcriptomics experiments rely on accurate gene models. We concentrate here on protein coding genes, however one would also try to identify non-protein coding RNAs such as tRNAs, rRNAs, miRNAs, transposons etc.

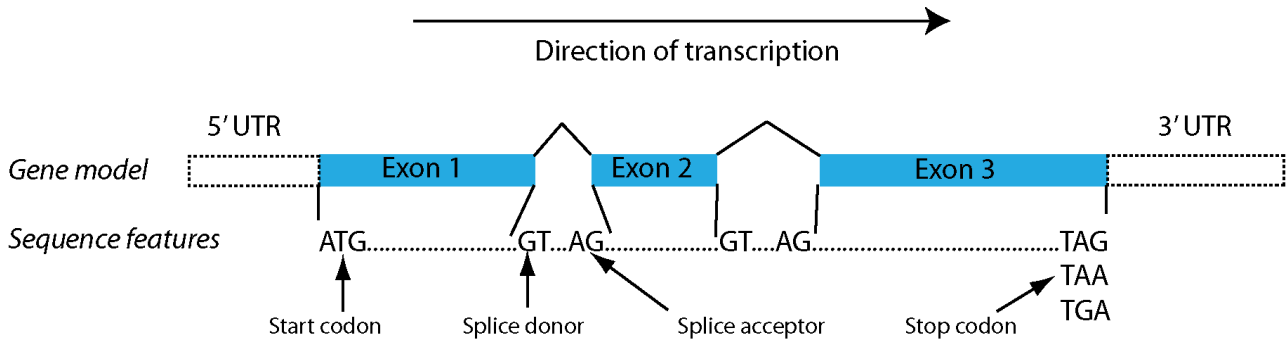
Producing accurate **gene models** is just as hard as producing a good assembly. You have seen that one way of producing a set of gene models is to transfer them from a closely related organism. However, if there is not a closely related genome, or the most closely related genome is not well annotated, this may not be an option. Furthermore, even if there is a closely related, well annotated reference genome as in the case of the malaria parasite - *Plasmodium falciparum* strain 3D7 and *P. falciparum* strain IT, there may be regions of your genome of interest which are not syntenic to the reference. Indeed, this is the case here, as the subtelomeric regions of *Plasmodium* chromosomes are highly variable and cannot be used to transfer gene models, even between strains of the same species.

When there is no reference genome, or for regions which are not syntenic to the reference, we can use **ab initio gene finding** methods. These identify genes based on properties of the genome sequence independent of whether they show homology to known genes. They can be trained and it is common to identify the most well conserved genes by homology, then to train an *ab initio* gene finding algorithm using these well conserved genes so that it can learn what a gene looks like in the particular genome you are interested in. In this module we will use the program **Augustus** to predict gene models *ab initio*.

Another approach to identifying gene models is to determine those regions of the genome which are transcribed. Prior to the advent of second generation sequencing technologies Sanger capillary sequencing was used to sequence mRNAs and generate Expressed Sequence Tags (ESTs). These could be used to identify the most highly expressed genes and improve some gene models. With second generation sequencing technologies we are able to sequence mRNA transcripts from essentially all the genes which are expressed. This is known as **RNA sequencing** or RNA-seq (Mortazavi et al., 2008; Wang et al, 2009) and the resulting data provides incredible resolution of gene structure. This method is not biased by which genes are present in previously sequenced genomes (as with RATT) or by how much their structure reflects that of other genes in the genome (as with Augustus), but rather by how highly expressed they are and how extensively we sequence the transcriptome.

In this module you will generate a set of gene models *ab initio* using the gene prediction tool Augustus. It has been trained using the highly accurate, manually curated gene models of *P. falciparum* 3D7. These gene models will be used to fill in those regions of the *P. falciparum* IT assembly which could not be annotated using RATT because they are not syntenic to *P. falciparum* 3D7. You will then map RNA-seq data to your assembly and use this to improve the gene models. There will be inaccuracies from the RATT transfer due to technical error and due to real differences in the gene structures. There will be inaccuracies in the Augustus predictions due to gene models which do not follow the expected pattern of a *Plasmodium* gene and due to inaccuracies in the genome assembly. These can be addressed using the RNA-seq mapping.

Below is a model of the eukaryotic protein-coding gene highlighting features relevant to their annotation. Note that compared to bacteria, eukaryotic genes frequently have multiple exons, separated by un-translated introns and 5' and 3' Un-Translated Regions (UTRs) which do not encode part of the protein sequence.



**Figure showing the key features of a eukaryotic gene. The exact DNA sequence of the splice sites may vary.**

## Module Summary

1. Generating an initial set of gene models (merging RATT and Augustus)
2. Mapping RNA-seq data to a reference
3. Viewing RNA-seq mapping in Artemis
4. Correcting gene models by hand
5. Automatically generating gene models based on RNA-seq data
6. Using RNA-Seq to improve annotation



# 1. Generating an initial set of gene models

## A. Generate gene models *ab initio*

The *ab initio* gene prediction algorithm Augustus has already been trained with gene models from *Plasmodium falciparum* 3D7. You will now run it on your *Plasmodium falciparum* IT strain chromosome to predict a set of gene models.

Navigate to the module 7 data directory. On the command line, type:

```
augustus --species=pfalciparum IT.genome.fa > augustus.gtf
```

The file `augustus.gtf` now contains your predicted gene models

Next we are going to convert the Augustus GFF to EMBL format. On the command line, type:

```
cat augustus.gtf | augustus2embl.pl > augustus.embl
```

Although Artemis can display gff files, the visualization is better for embl files (if you want you can also try loading the gff file into Artemis).

It is typically much more challenging to train an *ab initio* gene finder than to run it. Most *ab initio* gene finders require a set of very accurate models to train them, which may be predicted from highly conserved genes, RNA-Seq and typically manual curation of a few hundred gene models. The chosen training set can influence which types of gene models get better or worse predictions, so should be carefully chosen.

## B. Examine gene model predictions

We will open some gene models which we transferred from *P. falciparum* 3D7 using the tool RATT. We can compare them to the gene models predicted *de novo* by Augustus.

On the command line, type:

```
art Transfer.ordered_Pf3D7_05.final.embl &
```

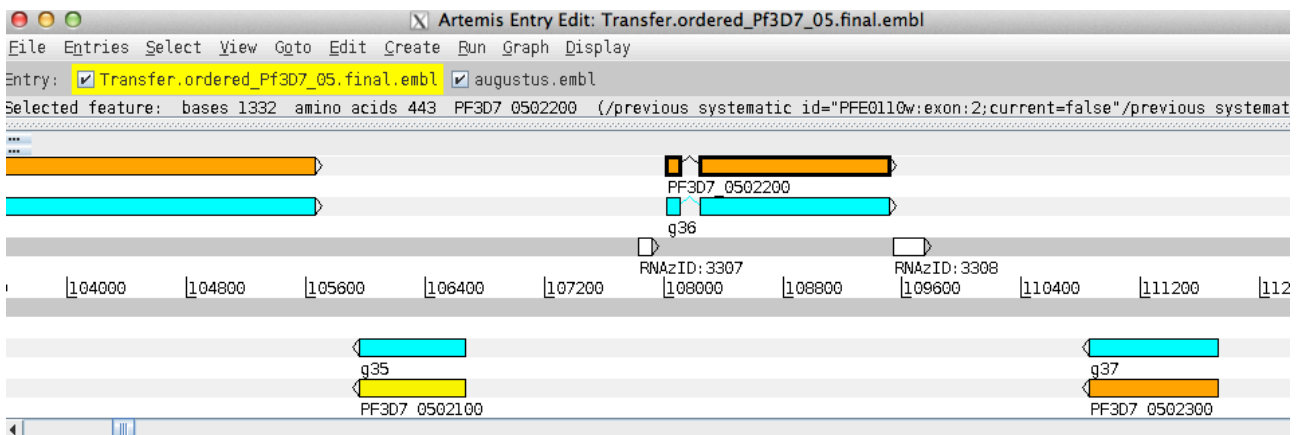
Load in Augustus models. In Artemis:

File -> Read An Entry and select the file `augustus.embl`.

Right click in genome window, select “One Line Per Entry”.

The transferred models have different colours and annotation, while the newly predicted genes have the default colour (blue), and are named `g1`, `g2`, `g3`...

Check out different loci in this chromosome to appreciate the difference in the annotation files



Have a look at the gene PF3D7\_0515600. Does Augustus perform better than RATT? What has gone wrong with the prediction? Go to the gene by Goto -> Navigator -> “Goto Feature With Gene Name”. Tip: you don’t have to type in the complete gene name.

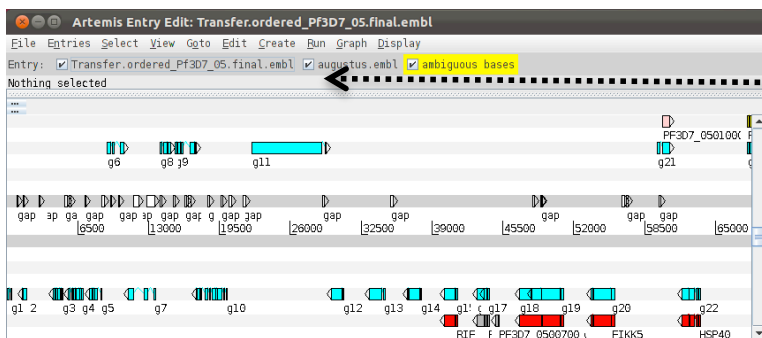
The screenshot shows the Artemis Navigator interface. At the top, a track displays gene models for PF3D7\_0515600 and q171. Below this, a genomic scale from 653600 to 662400 is shown. A search dialog box titled "Artemis Navigator" is open, with "Goto Feature With Gene Name" selected and "0515600" entered in the search field. Other search options like "Goto Base", "Goto Feature With This Qualifier Value", and "Find Base Pattern" are also visible. The dialog box includes checkboxes for "Overlaps With Selection", "Forward Strand", "Reverse Strand", "Search Backward", "Ignore Case", and "Allow Substring Matches".

Look at the gene PF3D7\_0504800. Perhaps due to the low GC content of this region Augustus decided that it is intronic (Graph -> GC content (%)). How could you verify the prediction?



## C. Combine RATT and Augustus gene models

Augustus is able to predict gene models in regions of the IT genome which have no synteny with the 3D7 genome, principally the subtelomeres. However, on the whole, the RATT-transferred gene models ought to be more accurate because the IT and 3D7 genome are otherwise very similar. Therefore it is perhaps most useful to add in only those Augustus gene models which do not overlap RATT-transferred models. To do this, the Augustus gene models must be converted into EMBL format.



Unselect the  
“augustus.embl” entry.

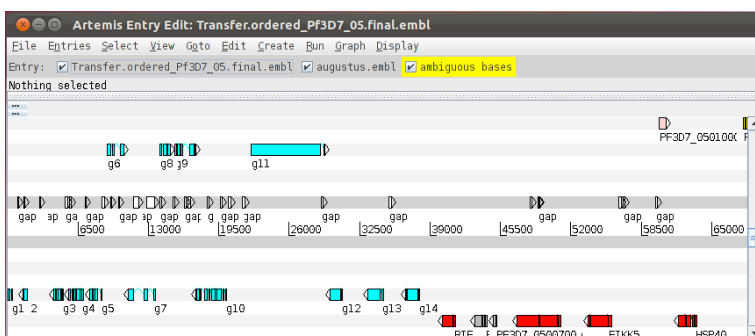
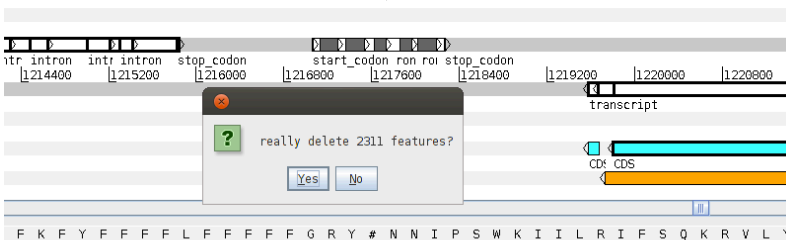
Go to Select -> All CDS features

Re-select the  
“augustus.embl” entry.

Go to Select -> Features  
Overlapping Selection

n.b. deselect any augustus  
(blue) gene models which you  
have edited and want to keep  
by shift-clicking that model.

The Augustus gene models which  
overlap RATT models should now  
be selected. Delete them! Edit ->  
Selected Features -> Delete



If you want to keep any augustus gene model you have edited, then delete the overlapping RATT model. How does the annotation look? How many extra gene models do we have compared to using the RATT-transferred ones alone? Use the function View -> Overview to see some stats. In the next section we are going to show how RNA-Seq data can be used to correct gene models.

We need to save the merged annotation:

File -> Save An Entry As -> New File -> augustus.embl

Save to ... “augustus\_keep.embl”

Now merge it into the original file:

File -> Read Entry Into -> Transfer... Transfer.ordered\_Pf3D7\_05.final.embl

Select a file ... “augustus\_keep.embl”

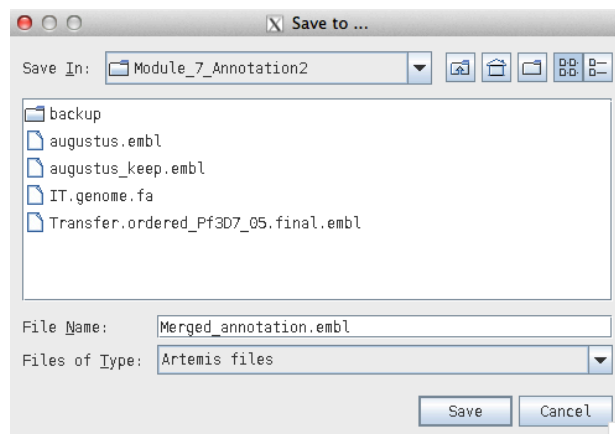
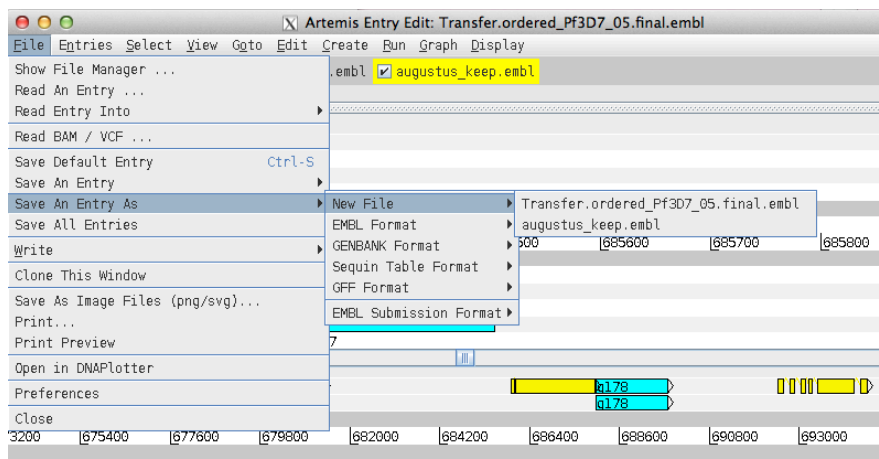
And now save the merged file:

File -> Save An Entry As -> New File -> Transfer.ordered\_Pf3D7\_05.final.embl

Save to ... “Merged\_annotation.embl”

This file “Merged\_annotation.embl” can now be used as a basis for further annotation.

Always remember to save your work!



Gene models can also be merged/overlapped/removed bioinformatically using custom scripts or the free command-line software BEDtools <http://bedtools.readthedocs.org/en/latest/> . The most convenient formats for manipulating annotation are GFF/GFF3/GTF and BED.

## D. Functional annotation

For those gene models transferred by RATT, you will have a range of functional information which will help you identify the types of genes present in your genome. This functional information has been manually curated for *P. falciparum* 3D7 based on the literature. For those genes predicted *de novo* by Augustus there is no such information. It is beyond the scope of this module to present a solution for assigning functional annotation for all these extra genes, but you can annotate a few of them yourself. Product calls for genes are usually defined by looking for orthologues or best BLAST hits in other organisms for which a gene has been annotated with a useful name. Many annotation databases exist for different purposes; Pfam for functional domains [pfam.sanger.ac.uk](http://pfam.sanger.ac.uk), Gene Ontology for GO-terms <http://www.geneontology.org/> and KAAS for enzymes [www.genome.jp/tools/kaas/](http://www.genome.jp/tools/kaas/). There are often specialized databases for specific classes of genes/organisms you might be particularly interested in.

For *Plasmodium* annotation, the best place to look is PlasmoDb, a large resource of comparative genomics data for these species.

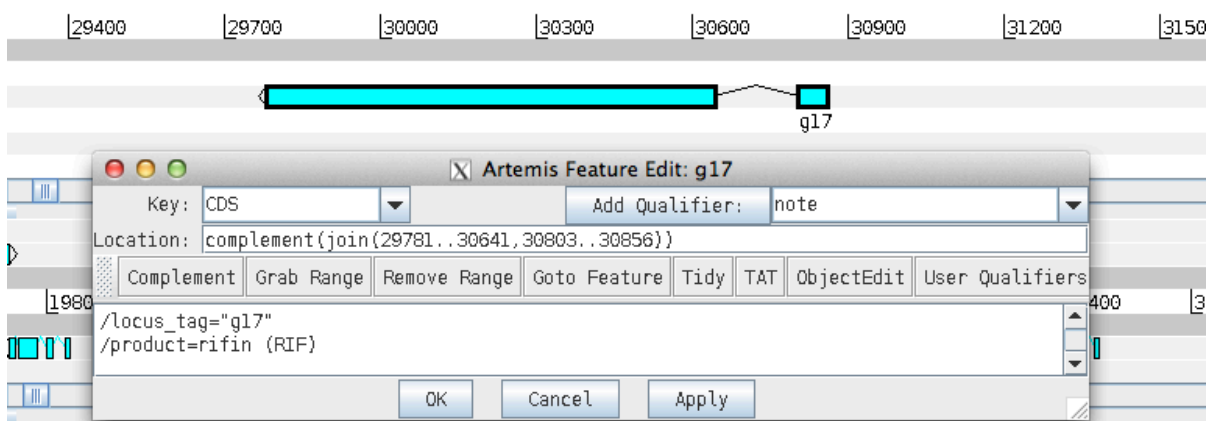
Right click on an augustus gene model (blue ones), View -> Amino Acids of Selection as Fasta -> Ctrl-A -> Ctrl-C

In a web browser, navigate to <http://plasmodb.org/>

Under the Tools menu, select BLAST. In the web form, select “Proteins”, “blastp”, select all Target Organisms, paste in your sequence (Ctrl-V) and “Get Answer”. It is essential that your BLAST-search parameters match the search you are trying to do (protein versus protein in this case).

If the top hit has a good E-value (e.g. less than  $1e-20$ ), select it and copy the description. Then select the gene model, press “Ctrl+e”, and add a new line as below. In the example below Augustus prediction g17 is a gene from the rifin (RIF) family.  
/product=“rifin (RIF)”

Click OK to save the annotation.



## 2. Mapping RNA-Seq data to a reference

We will use the program TopHat, part of the Tuxedo suite, to map RNA-Seq reads to our reference genome, chromosome 5 of *P. falciparum* IT. In this case we are mapping single-end reads generated from RNA extracted during the blood stage of malaria.

TopHat requires an index of the reference. Create the index:

```
bowtie2-build IT.genome.fa IT.genome
```

Now run TopHat (n.b. this may take several minutes depending on computer):

```
tophat -o 30h_map -I 10000 IT.genome
blood_stage_30h.fastq.gz
```

To read the BAM file:

```
samtools view 30h_map/accepted_hits.bam | head
```

TopHat will generate several files in the new “30h\_map” directory you have specified. The most important is `accepted_hits.bam`. This contains the mapping. Briefly read through the file if you like. For some reads, there are Ns in the Cigar line (column 6, see below). What do these mean?

```
/lustre/scratch108/parasites/mz3/Malawi2014/Module_7_Annotation2 >samtools view 30h_map/accepted_hits.bam |
M | head
SOLEXAWS1:1:6:103:907:1848.F 0 Transfer.ordered_Pf3D7_05.final 47852 50 7M568N69M *
CTATCACATTATCTTTCTTTTCATTCTTATTATTCTTTTTTTTACTCTTTTTTAACTTTTTTTTACTTCTTT
ABBCCA@ACCCAABCCB?BCBAACBA?C
A<A7@3@CCBC@;BB/+A<CCACBB7@7=@9@BC AS:i:-8 XN:i:0 XM:i:1 X0:i:0 XG:i:0 NM:i:1 MD:Z:60C15 YT:Z
:i:1
SOLEXAWS1:1:6:110:10:502.F 0 Transfer.ordered_Pf3D7_05.final 47852 50 7M568N69M *
CTATCACATTATCTTTCTTTTCATTCTTATTATTCTTTTTTTTACTCTTTTTTAACTTTTTTTTACTTCTTC
BCCCCCCCCCCCCCCCCCCCCBCCCCC
CCACCCCBCCBCB@CCCCBCCCCB=CCAC@% AS:i:-11 XN:i:0 XM:i:2 X0:i:0 XG:i:0 NM:i:2 MD:Z:60C14T@
:A:+ NH:i:1
SOLEXAWS1:1:6:113:1017:1915.F 0 Transfer.ordered_Pf3D7_05.final 47852 50 7M568N69M *
CTATCACATTATCTTTCTTTTCATTCTTATTATTCTTTTTTTTACTCTTTTTTAACTTTTTTTTACTTCTTT
9CCCCCCCCCCCCCCCCCCC>CCCCC
CCCCCCCCCCCCCCCCCCCCCCCCCCCCB@ACC AS:i:-9 XN:i:0 XM:i:1 X0:i:0 XG:i:0 NM:i:1 MD:Z:60C15 YT:Z
:i:1
SOLEXAWS1:1:6:114:1418:1525.F 0 Transfer.ordered_Pf3D7_05.final 47852 50 7M568N69M *
CTATCACATTATCTTTCTTTTCATTCTTATTATTCTTTTTTTTACTCTTTTTTAACTTTTTTTTACTTCTTC
BCCBC@BCBCCBCACCCCCCCCCC
#####
```

We have to index the bam using SAMtools, in order to view the data in Artemis.

```
samtools index 30h_map/accepted_hits.bam
```

The output index file is called `30h_map/accepted_hits.bam.bai`

The reads you just mapped do not cover the whole genome, so that the mapping would go faster. We will instead view the much larger pre-computed file which does cover the whole genome:

```
blood_stage_30h.bam
```



### 3. Viewing RNAseq mapping in Artemis

We will now examine the read mapping in Artemis using the BAM view feature.

If you have closed the Artemis window, then first type:

```
art Merged_annotation.embl &
```

In Artemis, highlight gaps in the assembly which might mislead you about the meaning of the RNAseq data:

Create -> Mark Ambiguities

Load the BAM file:

File -> Read BAM / VCF -> Select, “blood\_stage\_30h.bam” -> “OK”

This opens a new window at the top. Right click on the BAMview window, select Graph -> coverage.

**BAM view**

Right click on the coverage plot and select Options... Set the window size to 1 to see the exon boundaries (You have to disable “Automatically...”)

Examine the exon boundaries of a couple of genes. How well are splice sites identified by gene predictors vs. RNA-Seq?

What do the grey lines in the middle of some of the mapped reads mean? Right click on one of these reads, select “Show details of” and examine the cigar string.

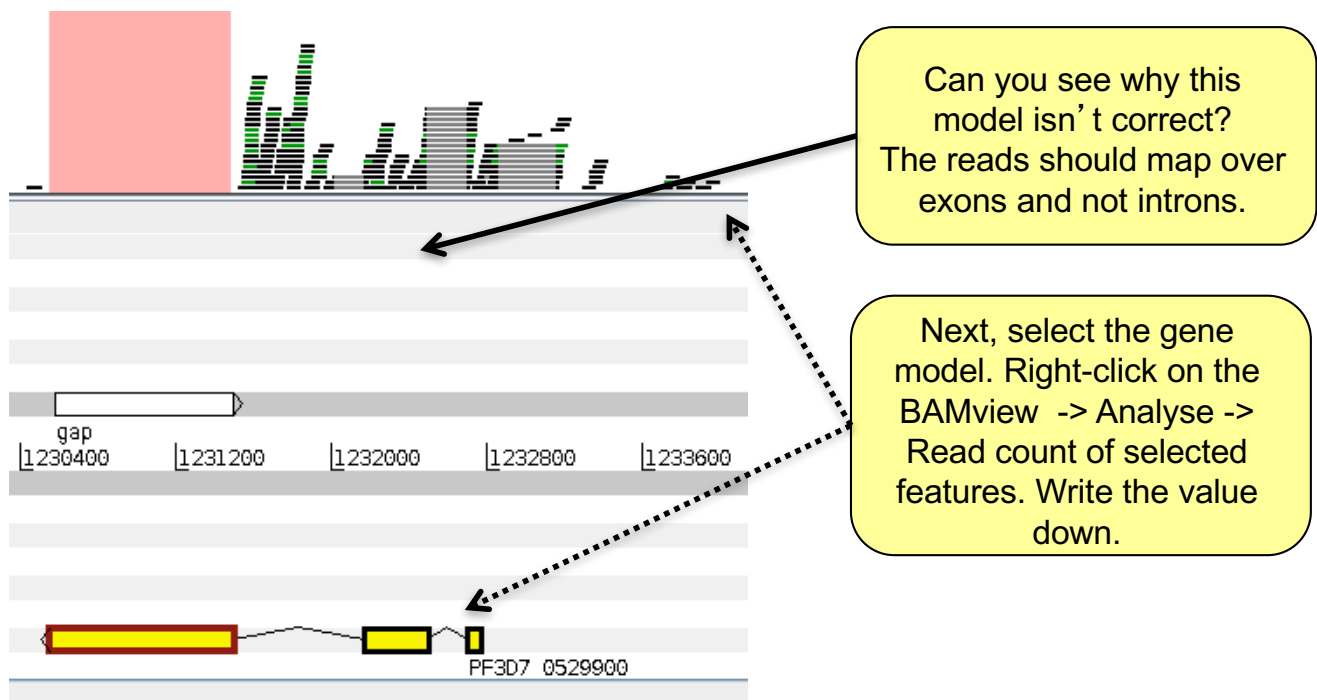
## 4. Correcting gene models by hand

Scroll along the chromosome and examine the read coverage. How well does it correlate with the gene models? Notice how different genes have different depths of coverage. Why do some genes have little or no coverage? What does this mean for annotation?

Why do some reads map where there are no genes?

Scroll along the chromosome. Can you see any gene models which might be incorrect based on the RNAseq data? Find one and correct it. PF3D7\_0529900 is a good example.

Bonus question: Can you figure out why RATT got this gene model so wrong!



Now correct the last exon. You can move gene-boundaries by left-clicking at the edge of a gene-model and dragging it to the right position. If you are very zoomed out and the model is hard to catch, try shift + left-click instead. You can add exons/introns to a model by selecting the model and then clicking “e” to open the Feature editor. Left-click and highlight the region on the genome where you want to add an exon. Then switch to the Feature editor box again, and click the button “Grab range” or “Remove range”. Click “Apply” and you can straight away see the gene-model change. Under Edit -> Trim../Extend... there are some useful short-cuts for adjusting gene-boundaries. Look again at read counts/RPKM values. Have they changed?  
**n.b. Don't forget to save any changes you make!**

**Optional:** Is the start of the gene model correct? There seem to be spliced reads confirming another exon.

## 5. Automatically generating gene models based on RNA-seq data

Cufflinks is another program in the Tuxedo suite. It can be used to generate gene models based on the RNA-seq mapping. Rather than fix each gene model by hand, we could replace them with RNA-seq based predictions if these are better.

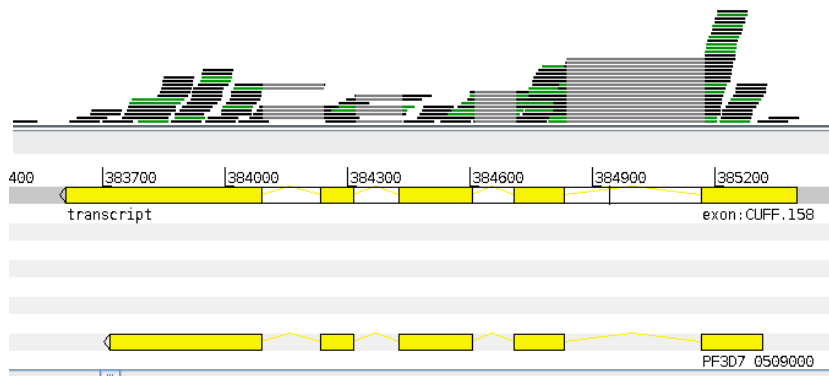
Run Cufflinks:

```
cufflinks blood_stage_30h.bam
```

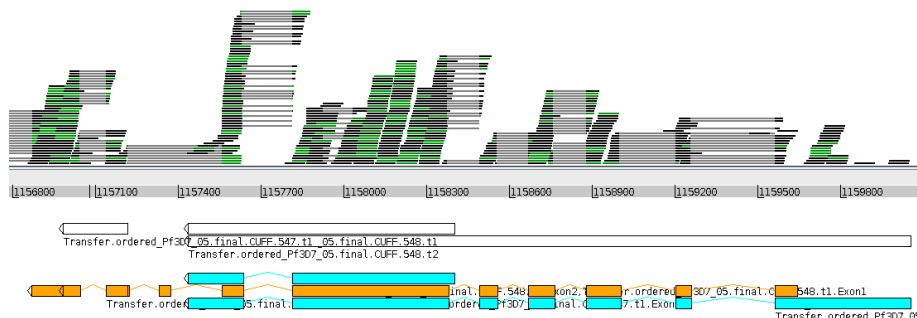
The key results file from Cufflinks is transcripts.gtf. When the output file transcripts.gtf has been created, change the format to display better in Artemis:

Read this into Artemis and compare the results to the RNAseq coverage plots and to the existing gene predictions. In Artemis, File -> Read An Entry, select “Files of type: All files”, select “transcripts.gtf”

Look through the annotation. Does cufflinks confirm the Augustus predictions? What can we say where a gene model has no coverage? How can we remedy this?



Why are the Cufflinks predictions often longer than the gene model predictions? Remember we are examining a eukaryote. Why does this make it difficult to incorporate these models directly into our gene set? Would this explain the spliced reads of the gene PF3D7\_0529900 on page 11?



**Optional:** Can cufflinks help us to find alternative splicing? Maybe check the gene PF3D7\_0527600 – Cufflinks has predicted two splice forms for this gene. Right-click in the genome window and choose “Feature Stack View” to see both splice forms.

## 6. Using RNA-seq to improve predictions

Perhaps the best option would be to use the RNA-seq to guide the *ab initio* predictions? Augustus allows you to create hints-files from RNA-seq. You can read more about how to create your own hints on augustus help pages: [augustus.gobics.de/binaries/readme.rnaseq.html](http://augustus.gobics.de/binaries/readme.rnaseq.html)

On the command line, you first have to copy a suitable configuration-file from the augustus folder:

```
cp
/usr/local/augustus.2.7/config/extrinsic/extrinsic.M.RM.E.W.cfg .
```

Make hints from the bam-file. On the command-line, type:

```
bam2augustusHints.pl blood_stage_30h.bam > hints.introns.gff
```

Now predict new models using RNA-seq hints, you'll notice it takes a bit longer than running without hints, expect a few minutes. On the command-line, type (as one line).

```
augustus --species=pfalciparum
--extrinsicCfgFile=extrinsic.M.RM.E.W.cfg
--hintsfile=hints.introns.gff IT.genome.fa >
augustus.hints.gtf
```

Convert the Augustus GFF to EMBL format, so you can look at it in Artemis. On the command line, type:

```
cat augustus.hints.gtf | augustus2embl.pl >
augustus.hints.embl
```

Load the new predictions into Artemis like you did before, and compare this prediction to your earlier prediction without hints, to see which one you think is better. Try for instance to look at model PF3D7\_0523600 and PF3D7\_0528500.

Tip: if you want to see the difference between the models more clearly, you can in Artemis un-tick all files except for augustus.hints.embl at the bar at the top, choose Select -> "All CDS features". Click Edit -> "Qualifier of selected feature(s)" -> Change. In the drop-down menu in the pop-up box choose "colour", and then click "Insert qualifier:". Change the text in the box to /colour=3 and click "Add". All your models predicted using hints are now green.

**Optional 1:** Learn more augustus; a) learn how to train augustus with your favorite species from the online manual, b) check out the useful scripts that come with augustus; /usr/local/augustus.2.7/scripts/ , c) What does the Augustus option --alternatives-from-evidence do?

**Optional 2:** Try to look at all the differences between augustus predictions with and without hints: 1. using programming, 2. in Artemis. How would you choose which predictions are the best? If you had to write a script to automatically choose between the models, what would that script contain?

## Key aspects of genome annotation

### Quality

The better the genome prediction is, the better the gene-models will be. If the genome is miss-assembled that can lead to partial or chimeric gene-models. The gold standard for gene-models is manual gene-model curation, but for draft genomes there often not resources available to do this. So for draft genomes you may have to accept that you will not have a perfect set of genes. Ten years of annotating the malaria genome by hand using all possible lines of evidence has not resulted in a perfect annotation (although it is a very good one)! How do you think the quality of the gene-models affect the analysis you can do, and the conclusions you can draw?

### Several lines of independent evidence are best

Predicting gene-models, you will find that one of the hardest things to do is to choose which set of models are the best; all methods are good at some types of genes and bad at others. Augustus has a built-in quality check, in which you can compare your training models with your predicted models. Use a variety of prediction approaches, as you have done here, to capture as many of the genes as possible and to improve their accuracy. It is always a good idea to try different programs for any particular problem in computational biology: if they all produce the same answer you can be more certain it is correct. In the case of gene model predictions they will frequently disagree. If several predictions are of similar high quality, perhaps the best option is to combine different sets of gene using tools such as Jigsaw (Allen & Salzberg, 2005) and EVM (Haas et al., 2008)?

### Over-prediction

There is a balance to strike between having almost all the genes and lots of erroneous ones as well, or to miss some genes but have relatively few incorrect ones. It may be important to find as many of the real genes as possible. However once you have published an erroneous model to the public sphere, for instance by submitting it to GenBank, it can be very hard to retract it later, and it may cause problems for other people using that gene for their analyses.

### Bacteria are simpler

If you work on bacteria you will encounter fewer problems with accurately predicting gene models as they almost always have single-exon genes. The program Glimmer3 (Delcher et al., 1999) is an alternative to Augustus for *ab initio* gene prediction in bacteria, but since version 2.7 Augustus has improved its prediction methods for bacterial genomes. Another alternative for both gene prediction and functional annotation for bacteria is Prokka  
[www.vicbioinformatics.com/software.prokka.shtml](http://www.vicbioinformatics.com/software.prokka.shtml)

### Alternative splicing

Many genes in more complex organisms have several alternative splice-forms, including/excluding different UTRs and exons. Predicting alternative splicing is much harder than predicting a “canonical”; most complete, gene-model (like the ones you have just predicted). It is currently only possible to predict alternative transcripts reliably for model genomes which already are very well assembled and annotated, and have a wealth of supporting evidence (like human and *C. elegans*).

## Key aspects of RNA-seq mapping

### Non-unique/repeat regions

A sequence read may map equally well to multiple locations in the reference genome. Different mapping algorithms have different strategies for this problem, so be sure to check the options in the mapper. A low GC content, such as in *Plasmodium falciparum* (81% AT) means that reads are more likely to map to multiple locations in the genome by chance.

### Insert size

When mapping paired reads, the mapper (e.g. TopHat) takes the expected insert size into account. If the fragments are expected to on average be 200bp, and the reads are 50bp, then the insert between the paired reads should be ~100bp. If the paired reads are significantly further apart than expected, we can suspect that the reads have not mapped properly and discard them. Removing poorly mapping reads can produce a more reliable mapping.

### Spliced mapping

Eukaryotic mRNAs are processed; after transcription introns are spliced out. Therefore some reads (those crossing exon boundaries) should be split when mapped to the reference genome sequence in which intron sequences are still present. TopHat is one of few mappers which can split reads while mapping them, making it very suitable for mapping RNA-seq. Beware that TopHat cannot recognize donor and acceptor splice-sites so it will split reads only based on optimizing the mapping, and you will occasionally see a couple of bases of the read having ended up on the wrong side of the intron.

### Alternative mappers

Alternative short read mappers which do not split reads include SOAP (Li et al., 2008b), SSAHA (Ning et al., 2001), BWA (Li et al., 2009) and Bowtie2 (Langmead B, Salzberg S. 2012), SMALT (Ponstingl, unpublished). All of these may be appropriate for bacterial RNA-seq. Where introns are an issue RUM (Grant et al., 2011) is one alternative to TopHat (Trapnell et al., 2009).

New tools for mapping sequence reads are continually being developed. This reflects improvements in mapping technology, but it is also due to changes in the sequence data to be mapped. The sequencing machines we are using now (e.g. Illumina HiSeq, 454 GS FLX etc) will perhaps not be the ones we are using in a few years time, and the data the new machines produce may not be best mapped with current tools.

### Beware of the genes!

In spite of our very best efforts, it is not always possible to predict genes accurately. There are many phenomena which can throw both automatic and manual predictions off track. For instance (but not limited to): seleno-proteins containing “stop-codons” as part of the coding sequence, polycistronic genes, splice-leader trans-splicing, long non-coding RNAs, repetitive genomic regions and pseudogenes. Before publishing, it is always a good idea to try to estimate how correct the models are, and doing some sanity-checking of the gene-models.



# Differential Expression

## Introduction to Differential Expression analysis

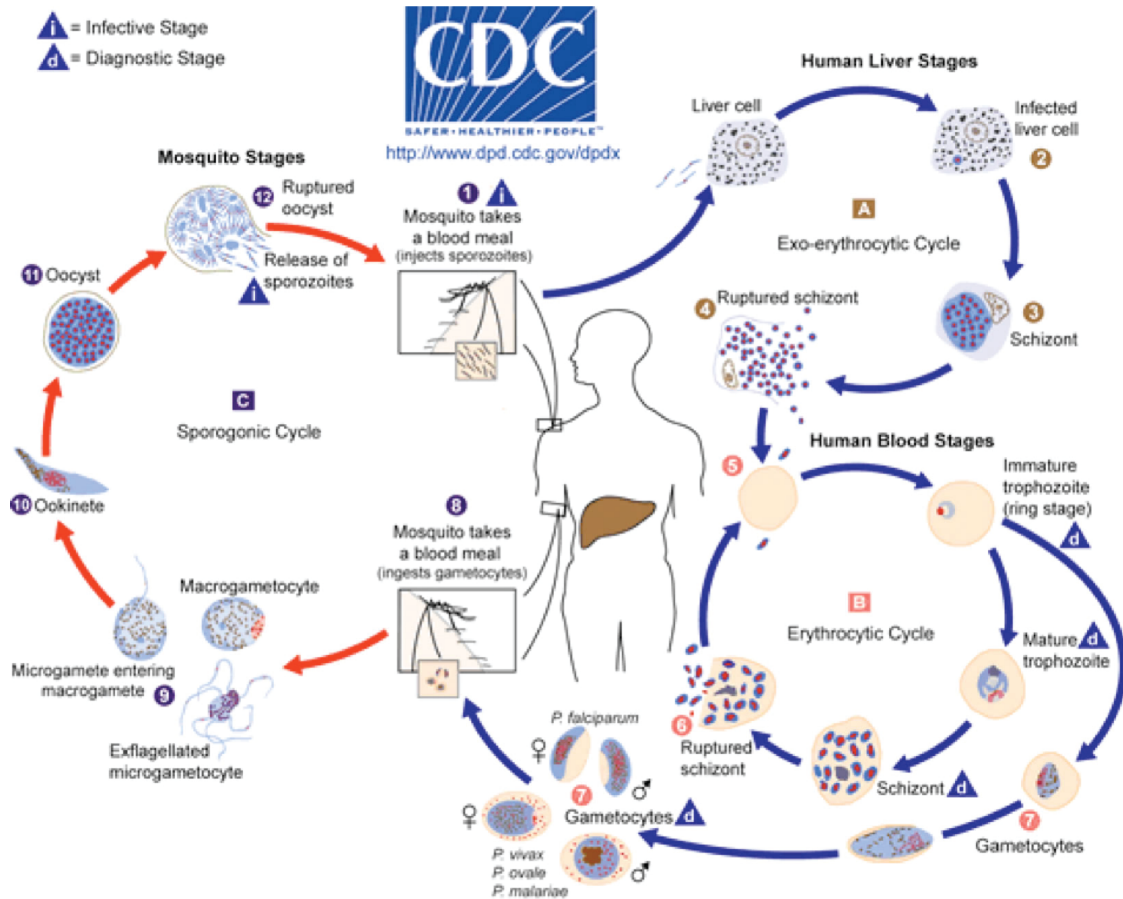
Understanding the genome is not simply about understanding which genes are there. Understanding when each gene is used helps us to find out how organisms develop and which genes are used in response to particular external stimuli. The first layer in understanding how the genome is used is the transcriptome. This is also the most accessible because like the genome the transcriptome is made of nucleic acids and can be sequenced using the same technology. Arguably the proteome is of greater relevance to understanding cellular biology however it is chemically heterogeneous making it much more difficult to assay.

Over the past decade or two microarray technology has been extensively applied to addressing the question of which genes are expressed when. Despite its success this technology is limited in that it requires prior knowledge of the gene sequences for an organism and has a limited dynamic range in detecting the level of expression, e.g. how many copies of a transcript are made. RNA sequencing technology, using for instance Illumina HiSeq machines, can sequence essentially all the genes which are transcribed and the results have a more linear relationship to the real number of transcripts generated in the cell.

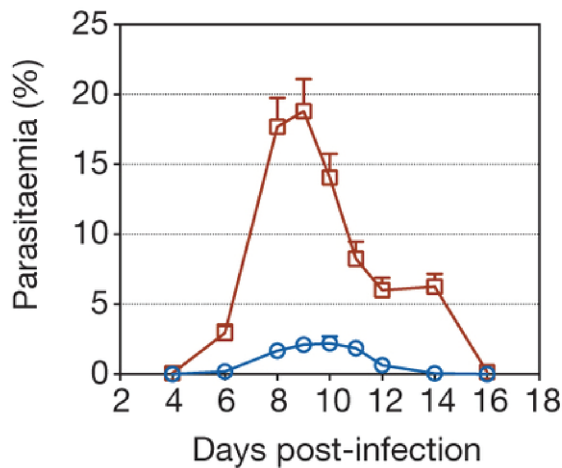
The aim of differential expression analysis is to determine which genes are more or less expressed in different situations. We could ask, for instance, whether a bacterium uses its genome differently when exposed to stress, such as excessive heat or a drug. Alternatively we could ask what genes make human livers different from human kidneys.

In this module we will address the effect of vector transmission on gene expression of the malaria parasite. Is the transcriptome of a mosquito-transmitted parasite different from one which has not passed through a mosquito? The key reason for asking this question is that parasites which are transmitted by mosquito are less virulent than those which are serially blood passaged in the laboratory. Figure 1A shows the malaria life cycle, the blue part highlighting the mosquito stage. Figure 1B shows the difference in virulence, measured by blood parasitemia, between mosquito-transmitted and serially blood passaged parasites. The data in this exercise, as well as figures 1B and 1C are taken from Spence et al. (2013).

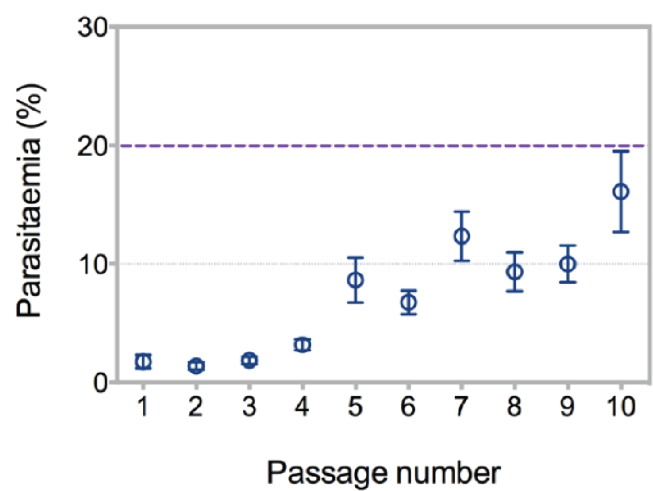
A



B



C



**Figure 1. Serial blood passage increases virulence of malaria parasites.** (A) The lifecycle of plasmodium parasites involves mammalian and mosquito stages. Experiments in the lab often exclude the mosquito stage (red) and instead remove parasites from the blood of a mouse to infect another mouse (serial blood passage). (B) Serially blood passaged parasites (red) are more virulent than mosquito-transmitted parasites (blue) as shown by their higher parasitemia over the course of infection. (C) As mosquito transmitted parasites are serially blood passaged an increasing number of times, they return to a higher level of parasitemia.

Figure 1C shows that increasing numbers of blood passage post mosquito transmission results in increasing virulence, back to around 20% parasitemia. Subsequent mosquito transmission of high virulence parasites renders them low virulence again. We hypothesise that parasites which have been through the mosquito are somehow better able to control the mosquito immune system than those which have not. This control of the immune system would result in lower parasitemia because this is advantageous for the parasite. Too high a parasitemia is bad for the mouse and therefore bad for the parasite. Are there any differences between the transcriptomes of serially blood passaged parasites and mosquito-transmitted parasites which might explain how they are able to do this?

## Module Summary

1. Mapping RNA-seq reads to the genome using *HISAT2*
2. Using *Artemis* to visualise transcription
3. Using *Kallisto* and *Sleuth* to identify differentially expressed genes
4. Using *Sleuth* to quality check the data
5. Interpreting the results

# 1. Mapping RNA-seq reads to the genome using *HISAT2*

We have two conditions: serially blood-passaged parasites (SBP) and mosquito transmitted parasites (MT). One with three biological replicates (SBP), one with two (MT). Therefore we have five RNA samples, each which has been sequenced on an Illumina HiSeq sequencing machine. For this exercise we have reduced the number of reads in each sample to around 2.5m to reduce the mapping time. However this will be sufficient to detect most differentially expressed genes.

Firstly, make a HISAT2 index for the *P. chabaudi* genome reference sequence.

```
hisat2-build PccAS_v3_genome.fa PccAS_v3_hisat2idx
```

Map the reads for the MT1 sample using HISAT2. Each of the following steps will take a couple of minutes.

```
hisat2 --max-intronlen 10000 -x PccAS_v3_hisat2idx -1  
MT1_1.fastq -2 MT1_2.fastq -S MT1.sam
```

Convert the SAM file to a BAM.

```
samtools view -b -o MT1.bam MT1.sam
```

Sort the BAM file (otherwise the indexing won't work)

```
samtools sort -o MT1_sorted.bam MT1.bam
```

Index the BAM file so that it can be read efficiently by Artemis

```
samtools index MT1_sorted.bam
```

Now map, convert SAM to BAM, sort and index with the reads from the MT2 sample.

Note the BAM files and .bai index files provided for the SBP samples:

```
ls *bam*
```

## 2. Using *Artemis* to visualise transcription

Index the fasta file so Artemis can view each chromosome separately

```
samtools faidx PccAS_v3_genome.fa
```

Load chromosome 14 into Artemis from the command line, displaying the mapped reads from each sample:

```
art -  
Dbam="MT1_sorted.bam,MT2_sorted.bam,SBP1_sorted.bam,SBP2_sorted.  
bam,SBP3_sorted.bam" PccAS_v3_genome.fa +PccAS_v3.gff.gz &
```

Select "Use index" so Artemis will show individual chromosomes.

The screenshot shows a genome browser interface with the following components:

- 1**: Menu bar (File, Entries, Select, View, Goto, Edit, Create, Run, Graph, Display).
- 2**: Entry selection area showing 'PccAS\_v3\_genome.fa', 'PccAS\_01\_v3', and 'PccAS\_v3.gff.gz'.
- 3**: BAM view showing sequencing reads mapped to the genome sequence.
- 4**: Sequence view panel showing forward and reverse DNA strands, stop codons, and annotated features like genes (PCHAS\_0100100, PCHAS\_0100200).
- 5**: Zoomed-in view of nucleotides and amino acids.
- 6**: Sliders for zooming view panels.
- 7**: Sliders for scrolling along the DNA.

### 1. Drop-down menus

2. **Entry (top line):** shows which entries are currently loaded with the default entry highlighted in yellow. You can select different chromosomes to view here.

3. **BAM view:** Displays reads mapped to the genome sequence. Each little horizontal line represents a sequencing read. Some reads are blue indicating that they are unique reads. Green reads represent multiple reads mapped to exactly the same position on the reference sequence. Grey lines in the middle of reads mean that the read has been split and this usually means it maps over an intron. If you click a read its mate pair will also be selected. If you want to know more about a read right-click and select 'Show details of: READ NAME'.

4. **Sequence view panel.** The central two grey lines represent the forward (top) and reverse (bottom) DNA strands. Above and below these are the three forward and three reverse reading frames (theoretical translations of the genome). Stop codons are marked as black vertical bars. Genes and other annotated features are displayed as coloured boxes. We often refer to predicted genes as coding sequences or CDSs.

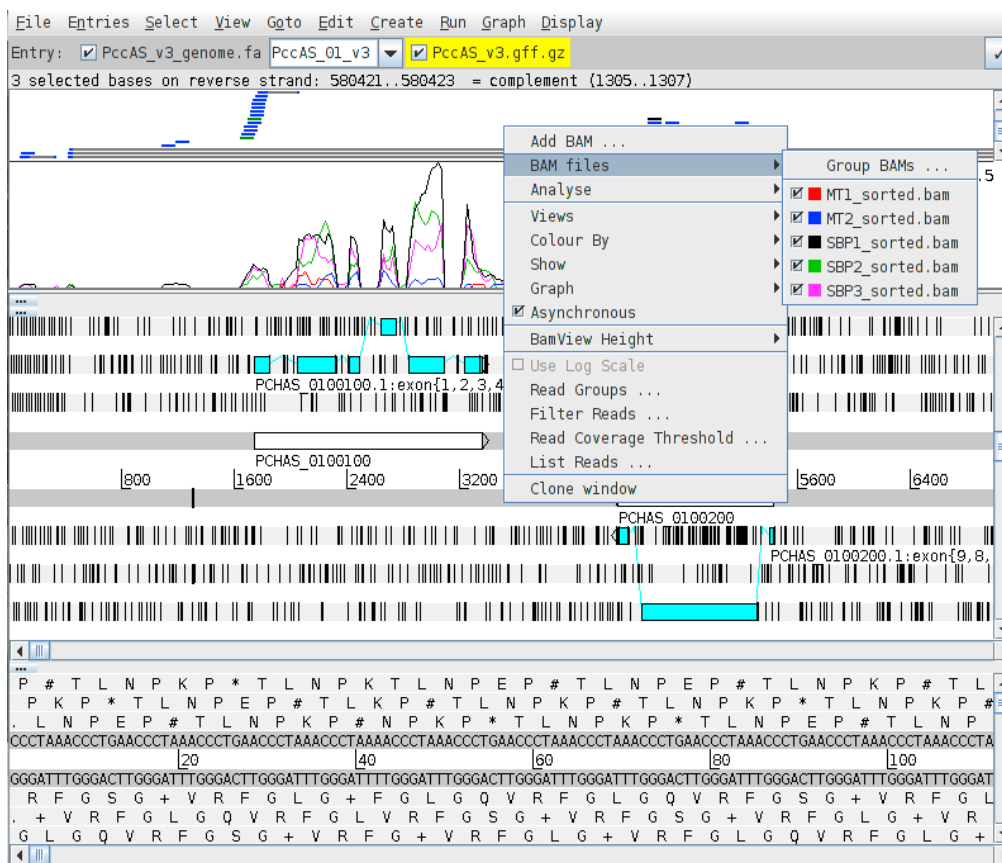
5. This panel has a similar layout to the main panel but is zoomed in to show nucleotides and amino acids.

6. **Sliders** for zooming view panels.

7. **Sliders** for scrolling along the DNA.

Right click on the BAM view, select *Graph*, then *Coverage*.

Right click on the BAM window showing the reads and hover over *BAM files*. This will show you which colours in the coverage plot relate to which samples. Scroll through the chromosome and see if you can identify genes which might be differentially expressed between SBP and MT parasites. Is looking at the coverage plots alone a reliable way to assess differential expression? Hint: what is the difference between read count and RPKM? Are the libraries all the same size?



Select chromosome PccAS\_14\_v3 from the drop down box on the Entry line.

Press Ctrl-g and use “Goto Feature With Gene Name” to navigate to the gene PCHAS\_1402500.



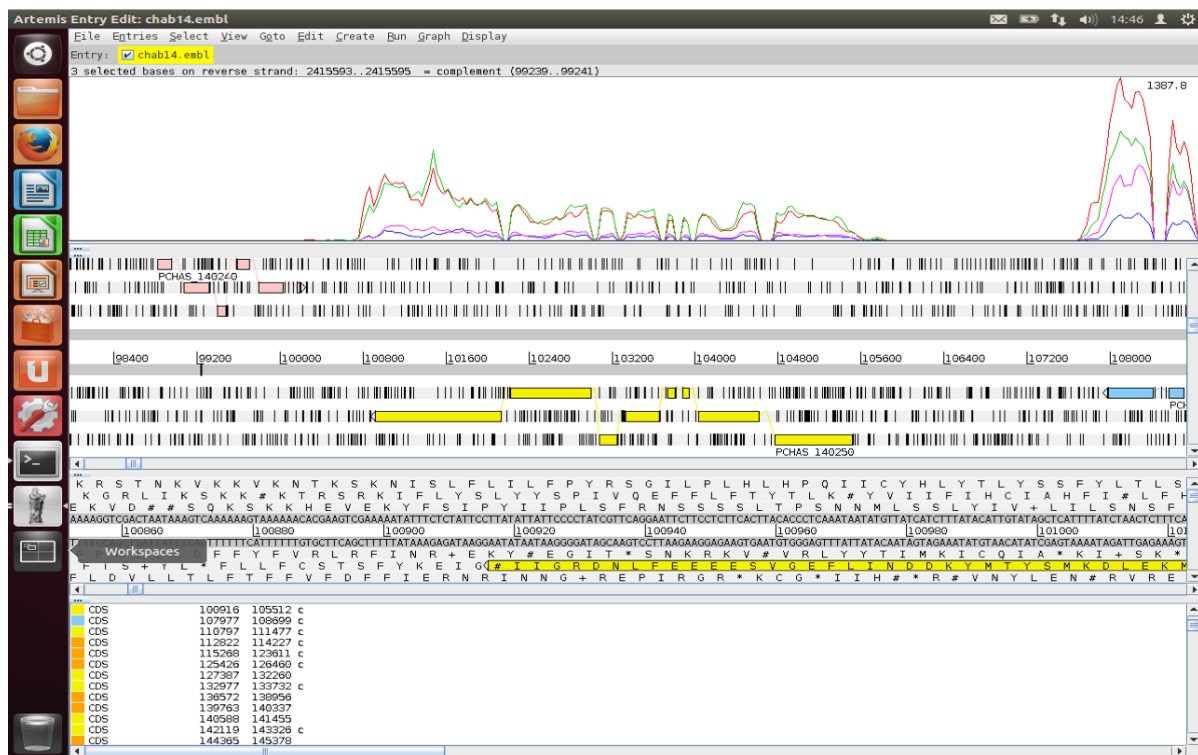
Investigate the coverage for this gene. Does the RNA-seq mapping agree with the gene model in blue?

You can determine read counts and RPKMs for individual genes within Artemis.

Click on the blue gene model, right click on the BAMview window, select *Analyse*, then *RPKM value of selected features*.

Artemis asks whether you want to include introns in the calculations. We are only interested in reads mapping to the spliced transcript, so you should exclude these. Select *Use reads mapped to all reference sequences*. Is this important?

After the analysis is done a window will appear behind Artemis.



This gene looks to be up-regulated in serially blood passaged parasites; SBP samples have RPKMs several times greater than the MT samples. Is it statistically significant? In the next section we will find out.

### 3. Using *Kallisto* and *Sleuth* to identify differentially expressed genes

*Kallisto* is a read mapper, but instead of mapping against the genome it is designed to map against the transcriptome, i.e. the spliced gene sequences inferred from the genome annotation. Rather than tell you where the reads map it's aim is in quantifying the expression level of each transcript. It is very fast because it uses pseudoalignment rather than true read alignment.

*Kallisto* needs an index of the transcript sequences.

```
kallisto index -i PccAS_v3_kallisto PccAS_v3_transcripts.fa
```

Quantify the expression levels of your transcripts for the MT1 sample.

```
kallisto quant -i PccAS_v3_kallisto -o MT1 -b 100
MT1_1.fastq MT1_2.fastq
```

The results are contained in the file MT1/abundance.tsv

Use the *kallisto quant* command four more times, for the MT2 sample and the three SBP samples.

*Sleuth* uses the output from *Kallisto* to determine differentially expressed genes. It is written in the R statistical programming language, as is almost all RNA-seq analysis software. Helpfully however it produces a web page that allows interactive graphical analysis of the data. However, I would recommend learning R for anyone doing a significant amount of RNA-seq analysis. It is nowhere near as hard to get started with as full-blown programming languages such as Perl or Python!

We have provided a series of R commands which will get *Sleuth* running. These are in the file *sleuth.R*. Open the file and have a look. It is not as hard as it seems, I copied most of this from the manual! To run this R script, you will have to open R:

R

And then copy and paste commands from the file *sleuth.R*

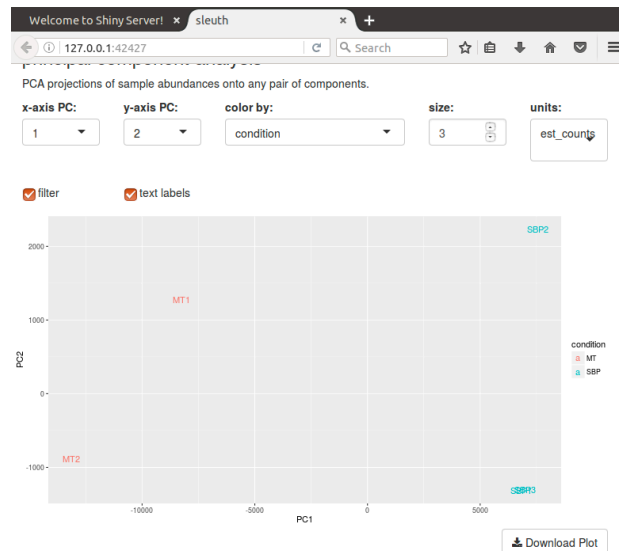
## 4. Using *Sleuth* to quality check the data

*Sleuth* provides several tabs which we can use to determine whether the data is of good quality and whether we should trust the results we get.

In the web page which has been launched click on Summaries->processed data.

Even though we have used the same number of reads for each sample, there are large differences in the number of reads mapping for each one. Why might this be? Is it a problem?

sample	reads_mapped	reads_proc	frac_mapped	bootstraps	condition
MT1	67266	500000	0.1345	100	MT
MT2	136556	500000	0.2731	100	MT
SBP1	407544	500000	0.8151	100	SBP
SBP2	381387	500000	0.7628	100	SBP
SBP3	386637	500000	0.7733	100	SBP



Click on map->PCA.

The Principal Components Analysis plot shows the relationship between the samples in two dimensions (PC1 and PC2). In this case almost all the variation between the samples is captured by just Principal Component 1. The MT samples are well separated from the SBP samples, meaning that the replicates are more similar to each other than they are to samples from the different condition. This is good.

In some cases we identify outliers, e.g. samples which do not agree with other replicates and these can be excluded. If we don't have many replicates, it is hard to detect outliers and our power to detect differentially expressed genes is reduced.

## 5. Interpreting the results

In the R script we printed out a file of results describing the differentially expressed genes in our dataset. This is called “kallisto.results”.

The file contains several columns, of which the most important are:

Column 1: target\_id (gene id)

Column 2: pval (p value)

Column 3: qval (p value corrected for multiple hypothesis testing)

Column 4: b (fold change)

Column 12: description (some more useful description of the gene than its id)

With a little Linux magic we can get the list of differentially expressed genes with only the columns of interest as above. The following command will get those genes which have an adjusted p value less than 0,01 and a positive fold change. These genes are more highly expressed in SBP samples.

```
cut -f1,3,4,12 kallisto.results | awk '$2 < 0.01 && $3 > 0'
```

These genes are more highly expressed in MT samples:

```
cut -f1,3,4,12 kallisto.results | awk '$2 < 0.01 && $3 < 0'
```

How many genes are more highly expressed in each condition?

Do you notice any particular genes that come up in the analysis?

The most highly up-regulated genes in MT samples are from the *cir* family. This is a large, malaria-specific gene family which had previously been proposed to be involved in immune evasion (Lawton et al., 2012). Here however we see many of these genes up-regulated in a form of the parasite which seems to cause the immune system to better control the parasite. This suggests that these genes interact with the immune system in a more subtle way, preventing the immune system from damaging the host.

Remember PCHAS\_1402500? Of course you do. It was the gene we looked at in Artemis that seemed absolutely definitely differentially expressed.

What does Sleuth think about it?

```
grep PCHAS_1402500 kallisto.results | cut -f1,3,4,12
```

Although this gene looked like it was differentially expressed from the plots in Artemis our test did not show it to be so. This might be because some samples tended to have more reads, so based on raw read counts, genes generally look up-regulated in the SBP samples. Alternatively the reliability of only two biological replicates and the strength of the difference between the conditions was not sufficient to be statistically convincing. In the second case increasing the number of biological replicates would give us more confidence about whether there really was a difference.

In this case, the lower number of reads mapping to MT samples mislead us in the Artemis view. Luckily careful normalisation and appropriate use of statistics saved the day!

If you want to read more about the work related to this data it is published: Spence et al. 2013 (PMID: 23719378).

## Key aspects of differential expression analysis

### Replicates and power

In order to accurately ascertain which genes are differentially expressed and by how much it is necessary to use replicated data. As with all biological experiments doing it once is simply not enough. There is no simple way to decide how many replicates to do, it is usually a compromise of statistical power and cost. By determining how much variability there is in the sample preparation and sequencing reactions we can better assess how highly genes are really expressed and more accurately determine any differences. The key to this is performing biological rather than technical replicates. This means, for instance, growing up three batches of parasites, treating them all identically, extracting RNA from each and sequencing the three samples separately. Technical replicates, whereby the same sample is sequenced three times do not account for the variability that really exists in biological systems or the experimental error between batches of parasites and RNA extractions.

n.b. more replicates will help improve power for genes that are already detected at high levels, while deeper sequencing will improve power to detect differential expression for genes which are expressed at low levels.

### P-values vs. q-values

When asking whether a gene is differentially expressed we use statistical tests to assign a p-value. If a gene has a p-value of 0.05 we say that there is only a 5% chance that it is not really differentially expressed. However, if we are asking this question for every gene in the genome (~5500 genes for *Plasmodium*), then we would expect to see p-values less than 0.05 for many genes even though they are not really differentially expressed. Due to this statistical problem we must correct the p-values so that we are not tricked into accepting a large number of erroneous results. Q-values are p-values which have been corrected for what is known as **multiple hypothesis testing**. Therefore it is a q-value of less than 0.05 that we should be looking for when asking whether a gene is differentially expressed.

### Alternative software

If you have a good quality genome and genome annotation such as for model organisms e.g. human, mouse, *Plasmodium*, I would recommend mapping to the transcriptome for determining transcript abundance. This is even more relevant if you have variant transcripts per gene as you need a tool which will do its best to determine which transcript is really expressed. As well as Kallisto (Bray et al. 2016; PMID: 27043002), there is eXpress (Roberts & Pachter, 2012; PMID: 23160280) which will do this.

Alternatively you can map to the genome and then call abundance of genes, essentially ignoring variant transcripts. This is more appropriate where you are less confident about the genome annotation and/or you don't have variant transcripts because your organism rarely makes them or they are simply not annotated. Tophat2 (Kim et al., 2013; PMID: 23618408), HISAT2 (Pertea et al. 2016; PMID: 27560171), STAR (Dobin et al., 2013; PMID: 23104886) and GSNAP (Wu & Nacu, 2010; PMID: 20147302) are all splice-aware RNA-seq read mappers appropriate for this task. You then need to use a tool which counts the reads overlapping each gene model. HTSeq (Anders et al., 2015; PMID: 25260700) is a popular tool for this purpose. Cufflinks (Trapnell et al. 2012; PMID: 22383036) will count reads and determine differentially expressed genes.

There are a variety of programs for detecting differentially expressed genes from tables of RNA-seq read counts. DESeq2 (Love et al., 2014; PMID: 25516281), EdgeR (Robinson et al., 2010; PMID: 19910308) and BaySeq (Hardcastle & Kelly, 2010; PMID: 20698981) are good examples.

### What do I do with a gene list?

Differential expression analysis results is a list of genes which show differences between two conditions. It can be daunting trying to determine what the results mean. On one hand you may find that there are no real differences in your experiment. Is this due to biological reality or noisy data? On the other hand you may find several thousands of genes are differentially expressed. What can you say about that?

Other than looking for genes you expect to be different or unchanged, one of the first things to do is look at Gene Ontology (GO) term enrichment. There are many different algorithms for this, but you could annotate your genes with functional terms from GO using for instance Blast2GO (Conesa et al., 2005; PMID: 16081474) and then use TopGO (Alexa et al., 2005; PMID: 16606683) to determine whether any particular sorts of genes occur more than expected in your differentially expressed genes.



# Task 1

# Sexual Development

## Introduction

*Plasmodium berghei* is used as a rodent model of malaria. It is known that in the lab it can evolve to stop producing sexual stages (Figure 1). We want to try and use this observation to our advantage. If we can understand how the parasite switches to the sexual, transmissible stage, then we might better understand how to prevent this from happening and prevent the spread of malaria.

Several cultures of a transmissible strain were grown continuously in the lab for several months (Figure 2). These all became gametocyte non-producers (GNPs). The genomes of these strains were then sequenced, the data mapped and the variants called. Your first job is to identify the mutations in these strains, which contribute to the GNP phenotype. Our hypothesis is that while each GNP strain will have many mutations compared to the parental strain, **only one gene will have unique mutations in every strain** and this gene will be a key regulator of gametocytogenesis. Luckily there is evidence from earlier work that the gene is located on chromosome 14, so we need only consider that one!

The variant call files for each mutant are available in the data directory. A full explanation is found in a README file in the directory.

Once you have found the gene you can explore its role in gametocytogenesis. We have RNA-seq data from a strain where the gene has been knocked out allowing us to examine how the gene affects the transcriptional landscape of the parasite. Which transcripts are affected by the knockout of this gene? What does this tell us about the importance of the gene in the switch to sexual development? What could this gene list be useful for in future?

Sequencing reads as well as the reference sequence are available in the data directory (please use this rather than download one). There are also files of genome annotation, product descriptions, GO terms and an R script for performing GO term enrichment. A full explanation is found in a README file in the directory.



## Summary

To achieve this goal you should:

- View the variant calls in Artemis
- Identify the gene responsible for gametocyte development
- Map the RNA-seq data to the reference
- Confirm the knockout in the mutant samples
- Call differentially expressed genes
- Perform a Gene Ontology enrichment analysis (we have provided an R script to help with this)

# Task 2

## Georeferencing genomic data

### Introduction

Phylogenetic trees based on whole genome data tell us the about the relationships of bacterial isolates to each other on a very fine scale. When we combine that high resolution information about the evolutionary relationships of isolates with geographical data it can inform our understanding of the current distribution of the pathogen and allow us to **infer the epidemiological processes** that have acted on the pathogen over time. The simplest example of this would be if a phylogeny showed that a pathogen was **geographically constrained** (e.g. isolates from the same region always cluster together). This might indicate that the pathogen is not highly mobile, whereas a pathogen with a phylogeny that shows isolates from distant regions are equally likely to be related to each other as isolates from nearby is likely to be highly mobile across regional borders. Geographical referencing of genomic data can also be **combined with temporal information** to study the movement of pathogens in space and time in real time for use in outbreak detection and monitoring.

For this task, you will be split into teams. Using the skills you learned in the structured modules, your team will use the all the **mapped sequences (.fa)** produced in Module 7 and to identify **single nucleotide polymorphism (SNP)** sites based on the reference sequence strain SL1344 and subsequently construct a **phylogenetic tree**. You will use the software **FigTree** to view and interpret your phylogeny and geo-reference your data using **Microreact** to develop your own hypotheses about the pathogen distribution. In addition, you will use the software **ARIBA (Antibiotic Resistance Identification By Assembly)** to investigate resistance genes in these strains and a free visualisation tool, **Phandango** to compare with the pathogen distribution observed in your tree.

### The aims of this exercise are:

- 1) Introduce the biology & workflow
- 2) Gain experience in building and interpreting phylogenetic trees
- 3) Introduce concepts and tools for geo-referencing metadata
- 4) Show how **Next Generation Sequencing data** can be used to describe the evolution and distribution of a pathogen across a geographical area
- 5) Combine phylogenetic analysis and comparative genomic techniques
- 6) Demonstrate the value of shared data resources
- 7) Gain experience in presenting the results of **Next Generation Sequencing Data** analysis

## Background

### Biology

To learn about phylogenetic reconstruction and geo-referencing for epidemiological inference, we will work with some software that has already been introduced as well as some new software introduced in this module. We will work with real data from *Salmonella enterica* serovar Typhimurium sampled from regional labs in England and Wales, United Kingdom in 2015.

### *Salmonella enterica* serovar Typhimurium

*Salmonella enterica* is a diverse bacterial species that can cause disease in both human and animals. Human infections caused by *Salmonella* can be divided into two, typhoidal *Salmonella* or non-typhoidal *Salmonella* (NTS). The former include Typhi and Paratyphi serovars that cause typhoid. NTS comprises of multiple serovars that cause self-limiting gastroenteritis in humans and is normally associated with zoonotic *Salmonella* reservoirs, typically domesticated animals, with little or no sustained human-to-human transmission.

*Salmonella enterica* serovar Typhimurium (*S. Typhimurium*), unlike the classical views of NTS, can cause an invasive form of NTS (iNTS), with distinct clinical representations to typhoid and gastroenteritis and normally characterized by a nonspecific fever that can be indistinguishable from malaria and in rare cases is accompanied by diarrhoea (Okoro *et al. Nature Genetics*, 2012).

Whole genome sequence analysis of this organism provides some insight into the short-term microevolution of *S. Typhimurium*. Understanding the level of diversity in this time-period is crucial in attempting to identify if this is an outbreak or sporadic infection.

## Your task

The Global Health Authority (GHA) has asked you to provide an overview of *Salmonella enterica* serovar Typhimurium in England and Wales, using retrospective samples. In teams you will develop a whole-genome sequencing based tree from all 24 sequences and correlate this to the geography of the city. You will also look into the distribution of antimicrobial resistance and investigate the genetic basis for the resistance phenotype you identified in the laboratory. At the end of the task each group will present their findings.

The five teams that will have been assigned in the previous day.

The following division of responsibilities in your teams is recommended. If there are extra people, then they should help with the tree builder and both should work on the georeferencing task once you have a tree.

- **Tree Builder** – SNP-calling and phylogenetic inference
- **Antimicrobial Resistance Investigator** – ARIBA and Phandango
- **Geo-referencer and Reporter** – geo-referencing with Microreact
- **Presentation** – All group members

# GENERAL INFORMATION

## Data provided

As a team you will create a phylogenetic tree of the *S. Typhimurium* isolates from England and Wales. You will geo-reference this information, as well as antimicrobial resistance data, against the address of the isolates to form ideas about the distribution and epidemiology of the pathogen. Additionally, the genetic basis for the antimicrobial resistance will be explored.

To achieve this, each team is provided with the following files in the Task folder:

- A metadata table (**metadata.xls**) which contains information on the isolates including the date and address of collection.
- Your sequence data folder in each of your **groups folders**, which contain symlinked or symbolic linked sequenced data, fastq.gz (unix command: **ln -s**). These act like 'hyperlink' data. Symlinked data is often used to save space when you do not want to copy large files.
- An **ariba\_reports** folder that contains a summary the resistance reports from ARIBA to save time. You are encouraged to run the ARIBA analysis on the samples allocated to your group.
- *S. Typhimurium* **fasta** and **embl** files, which you will use as a reference.
- A **pseudogenomes** folder that contains an .fa file
- And a **PDF** of the literature reference cited on page 2.

```
wt@Pathogens: ~/Group_task_Georeferencing
wt@Pathogens:~/Group_task_Georeferencing$ ls
ariba_reports  group_7
group_1       group_8
group_10     group_9
group_2      metadata.xls
group_3      Okoro_2012.pdf
group_4      pseudogenomes
group_5      Salmonella_enterica_serovar_Typhimurium_SL1344_2.5MB.embl
group_6      Salmonella_enterica_serovar_Typhimurium_SL1344_2.5MB.fasta
wt@Pathogens:~/Group_task_Georeferencing$
```

assigned group	sample	MLST	year	month	Address	longitude	latitude
1	Stm_1	19	2015	May	Bệnh viện quận 8, 82 Cao Lỗ, phường 4, Quận 8		
	Stm_2	19	2015	May	Hanh Phuc International Hospital,97, Nguyen Thi Minh Khai Street, Ben Nghe Ward, Quận 1		
	Stm_3	19	2015	April	Bệnh viện Nhân dân 115, 527 Sư Vạn Hạnh, 12th Ward, Quận 10, Ho Chi Minh City		
2	Stm_4	19	2015	March	Bệnh viện Nhân dân 115, 527 Sư Vạn Hạnh, 12th Ward, Quận 10, Ho Chi Minh City		
	Stm_5	19	2015	June	Bệnh viện quận 8, 82 Cao Lỗ, phường 4, Quận 8		
	Stm_6	19	2015	March	Hanh Phuc International Hospital,97, Nguyen Thi Minh Khai Street, Ben Nghe Ward, Quận 1		
3	Stm_7	19	2015	June	Bệnh viện quận 8, 82 Cao Lỗ, phường 4, Quận 8		
	Stm_8	19	2015	September	Bệnh viện Bệnh Nhiệt đới, 764 Võ Văn Kiệt, phường 1, Quận 5		
	Stm_9	19	2015	July	An Binh Hospital,146 An Binh, 7th Ward, Quận 5		
4	Stm_10	19	2015	October	Bệnh viện Bệnh Nhiệt đới, 764 Võ Văn Kiệt, phường 1, Quận 5		
	Stm_11	19	2015	July	Bệnh viện Bệnh Nhiệt đới, 764 Võ Văn Kiệt, phường 1, Quận 5		
	Stm_12	19	2015	July	Bệnh viện Bệnh Nhiệt đới, 764 Võ Văn Kiệt, phường 1, Quận 5		
5	Stm_13	19	2015	May	Hanh Phuc International Hospital,97, Nguyen Thi Minh Khai Street, Ben Nghe Ward, Quận 1		
	Stm_14	19	2015	April	Bệnh viện Nhân dân 115, 527 Sư Vạn Hạnh, 12th Ward, Quận 10, Ho Chi Minh City		
	Stm_15	19	2015	March	Bệnh viện quận 8, 82 Cao Lỗ, phường 4, Quận 8		
6	Stm_16	19	2015	April	Bệnh viện Nhân dân 115, 527 Sư Vạn Hạnh, 12th Ward, Quận 10, Ho Chi Minh City		
	Stm_17	19	2015	January	Bệnh viện Nhân dân 115, 527 Sư Vạn Hạnh, 12th Ward, Quận 10, Ho Chi Minh City		
	Stm_18	19	2015	July	Bệnh viện Bệnh Nhiệt đới, 764 Võ Văn Kiệt, phường 1, Quận 5		
8	Stm_19	19	2015	July	An Binh Hospital,146 An Binh, 7th Ward, Quận 5		
	Stm_20	19	2015	July	Bệnh viện Bệnh Nhiệt đới, 764 Võ Văn Kiệt, phường 1, Quận 5		
	Stm_21	19	2015	July	Bệnh viện Bệnh Nhiệt đới, 764 Võ Văn Kiệt, phường 1, Quận 5		
9	Stm_22	19	2015	June	An Binh Hospital,146 An Binh, 7th Ward, Quận 5		
	Stm_23	19	2015	June	An Binh Hospital,146 An Binh, 7th Ward, Quận 5		
	Stm_24	19	2015	June	An Binh Hospital,146 An Binh, 7th Ward, Quận 5		

# GENERAL INFORMATION

## Your isolate names

The isolate names you can see in the subfolders of your group folder. There are two files for every isolate `_1.fastq.gz` and `_2.fastq.gz`. These represent the forward and reverse reads of paired end sequencing for that isolate.

When you work with your own sequencing data after the course, other naming conventions will be used. As in the example above, it is likely this formats will include helpful pieces of information, so find out what your own sequencing data names mean when the time comes!

## How to use this module

As in some previous modules, you will be provided with many of the commands you will need to perform the analysis. As you will be distributing the tasks between people in your group, each role has their own set of guiding pages and focuses on different skills. You will learn about the other roles while integrating the results of your individual analyses and have the opportunity to work through the other sections in your own time.



# GENERAL INFORMATION

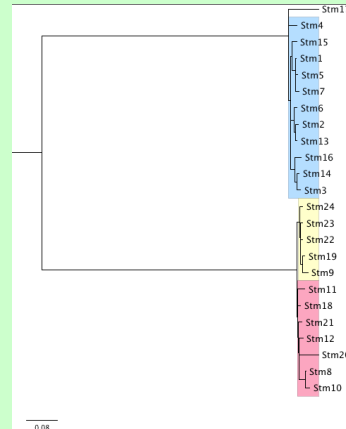
## Team Presentation

At the end of the task compile your findings and interpretations into a 5 minute presentation. All team members should contribute to making the presentation. Examples of some of the exciting key images you might produce are below, but don't be limited by these ideas - please be as creative as you like!

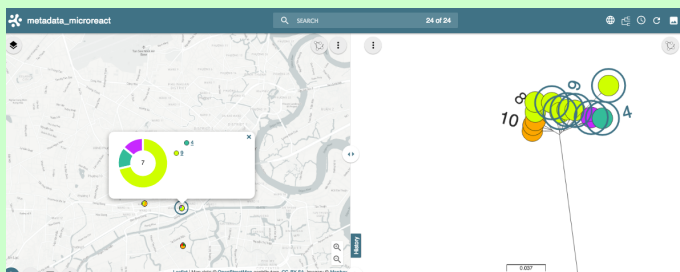
## ANTIMICROBIAL RESISTANCE INVESTIGATORS



Presentation



## TREE BUILDERS



## GEOREFERENCERS

# TREE BUILDERS

## General Information

In this role you are responsible for the construction of a tree from whole genome sequencing data for your country. During this section you and your fellow tree maker will map the sequence data of your country isolates to the *Salmonella enterica* Typhimurium reference genome SL1344. To save on time, you will only be mapping the fastq files to 2.5 million base pairs of the genome. Although this will take a long time, keep in mind that this step would ordinarily take many more hours of computation time. If there is more than one of you working on the mapping portion, you can work on half the samples in step 1. When you get to Step 2, combine your data and work on one computer together.

Bioinformatic processing of data into biologically-meaningful outputs involves the **conversion of data** into many different forms. Just like working in the laboratory, it's useful to break this process down into individual steps and have a plan.

A rough guide of the steps for this task is below and in the following schematic. Check that you understand the principles of each one and then get started:

**Step 1.** Map and call SNPs for each isolate using commands introduced earlier in the course

**Step 2.** Create a whole genome sequence alignment

**Step 3.** Build a phylogenetic tree from the SNP data in your alignment

**Step 4.** Interpret your phylogeny and report the lineages to the geo-referencer

## Step 1: Map and call SNPs for each isolate

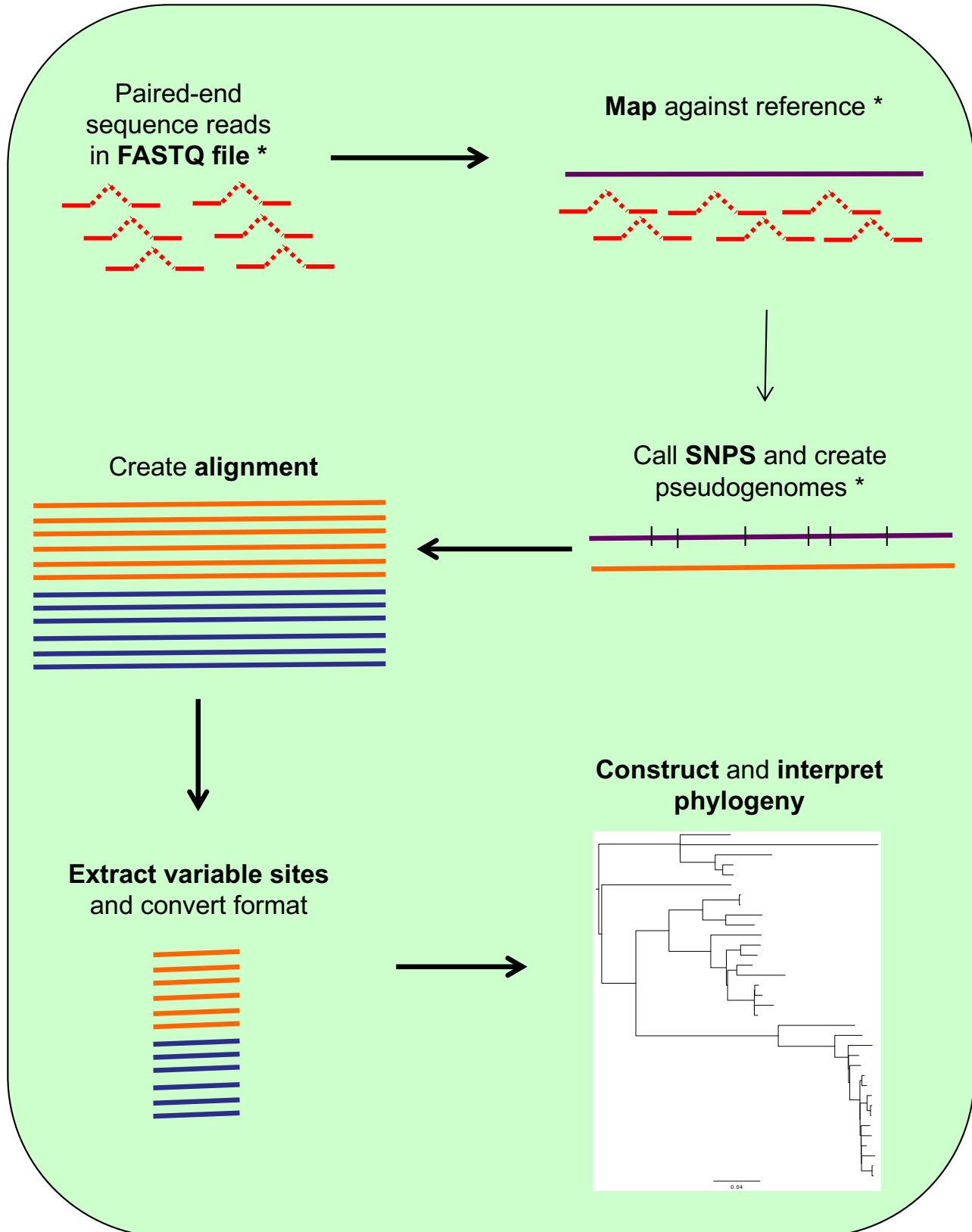
Your reference sequence for this is *Salmonella enterica* serovar Typhimurium strain SL1344, called **Salmonella\_enterica\_serovar\_Typhimurium\_SL1344\_2.5MB.fasta** in the task folder. You may want to create a local copy in your working directory by using the **cp** command.

Map the sequencing data for each isolate to the reference genome and obtain a pseudogenome (incorporating the isolate SNPs into the reference sequence). The required commands were covered in the **mapping and phylogeny** modules. If you struggle with the commands, ask the instructors for a command cheat sheet.

**NOTE: Before you continue onto the next step, you must do some housekeeping. Refer to the mapping and phylogeny module, for which files you should remove.**

# TREE BUILDERS

## Schematic of task workflow



\*do for each isolate

# TREE BUILDERS

## Step 2: Create a whole genome sequence alignment for your data

Now you have created pseudogenomes (.fasta **NOT** .fastq) for each of your samples, you can use this data to **create a sequence alignment** to build a phylogenetic tree. Using this mapping based approach we are able to avoid the computational power required to align millions of base pairs of DNA that would be needed with e.g. CLUSTAL or MUSCLE. Here, because all of the isolates were mapped to the same reference genome, they are already the same length, so they can just be pasted together to form an alignment. Then, you can combine them with **information from global reference isolates** that were created for you in the same way.

Due to time constrains we have mapped all the samples to the same reference. The file can be found in the **pseudogenome** folder. The pseudogenomes of all 24 strains were combined together using the **cat** command as below.

```
cat *_pseudogenome.fasta > All_pseudogenomes.fa
```

This produces a mutlifasta file '**All\_pseudogenomes.fa**' that contains all 24 sequences. You can check all 24 sequences are present by opening it in seaview.

Here, the \* acts as a wildcard symbol and a single file containing all of the pseudogenome sequences pasted one after the other is created. Both .fa and .fasta files are sequence files, but the extension is useful for distinguishing files with single (.fasta) and multiple (.fa) sequences

You should now have a file containing 24 taxa each 2.5MB long. Most of the sites in this alignment will be conserved and not provide useful information for phylogenetic inference, so we will shorten the alignment by **extracting the variable sites** using the program **snp-sites**

```
snp-sites -o All_snps.aln All.aln
```

# TREE BUILDERS

## Step 3: Build a phylogenetic tree from the SNP data in your genome alignment

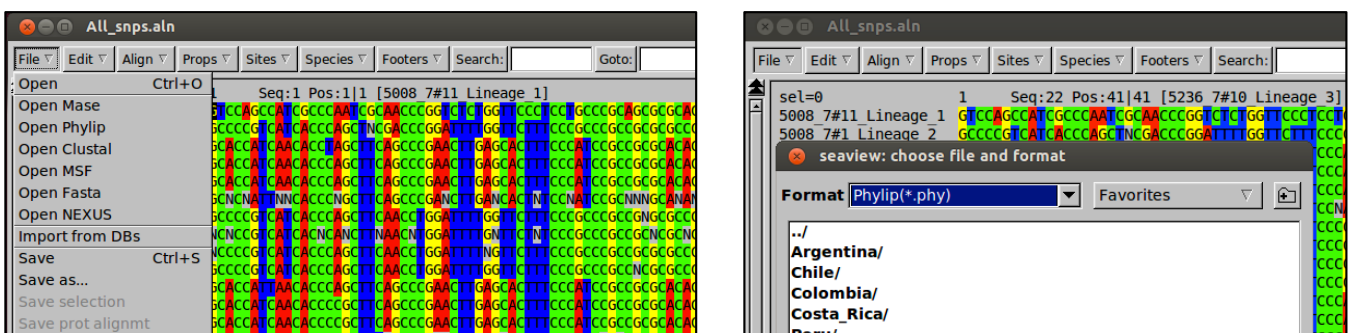
Now you will build a phylogenetic tree from the SNP alignment that you created in the last step. There are a lot of programs for building phylogenetic trees, and here we are going to use one called RAxML which evaluates trees based on maximum likelihood.

The reference is:

A. Stamatakis: "RAxML-VI-HPC: maximum likelihood-based phylogenetic analyses with thousands of taxa and mixed models". In *Bioinformatics*, 2006

Like all programs, RAxML has requirements for the format of input files. Your **All\_snps.aln** file is multifasta format and RAxML requires **phylip format**, so open the file in **seaview** and save it as phylip format under the name **All\_snps.phy** by typing

**seaview All\_snps.aln** then doing **File > Save As > Format > Phylip(\*.phy)**



Then, back at the command line, run **RAxML** by typing the following:

```
raxmlHPC -m GTRGAMMA -p 12345 -n STm -s All_snps.phy
```

Recall that with a single iteration of a maximum likelihood method you risk recovering a tree from a local maximum, which means it might not be the best one. This can be avoided by running multiple iterations with different starting points (we can't do that now because of time). The addition of multiple runs is done by adding the following flag to the command.

**-N 20** would run the program with 20 different starting trees (which is typically enough to find a problem if one exists)

# TREE BUILDERS

## Step 4: Interpret your phylogenetic tree

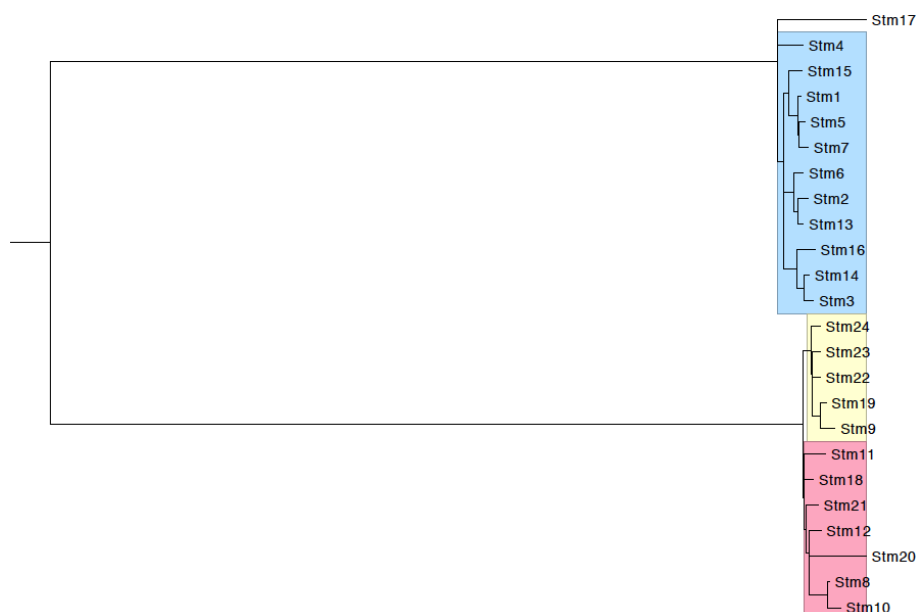
Open your final tree file (**RaxML\_result.STm**) in FigTree and midpoint root it by selecting **Tree > Midpoint Root**.

You will now need to give this file that you have saved in FigTree to your georeferencers in your group. Go to **File > Export trees >** select **Newick** file format. Remember to save your file with a **.nwk** suffix so that you know what type of file it is.

Interpret your phylogenetic tree by first taking some time to make some general observations:

- Are there distinct clades present in the isolates?
- Are there isolates that do not cluster with other isolates?

Then, using the relationships with the known lineages, define each of your isolates as belonging to lineage 1, 2, 3, 4 or Other and pass the information on to your georeferencer. A picture of your tree as well as your general observations about it should go into your team presentation. Take some time to make figure(s) you are happy with and create a pdf picture file by selecting **File > export PDF**



# ANTIMICROBIAL RESISTANCE INVESTIGATORS

## General Information

In this role, you are responsible for the investigation of antimicrobial resistance in isolates from your country. You will correlate the phenotypic metadata with the genetic information contained in the isolates using a local assembly approach with **ARIBA**. ARIBA, Antimicrobial Resistance Identifier by Assembly, is a freely available tool that can be installed from the ARIBA github repository. This tool required a FASTA input of reference sequences, which can be a mutli-fasta file or database of antibiotic resistance genes or non-coding sequences. This **database** will serve as one of your inputs and the other is **paired sequence reads**. ARIBA reports which of the reference sequences were found, plus detailed information on the quality of the assemblies and any variants between the sequencing reads and the reference sequences.

We have installed ARIBA in the virtual machine. You will download the CARD database (<https://card.mcmaster.ca/home>) for resistance detection for your samples, however other databases can be installed.

Further information and installation instructions are detailed in the github wiki page: <https://github.com/sanger-pathogens/ariba/wiki>. The data can then be visualised using Phandango, an interactive also freely available tool to visualise your outputs <http://jameshadfield.github.io/phandango/>.

**Step 1.** Run ARIBA

**Step 2.** Visualise outputs (**phandango.csv** and **.phandango.tre**) in Phandango

**Step 3.** Compare resistance gene present with metadata

**Step 4.** Summarise your findings in text and screen shots for the presentation

## Step 1: Run ARIBA

On the command line, navigate to your group folder in the **Group\_task\_Georeferencing** folder. To run ARIBA you will need to download and format the database. Type:

```
ariba getref card out.card
```

Next you will need to format the reference database for ARIBA. Type:

```
ariba prepareref -f out.card.fa -m out.card.tsv out.card.prepareref
```



# ANTIMICROBIAL RESISTANCE INVESTIGATORS

Next you will need to run local assemblies and call variants, type:

```
ariba run out.card.prepareref reads_1.fastq.gz reads_2.fastq.gz
out.run
```

The command should take about 5 minutes per sample. Be patient and wait for the command prompt (denoted by a \$ sign).

Next you will need to summarise the data from several runs. These are included in newly generated folders. You will combine the data in **report.tsv** files.

```
ariba summary out.summary out.run1/report1.tsv out.run2/report2.tsv
out.run3/report3.tsv
```

Three files will be generated, a **.csv** file with the summary of all the runs and two **.phandango** files. You will need to drag and drop the **out.summary.phandango.tre** and **out.summary.phandango.csv** into the Phandango window.

To understand the whole picture you will need to run ARIBA for **ALL 24 samples**. To save time, we have already done this for all 24 samples. Please use the files in the **ariba\_reports** folder for subsequent steps. We suggest you run the analysis for a few samples to get an idea of the results.

## Step 2. Visualise in Phandango

Interactive visualization of genome phylogenies

phandango

drop your data on to begin

[About / Help \(GitHub wiki\)](#)

[Example Datasets](#)

[Github \(source code\)](#)

[Contact \(email\)](#)

# ANTIMICROBIAL RESISTANCE INVESTIGATORS



On the left hand side is a dendrogram of the phylogenetic relationship of the resistance data and the strains. On the top panel are the matching resistance genes found. The green colour indicates positive match and salmon pink is a negative match.

Consult the CARD database (<https://card.mcmaster.ca/home>) for the resistance phenotype of the genes detected. Note that underscores (\_) in the output data denotes prime (') or bracket, therefore AAC\_3\_-II is AAC(3)-II. The codes for these in a file names **'01.filter.check\_metadata.tsv'** produced when you **prepared** your database (p.11). Consult the report.tsv of the particular sample of interest for the gene names. You can open both .tsv files in excel.

The screenshot shows the CARD website interface. At the top, there are navigation links: Browse, Analyze, Download, and About. Below these is a search bar with the text "Search". The main content area features the title "The Comprehensive Antibiotic Resistance Database" and a subtitle "A bioinformatic database of resistance genes, their products and associated phenotypes." Below this, statistics are listed: "3598 Ontology Terms, 2346 Reference Sequences, 867 SNPs, 2160 Publications, 2272 AMR Detection Models". There are three columns of text: "Browse" (describing the database's rigor), "Analyze" (describing tools for molecular sequence analysis), and "Download" (describing data availability and RGI software).

Some general points to consider are:

- Does the presence of the gene correlate well with the phenotypic results?
- Is it the same in multiple isolates that share the resistance?
- Do you think it is vertically or horizontally transmitted?

# ANTIMICROBIAL RESISTANCE INVESTIGATORS

## **Step 4: Summarise your findings for the presentation**

Coordinate with your other team members to investigate the relationship of your resistance with where the isolates lie in the phylogenetic tree (that the Tree builders produced) and in the country (Georeferencer).

Consolidate your findings into some slides for the presentation and ensure the georeferencer produces a map of the distribution of resistance to complement your work.

# GEO-REFERENCERS AND REPORTERS

## General Information

Geo-referencing information from pathogens can provide insight into the processes that drive their epidemiology. This can be used to infer whether single introductions of a pathogen have occurred followed by local evolution (as in the *S. sonnei* in Vietnam story described in the introductory talk and Holt *et al*, *Nature Genetics*, 2013) or whether it transmits frequently across borders. It can also indicate regions affected by antimicrobial resistance.

In this role, you are going to use the metadata provided and the tools [spatialepidemiology.net](http://spatialepidemiology.net) and **Microreact**.

In this role, you will complete the following steps:

**Step 1.** Identify the global positioning coordinates (longitude and latitude) of the addresses where the isolates were collected

**Step 2.** Create a map of the metadata of the isolates for your country



To start, open the metadata file for the strains in the inbuilt spreadsheet program on the virtual machine and note the address column, as well as the two empty global positioning columns.

In **Step 1** you will be locating isolates based on their **Address** and filling out the information in the Latitude and Longitude columns.

Eventually you will obtain phylogenetic relationships for your samples from the Tree Builders in your team, which you will use as the input for **Microreact**. For now **get started with Steps 1 and 2.**

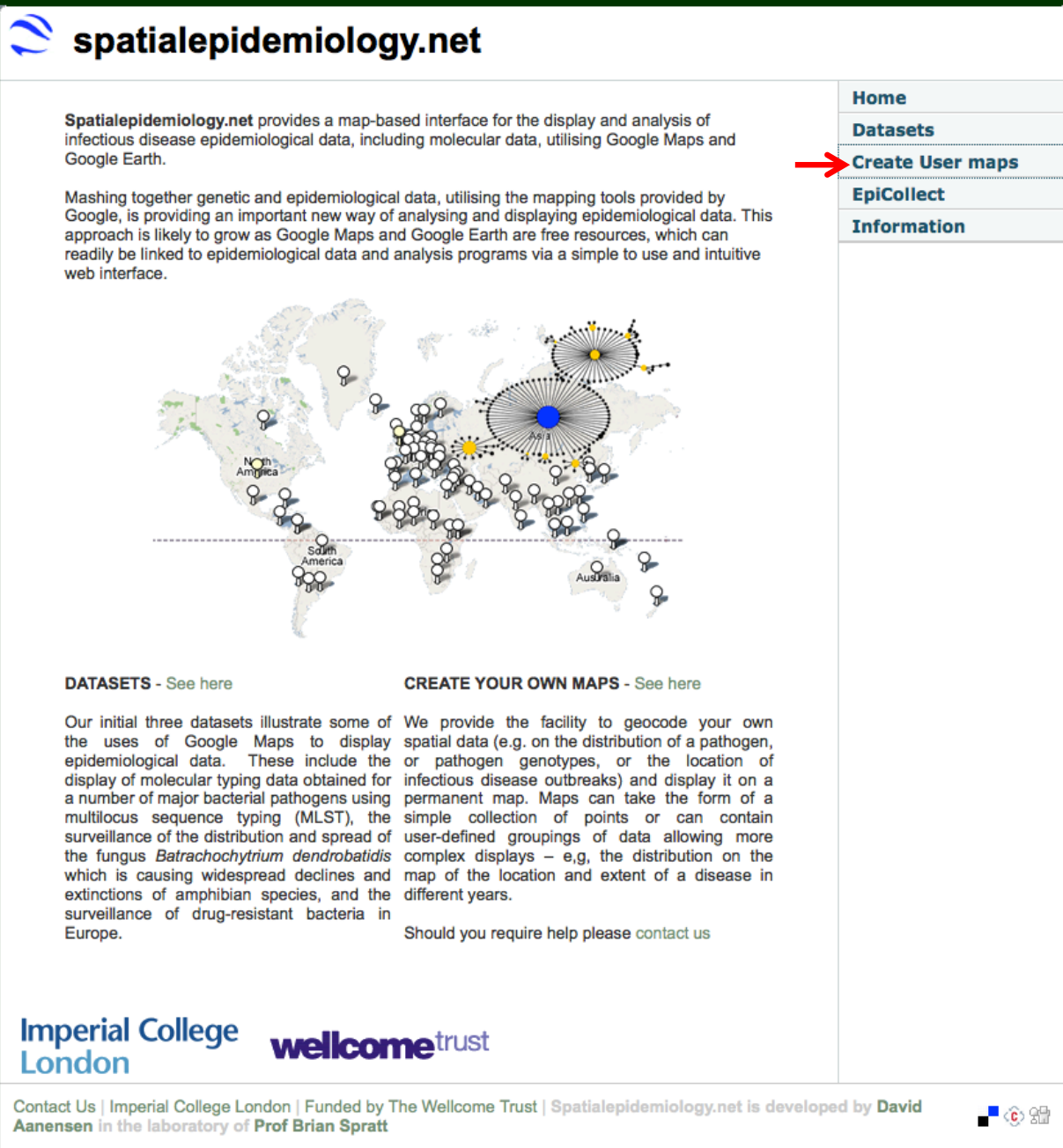
# GEO-REFERENCERS AND REPORTERS

## Step 1: Identify the longitude and latitude of the isolate addresses

Open a browser window and navigate to [www.spatialepidemiology.net](http://www.spatialepidemiology.net) to obtain latitude and longitude coordinates for your addresses.

Click on the **Create User Maps** option on the right hand column (red arrow)

Then click on the 2<sup>nd</sup> tab **Batch Geocode addresses** and copy and paste the address column from your metadata file into this field. Click **Start geocoding**.



**spatialepidemiology.net**

**Spatialepidemiology.net** provides a map-based interface for the display and analysis of infectious disease epidemiological data, including molecular data, utilising Google Maps and Google Earth.

Mashing together genetic and epidemiological data, utilising the mapping tools provided by Google, is providing an important new way of analysing and displaying epidemiological data. This approach is likely to grow as Google Maps and Google Earth are free resources, which can readily be linked to epidemiological data and analysis programs via a simple to use and intuitive web interface.

**Home**  
**Datasets**  
**Create User maps**  
**EpiCollect**  
**Information**

**DATASETS** - See here      **CREATE YOUR OWN MAPS** - See here

Our initial three datasets illustrate some of the uses of Google Maps to display epidemiological data. These include the display of molecular typing data obtained for a number of major bacterial pathogens using multilocus sequence typing (MLST), the surveillance of the distribution and spread of the fungus *Batrachochytrium dendrobatidis* which is causing widespread declines and extinctions of amphibian species, and the surveillance of drug-resistant bacteria in Europe.

We provide the facility to geocode your own spatial data (e.g. on the distribution of a pathogen, or pathogen genotypes, or the location of infectious disease outbreaks) and display it on a permanent map. Maps can take the form of a simple collection of points or can contain user-defined groupings of data allowing more complex displays – e.g. the distribution on the map of the location and extent of a disease in different years.

Should you require help please [contact us](#)

**Imperial College London**      **wellcome trust**

Contact Us | Imperial College London | Funded by The Wellcome Trust | Spatialepidemiology.net is developed by David Aanensen in the laboratory of Prof Brian Spratt

# GEO-REFERENCERS AND REPORTERS

spatialepidemiology.net

## Create User maps

Map Latitude/Longitude Lookup | Batch Geocode addresses | Simple Map Creation | Advanced Map Creation

Batch Geocoding - Here you can enter a list of addresses and each is geocoded and latitude / longitude values returned. [Instructions](#)

Please enter your list of addresses one per line - PLEASE READ INSTRUCTIONS BEFORE DOING SO

North West, UK  
West Midlands, UK  
Wales, UK  
London, UK  
South East, UK  
London, UK  
London, UK  
South East, UK  
South East, UK  
South East, UK

Start geocoding | Reset Forms

Geocoding results - Results are displayed in TAB delimited format allowing you to copy and paste directly into Excel. Six columns are given details of which can be found here.

Address	Latitude	Longitude	Accuracy	Number of Addresses Returned	Address or error code
North west, UK	41.5545253	-87.33737550000001	9	2	2135 W 35th Ave, Gary, IN 46408, USA
London, UK	51.5073509	-0.12775829999998223	4	1	London, UK
London, UK	51.5073509	-0.12775829999998223	4	1	London, UK
London, UK	51.5073509	-0.12775829999998223	4	1	London, UK
London, UK	51.5073509	-0.12775829999998223	4	1	London, UK
London, UK	51.5073509	-0.12775829999998223	4	1	London, UK
West Midlands, UK	52.4750743	-1.8298330000000078	3	1	West Midlands, UK
West Midlands, UK	52.4750743	-1.8298330000000078	3	1	West Midlands, UK
North West, UK	48.6655294	-123.40820329999997	9	2	2212 Harbour Rd, Sidney, BC V8L 2P6, Canada
North West, UK	41.5545253	-87.33737550000001	9	2	2135 W 35th Ave, Gary, IN 46408, USA
West Midlands, UK	52.4750743	-1.8298330000000078	3	1	West Midlands, UK
Wales, UK	52.1306607	-3.783711700000026	2	1	Wales, UK
London, UK	51.5073509	-0.12775829999998223	4	1	London, UK
South East, UK	0	0	0	0	Unknown Address: No corresponding geographic location could be found for the specified address.

As described on the website, this returns the address geocoding in six columns. The first three: **Address, Latitude and Longitude** are what we are after to update our metadata file.

You are also given a measure of address accuracy (for how specific the address is) and multiple addresses where a single one could not be specified.

**Copy and Paste** this information directly from the field into the spread sheet program and **manually curate** (i.e. decided between the options for each one) the address until you have a single latitude and longitude for each isolate.

Then, update the latitude and longitude columns in your metadata file.

# GEO-REFERENCERS AND REPORTERS

## Step 2: Create a map of the metadata of the isolates for your country

Although [www.spatialepidemiology.net](http://www.spatialepidemiology.net) is a complete geo-referencing tool, we are going to use some of the added functionality available in **Microreact** to visualize and explore your trees and metadata.

Microreact enables you to visualize phylogenetic relationships of isolates linked to geographic locations. Dynamic visualization of the data with interactive map, tree and metadata windows. <http://microreact.org>

Microreact allows you to link, visualise and explore your data using trees, maps and timelines.

*Streptococcus pneumoniae* PMEN2  
Croucher NJ et al. 2014. Variable recombination dynamics during

*Salmonella* Typhi  
Wong V et al. 2015. Phylogeographical analysis of the

Y-chromosome Human Phylogeny  
Hallast P et al. 2015. The Y-chromosome tree bursts into leaf:

To prepare for the next step, save your updated metadata file with GPS locations as a **.csv** by doing **File > Save as > metadata.csv**

Read the instructions on how to set format your metadata file to visualise in microreact. This is vital for the next steps.

**Note:** You can obtain more HTML colour codes at <http://htmlcolorcodes.com/>

You will need a **NEWICK (.nwk)** file from the tree builders and the **.csv** metadata file you have just saved for the next step.



# GEO-REFERENCERS AND REPORTERS

Drag the relevant **.csv** file and **.nwk** in the 'UPLOAD' section on the website.

Simply **drag and drop** files anywhere on this page, [browse for files](#), or enter file URLs:

.csv file

.nwk file

One data file (.csv or .tsv) is required and a tree file (.nwk or .newick) is optional.

CONTINUE

## .csv file?

This is your **data** file. It must contain an **id** column with a valid identifier for every row, which must be unique and must not contain full stops or commas:

Geolocations can be specified by **latitude** and **longitude** columns. You can find the latitude and longitude for a certain location using [this service](#).

Temporal data can be specified by **year**, **month** and **day** columns.

## .nwk file?

This is your **tree** file which must be in valid Newick format.

Every leaf label must correspond to an identifier that is specified in the **id** column of your **data** file

The number of labels in the **.nwk** file must match the number of identifiers within the **id** column of the **data** file.

## Create a new project

What is the name of your project: \*

metadata\_microreact

Describe your project (briefly):

What is your email address:

Your project website:

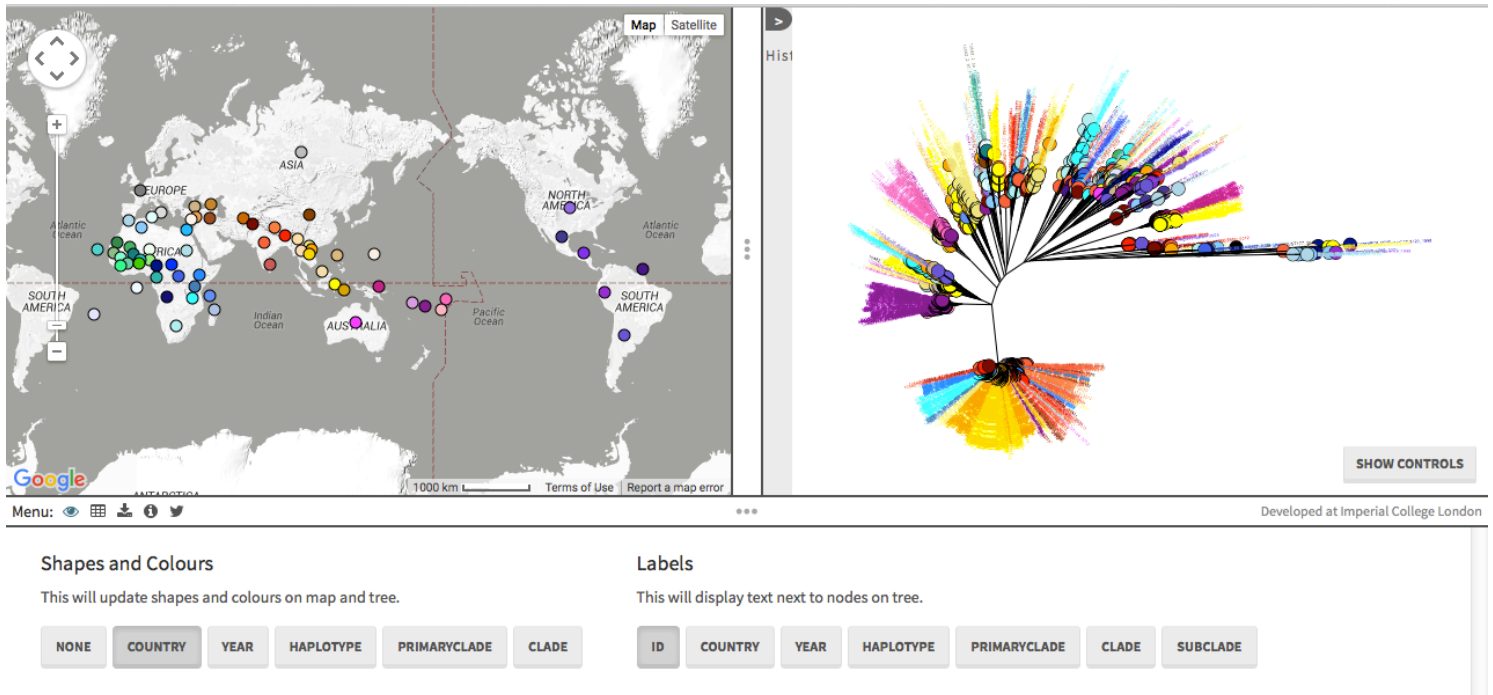
CANCEL

CREATE PROJECT

You can include a different name for your projects and a brief description if you would like.

Leave 'project website' section blank and your email is optional. Then 'create project'.

# GEO-REFERENCERS AND REPORTERS



The resulting map and tree enables you to query your data.

Look at the distribution of your isolates across the country when coloured by:

- **Clade** – how are the isolates distributed? Are there any patterns you can see to the distribution? What factors might be driving the distribution?
- **Antimicrobial resistances** – coordinate with your team antimicrobial resistance investigator for this – are there patterns to any of the resistances? And is this related to clades?

Take snap shots of the images and report your findings in the group presentation.  
Some examples are below.

# GEO-REFERENCERS AND REPORTERS

## Other geo-tagging resources

Pathogens do not respect borders and global travel is increasingly frequent. For this reason the effective tracking and tracing of pathogens internationally is more important than ever. The analysis of your *S. Typhimurium* isolates tells us about how the pathogen behaves on a city-wide scale. To see if the epidemiology and resistance patterns you observed in your HCMC translate to the global scale, we need effective collaboration. For the geo-tagging resources mentioned below, you can use either your own country data or data from the course for practice.

## Other free geo-tagging resources.

**WGSA:** Is a web application for the processing, clustering and exploration of microbial genome assemblies. You can upload your assemblies and accompanying metadata to view assembly stats and view other metadata. <https://www.wgsa.net/>

**EpiCollect:** Is a freely available web and mobile app tool that is used for data collection (questionnaires), using multiple mobile phones and the data can be centrally viewed using Google maps/ tables and charts. [www.epicollect.net](http://www.epicollect.net)

**PhyloCanvas:** Metadata in binary format can be displayed next to the tree leaves by uploading a .csv file together with the tree file. <http://phylocanvas.net>

**CartoDB:** Much like the Google maps exercise you completed, CartoDB allows the user to map and analyze location data. This tool can take multiple file formats as input e.g. XLS, CSV and SQL amongst others. <https://cartodb.com/>

**DISCLAIMER:** All the locations and dates of the *Salmonella* isolates are fictitious and solely for educational purposes. No data was collected from Public Health England.

**End of module...**

**ANY QUESTIONS?**

**Please feel free to ask at any time!**

# References

Abbot, J. C. et al. (2005) *Bioinformatics* 21(18) 3665-3666. WebACT – an online companion for the Artemis Comparison Tool.

Allen JE & Salzberg SL (2005). *Bioinformatics* 21: 3596-3603. JIGSAW: integration of multiple sources of evidence for gene prediction.

Alexa A. et al. (2006) *Bioinformatics* 22: 1600-1607. Improved scoring of functional groups from gene expression data by decorrelating GO graph structure.

Anders S & Huber W (2010) *Genome Biol* 11: R106. Differential expression analysis for sequence count data.

Anders S. HTSeq: Analysing high-throughput sequencing data with Python. 2010. Software. [<http://www-huber.embl.de/users/anders/HTSeq/>]

Assefa, S. et al. (2009) *Bioinformatics* 25 (15) 1968-9. ABACAS: algorithm-based automatic contiguation of assembled sequences.

Berriman, M., and K. Rutherford (2003) *Brief Bioinform* 4 (2) 124-132. Viewing and annotating sequence data with Artemis.

Bozdech Z. et al. (2003) *PLOS Biol* 1: E5. The transcriptome of the intraerythrocytic developmental cycle of *Plasmodium falciparum*.

Carver T. J. et al. (2010) *Bioinformatics* (doi:10.1093/bioinformatics/btq010)  
BamView: Viewing mapped read alignment data in the context of the reference sequence.

Carver T. J. et al. (2005) *Bioinformatics* 21: 3422-3. ACT: the Artemis Comparison Tool.

Conesa A, et al. (2005) *Bioinformatics* 21: 3674-3676. Blast2GO: a universal tool for annotation, visualization and analysis in functional genomics research.

- Delcher AL. et al. (1999) *Nucleic Acids Res* 27: 4636-4641. Improved microbial gene identification with GLIMMER.
- Gardner et al. (2002). *Nature* 419(6906):498-511. Genome sequence of the human malaria parasite *Plasmodium falciparum*.
- Grant GR, et al. (2011) *Bioinformatics* 27: 2518-2528. Comparative analysis of RNA-Seq alignment algorithms and the RNA-Seq unified mapper (RUM).
- Hacker, J. et al. (1997) *Mol Microbiol* 23: 1089-97. Pathogenicity islands of virulent bacteria: structure, function and impact on microbial evolution.
- Haas BJ, et al. (2008). *Genome Biol* 9: R7. Automated eukaryotic gene structure annotation using EVIDENCEModeler and the Program to Assemble Spliced Alignments.
- Hardcastle TJ & Kelly KA (2010). *BMC Bioinformatics* 11: 422. baySeq: empirical Bayesian methods for identifying differential expression in sequence count data
- Kozarewa, I., Z. Ning, et al. (2009). *Nature Met* 6(4): 291-295. Amplification-free Illumina sequencing-library preparation facilitates improved mapping and assembly of (G+C)-biased genomes.
- Langmead et al. (2009). *Genome Biol* 10:R25. Ultrafast and memory-efficient alignment of short DNA sequences to the human genome.
- Li et al. (2008a). *Genome Res* 18:1851-8. Mapping short DNA sequencing reads and calling variants using mapping quality scores.
- Li et al. (2008b). *Bioinformatics* 24(5):713-714. SOAP: short oligonucleotide alignment program.
- Li et al. (2009). *Bioinformatics*, 25:1754-60. Fast and accurate short read alignment with Burrows-Wheeler Transform.
- Majoros et al. (2003) *Nucleic Acids Res* 31 (13) 3601-3604. GlimmerM, Exonomy and Unveil: three *ab initio* eukaryotic genefinders.
- Mortazavi et al. (2008). *Nature Met* 5: 621 – 628. Mapping and quantifying mammalian transcriptomes by RNA-Seq.
- Ning et al. (2001). *Genome Res* 10:1725-9. SSAHA: a fast search method for large DNA databases.

---

Otto et al. (2010) *Mol Microbiol* Apr;76(1):12-24. New insights into the blood stage transcriptome of *Plasmodium falciparum* using RNA-Seq.

Otto, T. D., G. P. Dillon, et al. (2011). *Nucleic Acids Res* **39**(9): e57. RATT: Rapid Annotation Transfer Tool.

Otto, T. D., M. Sanders, et al. (2010). *Bioinformatics* **26**(14): 1704-1707. Iterative Correction of Reference Nucleotides (iCORN) using second generation sequencing technology.

Parkhill, J. (2002) *Method Microbiol* 33: 1-26. Annotation of microbial genomes.

Robinson MD, et al. (2010) *Bioinformatics* 26: 139-140. edgeR: a Bioconductor package for differential expression analysis of digital gene expression data.

Rutherford et al. (2000) *Bioinformatics* 16 (10) 944-945. Artemis: sequence visualization and annotation.

Simpson, J. T., K. Wong, et al. (2009). *Genome Res* **19**(6): 1117-1123. ABySS: a parallel assembler for short read sequence data.

Stephens et al. (1998). *Science* 282(5389): 754 – 759. Genome sequence of an obligate intracellular pathogen of humans: *Chlamydia trachomatis*.

Tsai, I. J., T. D. Otto, et al. (2010). *Genome Biol* **11**(4): R41. Improving draft assemblies by iterative mapping and assembly of short reads to eliminate gaps.

Trapnell et al. (2009). *Bioinformatics* 25(9):1105-1111. TopHat: discovering splice junctions with RNA-Seq.

Wang et al. (2009). *Nat Rev Genet* 10(1):57-63. RNA-Seq: A revolutionary tool for transcriptomics.

Zerbino, D. R. and E. Birney (2008). *Genome Res* **18**(5): 821-829. Velvet: algorithms for *de novo* short read assembly using de Bruijn graphs.

# Appendices



## Appendix I: Course Virtual Machine (VM) Quick Start Guide

Using a VM enables us to encapsulate the course data and software in such a way that you can still make use of them when you return to your own laboratory.

To use the VM on the USB stick provided, you will first need to download VirtualBox (<http://www.virtualbox.org/>). This software is required to run the VM on your machine, it is free and available for windows, MacOSX and linux,

For a detailed description of VirtualBox and the installation see the on-line manual (<http://www.virtualbox.org/manual/>).

### Download and Install VirtualBox

- Download VirtualBox for the type of workstation you are using (e.g. Windows) from <http://www.virtualbox.org/wiki/Downloads>.
- Double click on the executable file (Windows). The installation welcome dialog opens and allows you to choose where to install VirtualBox to, and which components to install. Depending on your Windows configuration, you may see warnings about "unsigned drivers" or similar. Please select "Continue" on these warnings; otherwise VirtualBox might not function correctly after installation.
- Launch the VirtualBox software from the desktop shortcut or from the program menu.

### Setting up the VM

VirtualBox needs to be pointed at the VDI (This is the file that is on the memory stick used during the course) file as follows:

- Insert the USB memory stick provided. This contains a Virtual Disk Image (VDI) file.

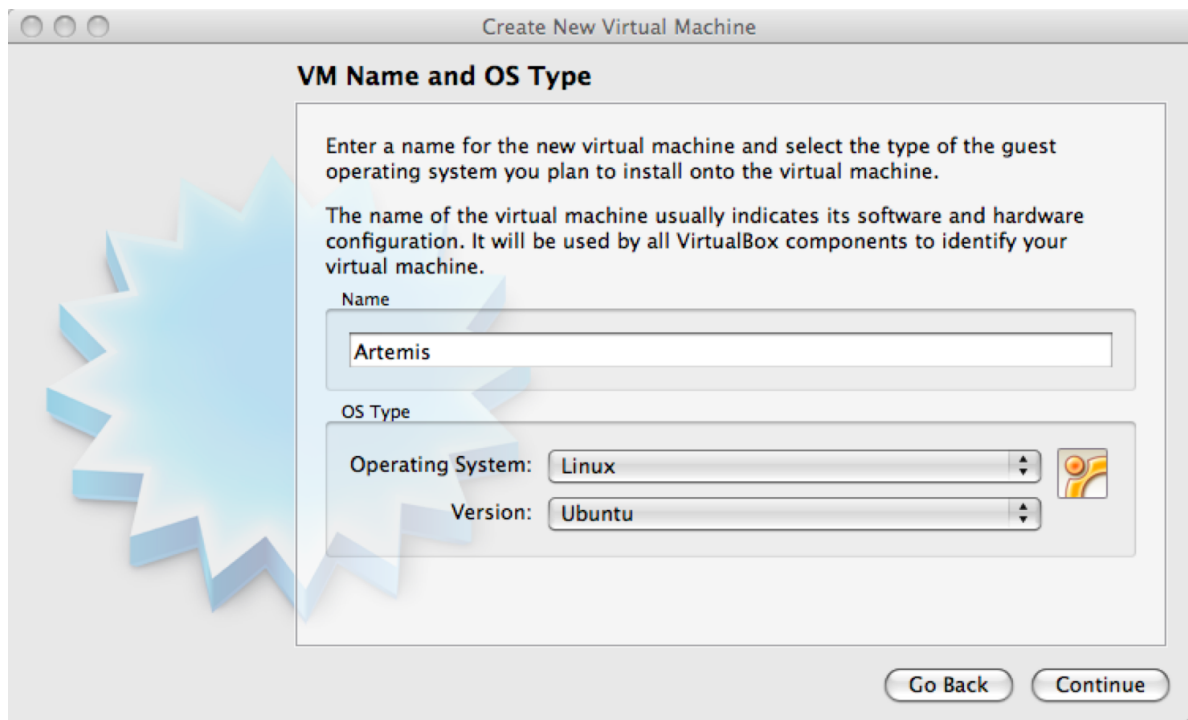
Create a new virtual machine by selecting 'New' from the options at the top. Then fill the boxes in as shown below:

In the first window enter:

Name: **Artemis**

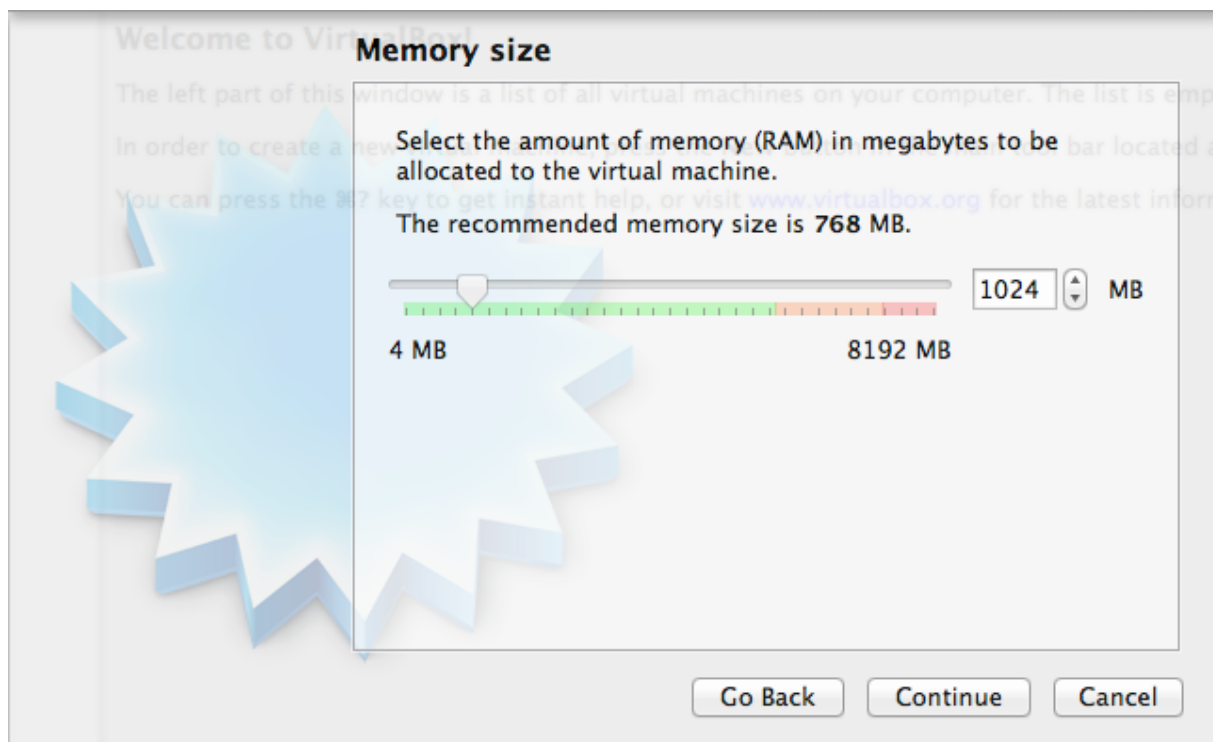
Operating System: **Linux**

Version: **Ubuntu**



Click 'Continue'

In the next window set the memory to at least 1GB (as shown), but 2GB (2048 MB) will give you better performance. You can use more but no more than half the amount of memory on your PC.



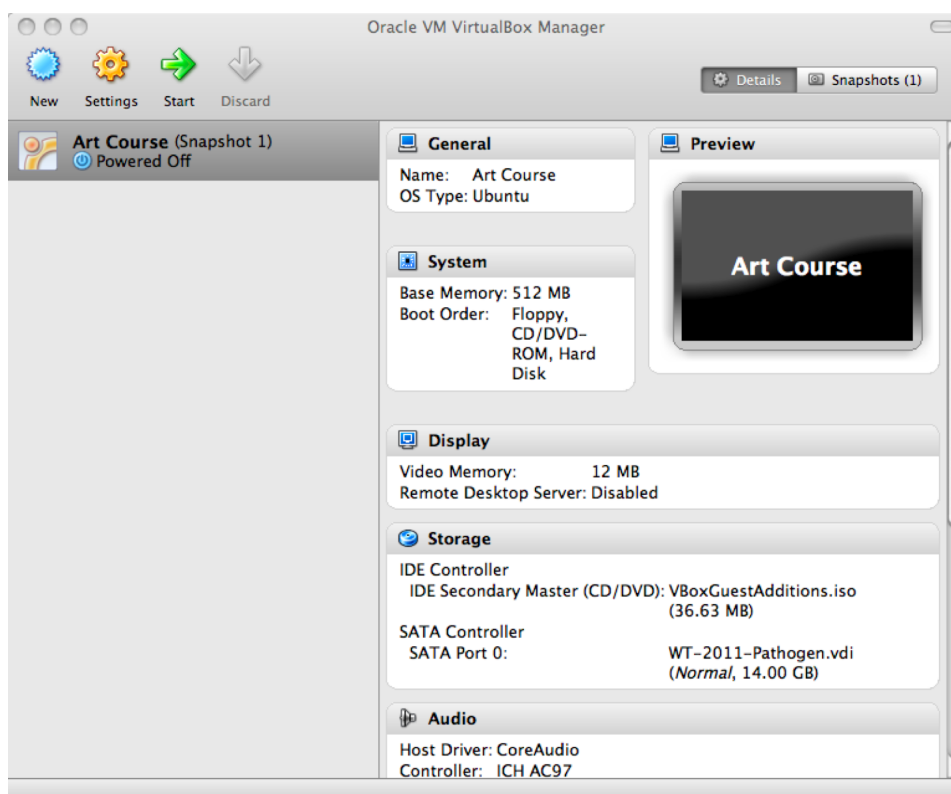
Click 'Continue'.

In the next window select 'Use existing hard disk' and from the folder icon on the right hand side navigate to the memory USB stick and select the VDI file located on the memory stick



Click 'Continue'.

There will now be an 'Artemis' (powered off) button in the left hand side of VirtualBox.



Double click on this new Artemis course power button to start the VM. It will then log you into the Ubuntu desktop.

### **Setting up a Shared Folder**

This allows you to share a folder between the VM and your workstation. This means you can put files that you want to share between the operating systems in this folder.

Create a directory to share called 'VMshare' on your machine. With the VM shutdown select the 'Artemis' button in VirtualBox and click 'Settings' in the top menu bar. Go to 'Shared Folders' and select the '+' button on the right. In the 'Folder Path' select 'Other' and navigate to and select the 'VMshare' folder that you have created. Then click on 'OK'.

When the 'Artemis' VM is next started double click on the 'mount' icon in your home folder. This will open a window that you need to type the password into:

```
wt
```

It will show the contents of this folder in the /home/wt/host directory in Ubuntu.

### **A note on memory usage:**

Some computing processes are very memory hungry. Should you find that your computer processes are killed without a clear reason, one aspect to check is the amount of memory allocated to the VM. The 1024MB you have allocated using this tutorial has been check and should be enough. Nonetheless, the amount of memory allocated to the VM can be changed at any time.

## **Appendix II: Artemis minimum hardware and software requirements.**

Artemis and ACT will, in general, work well on any standard modern machine and with most common operating systems. It is currently used on many different varieties of UNIX and Linux systems as well as Apple Macintosh and Microsoft Windows systems.

## **Appendix III: ACT comparison files**

ACT supports three different comparison file formats:

- 1) BLAST version 2.2.2 output: The blastall command must be run with the -m 8 flag which generates one line of information per HSP.
- 2) MegaBLAST output: ACT can also read the output of MegaBLAST, which is part of the NCBI blast distribution.
- 3) MSPcrunch output: MSPcrunch is program for UNIX and GNU/Linux systems which can post-process BLAST version 1 output into an easier to read format. ACT can only read MSPcrunch output with the -d flag.

Here is an example of an ACT readable comparison file generated by MSPcrunch -d.

```
1399 97.00 940 2539 sequence1.dna 1 1596 AF140550.seq
1033 93.00 9041 10501 sequence1.dna 9420 10880 AF140550.seq
828 95.00 6823 7890 sequence1.dna 7211 8276 AF140550.seq
773 94.00 2837 3841 sequence1.dna 2338 3342 AF140550.seq
```

The columns have the following meanings (in order): score, percent identity, match start in the query sequence, match end in the query sequence, query sequence name, subject sequence start, subject sequence end, subject sequence name.

The columns should be separated by single spaces.

## Appendix IV: Feature Keys and Qualifiers – a brief explanation of what they are and a sample of the ones we use.

**1 – Feature Keys:** They describe features with DNA coordinates and once marked, they all appear in the Artemis main window. The ones we use are:

**CDS:** Marks the extent of the coding sequence.

**RBS:** Ribosomal binding site

**misc\_feature:** Miscellaneous feature in the DNA

**rRNA:** Ribosomal RNA

**repeat\_region**

**repeat\_unit**

**stem\_loop**

**tRNA:** Transfer RNA

**2 – Qualifiers:** They describe features in relation to their coordinates. Once marked they appear in the lower part of the Artemis window. They describe the feature whose coordinates appear in the ‘location’ part of the editing window. The ones we commonly use for annotation at the Sanger Institute are:

**/class:** Classification scheme we use “in-house” developed from Monica Riley’s MultiFun assignments (see Appendix VI).

**/colour:** Also used in-house in order to differentiate between different types of genes and other features.

**/gene:** Descriptive gene name, eg. *ilvE*, *argA* etc.

**/label:** Allows you to label a gene/feature in the main view panel.

**/note:** This qualifier allows for the inclusion of free text. This could be a description of the evidence supporting the functional prediction or other notable features/information which cannot be described using other qualifiers.

**/product:** The assigned possible function for the protein goes here.

**/pseudo:** Matches in different frames to consecutive segments of the same protein in the databases can be linked or joined as one and edited in one window. They are marked as pseudogenes. They are normally not functional and are considered to have been mutated.

**/locus\_tag :** Systematic gene number, eg SAS1670, Sty2412 etc.

The list of keys and qualifiers accepted by EMBL in sequence/annotation submission files are list at the following web page:

<http://www3.ebi.ac.uk/Services/WebFeat/>

## Appendix V: Generating ACT comparison files using BLAST

The following pages demonstrate how you can generate your own comparison files for ACT from a stand-alone version of the BLAST software. In Appendix X the NCBI BLAST distribution was downloaded onto a PC with Windows XP. The exercises in this module are based on the Linux version of the BLAST software. Although the operating systems are different, the command lines used to run the programs are the same. One of the main differences between the two operating systems is that in Windows the BLAST program command line is run in the DOS Command Prompt window, whereas in Linux it is run from a Xterminal window.

In the exercises below you are going to download two small sequences (plasmids), and for two large sequences (whole genomes). You are then going to generate files containing DNA sequences in FASTA format for these sequences, which will then be compared using two different programs from the NCBI BLAST distribution to generate ACT comparison files.

### Exercise 1

In this exercise you are going to download two plasmid sequences in EMBL format from the EBI genomes web page. You are then going to use Artemis to write out the DNA sequences of both plasmids in FASTA format. These two FASTA format sequences will then be compared using the blastall program from the NCBI BLAST distribution. Using blastall you can run BLASTN to identify regions of DNA-DNA similarity and write out a ACT readable comparison file. If required, blastall can also be used to run other flavours of BLAST with the appropriate input files (i.e. DNA files for TBLASTX, protein files for BLASTP, and protein and DNA for BLASTX). For the purpose of generating ACT comparison files BLASTN and TBLASTX are appropriate.

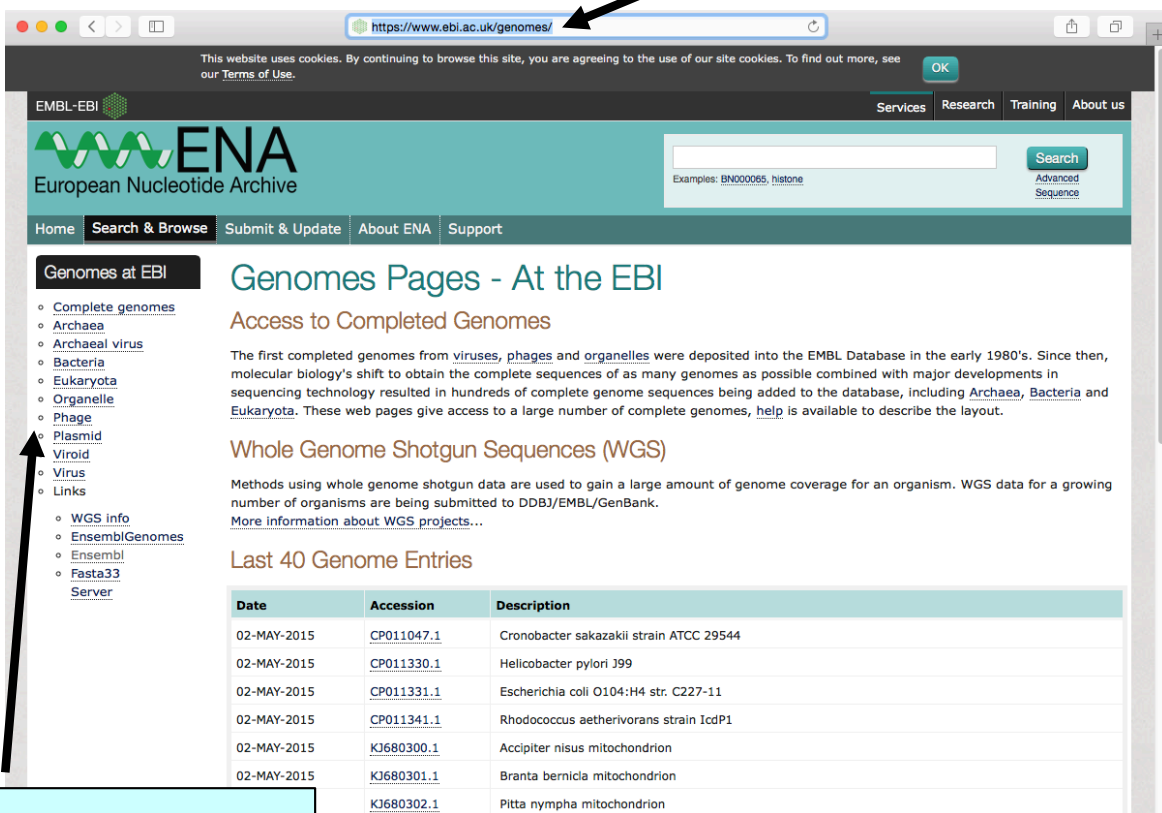
In this example two relative small sequences have been chosen (<500 kb). BLAST running on a relatively modern stand alone machine can easily deal with required computations, and thus the comparison file should be produced in a matter of seconds. However as the size of the compared sequences increases the time taken to produce the output will dramatically increase. Therefore for very large sequences (several Mb) it will be impractical to run them using blastall. In **Exercise 2** you will use megablast, another program in the NCBI BLAST distribution, which is useful for comparing large sequences that are very similar.

The plasmids chosen for this comparison are the multiple drug resistance incH1 plasmid pHCM1 from the sequenced strain of *Salmonella typhi* CT18 originally isolated in 1993, and R27, another incH1 plasmid first isolated from *S. typhi* in the 1960s.

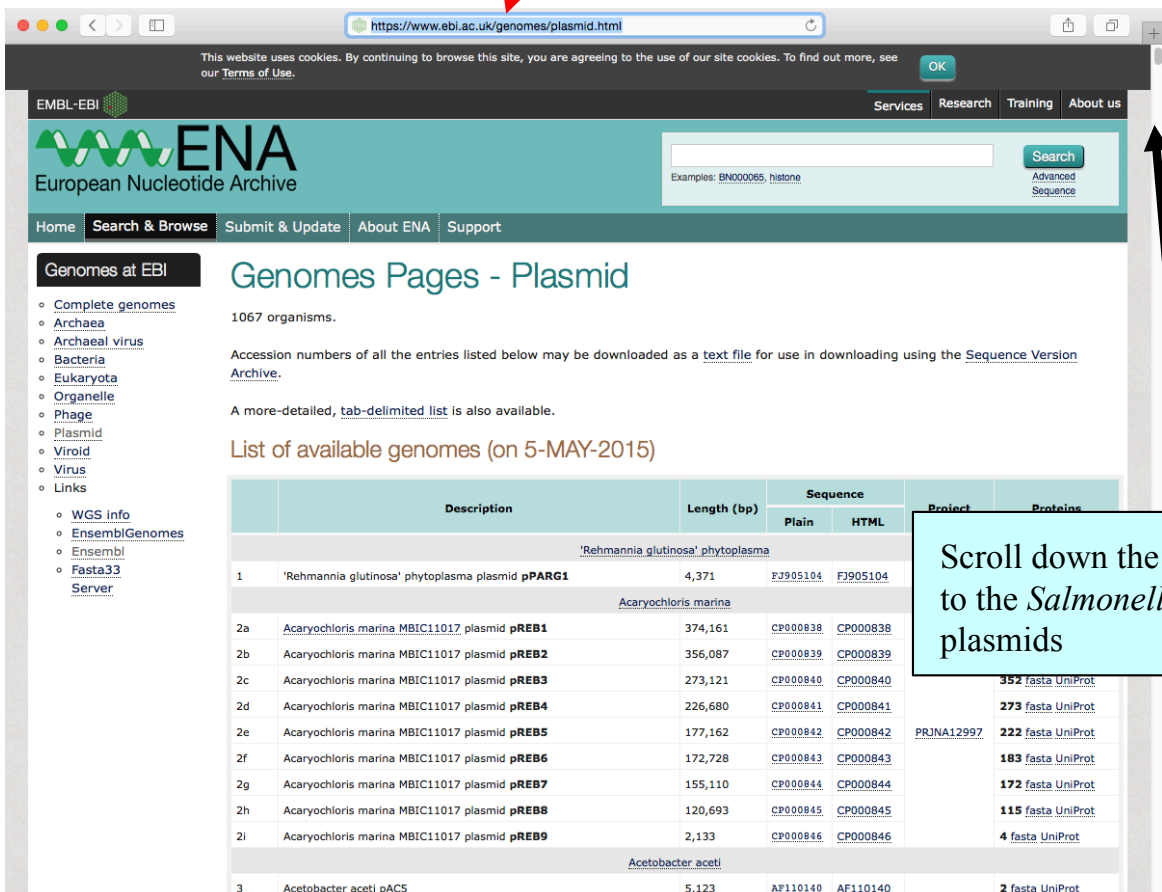


# Downloading the *S. typhi* plasmid sequences

Go to the EBI genomes web page (<http://www.ebi.ac.uk/genomes>)



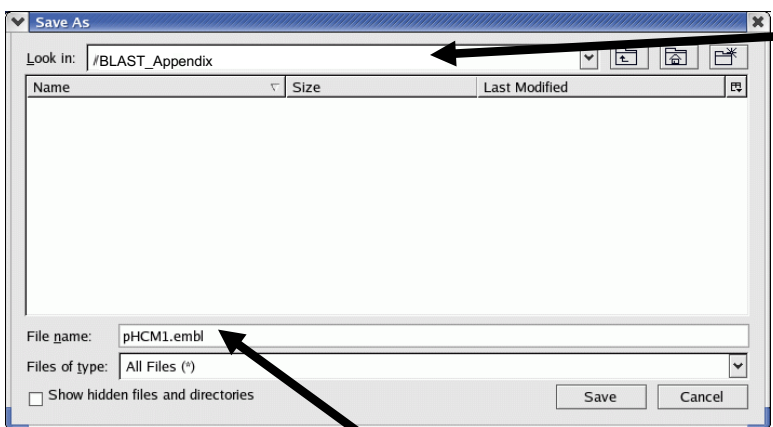
Click on the Plasmid hyperlink



Scroll down the page to the *Salmonella* plasmids

Press the Shift key and left Click on the accession number hyperlink for pHCM1 (AL513383) in the Plain Sequence column

Accession	Plasmid Name	Size (bp)	Accession	Accession	FASTA SRS
<b>Riemerella anatipestifer</b>					
158a	Riemerella anatipestifer plasmid pCFC1	3,966	AF048718	AF048718	4 FASTA SRS
158b	Riemerella anatipestifer plasmid pCFC2	5,609	AF082180	AF082180	3 FASTA SRS
<b>Ruminococcus flavefaciens</b>					
159	Ruminococcus flavefaciens R13e2 cryptic plasmid pBAW301	1,768	U22411	U22411	1 FASTA SRS
<b>Salmonella choleraesuis</b>					
160	Salmonella choleraesuis strain 79500 plasmid pSFD10	4,071	AY048853	AY048853	6 FASTA SRS
<b>Salmonella enterica</b>					
161	Salmonella enterica subsp. enterica serovar Berta plasmid pBERT	4,656	AF025795	AF025795	9 FASTA SRS
162a	Salmonella enterica subsp. enterica serovar Typhi str. CT18 plasmid pHCM1	218,160	AL513383	AL513383	234 FASTA SRS
162b	Salmonella enterica subsp. enterica serovar Typhi str. CT18 plasmid pHCM2	106,516	AL513384	AL513384	132 FASTA SRS
163	Salmonella enterica subsp. enterica serovar Typhimurium plasmid pFPTB1	12,656	AJ634602	AJ634602	6 FASTA SRS
<b>Salmonella enteritidis</b>					
164a	Salmonella enteritidis serovar Enteritidis plasmid pC	5,269	AY079201	AY079201	4 FASTA SRS
164b	Salmonella enteritidis serovar Enteritidis plasmid pK	4,245	AY079200	AY079200	3 FASTA SRS
164c	Salmonella enteritidis serovar Enteritidis plasmid pP	4,301	AY079199	AY079199	3 FASTA SRS
<b>Salmonella typhi</b>					
165a	Salmonella typhi R27 plasmid	180,461	AF250878	AF250878	204 FASTA SRS
165b	Salmonella typhi plasmid R27	38,245	AF105019	AF105019	34 FASTA SRS
<b>Salmonella typhimurium</b>					



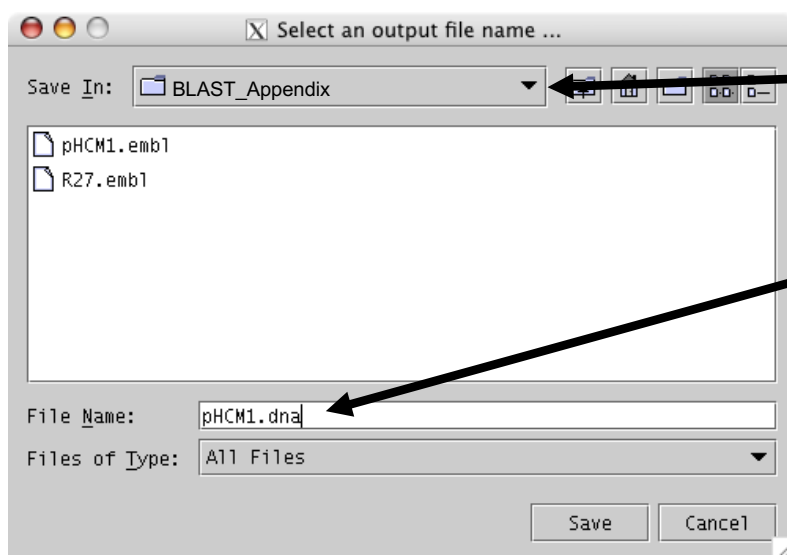
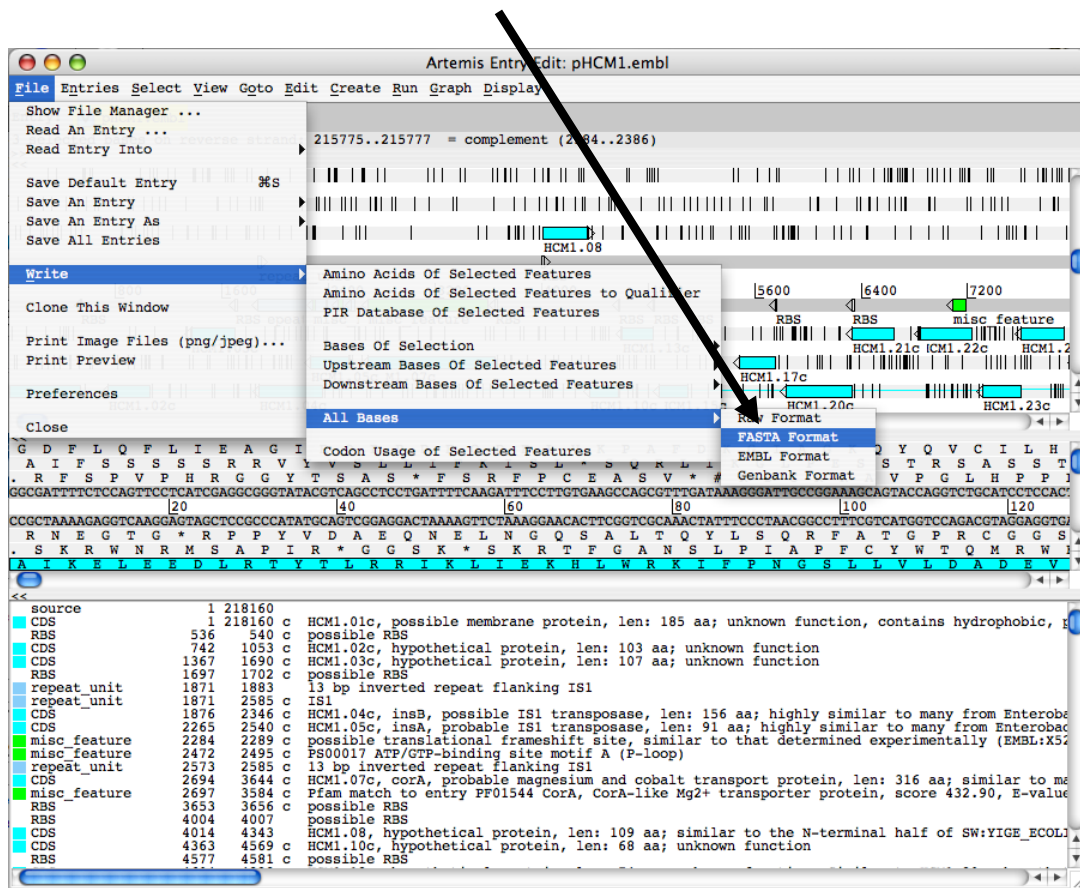
Save the EMBL sequence in a suitable directory. For example: BLAST\_Appendix

Save the file as pHCM1.embl

Repeat for the *Salmonella typhi* R27 plasmid (AF250878). Be careful when choosing the plasmid to download as there is also a *Salmonella typhi* plasmid R27 entry (AF105019), the one that you want is the larger of the two, 180,461 kb as opposed to 38,245 kb – make sure the accession number is correct. Save as R27.embl.

In order to run BLASTN you require two DNA sequences in FASTA format. The pHCM1 and R27 sequences previously downloaded from the EBI are EMBL format files, i.e. they contain protein coding information and the DNA sequence. In order to generate the DNA files in FASTA format, Artemis can be used as follows.

Load up the plasmid EMBL files in **Artemis** (each plasmid requires a separate Artemis window), select **Write, All Bases, FASTA format**.



Save the DNA sequence in the BLAST\_Appendix directory

Save as pHCM1.dna

Also do this for R27.embl

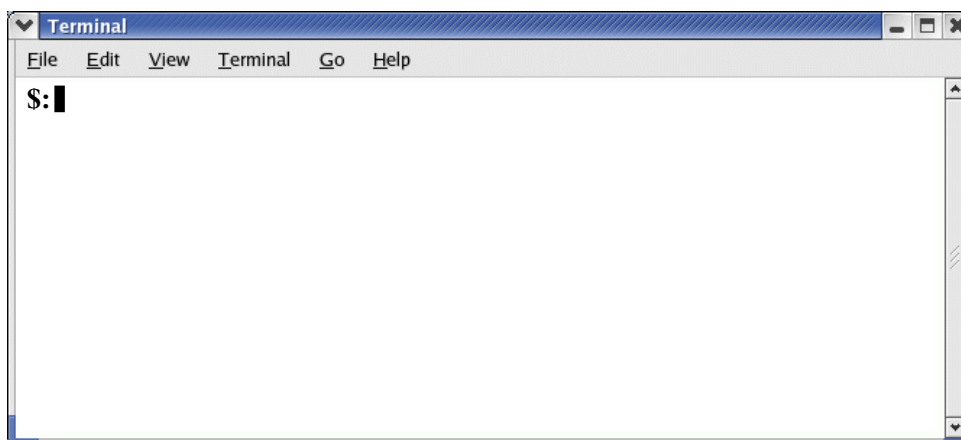
## Running Blast

There are several programs in the BLAST package that can be used for generating sequence comparison files. For a detailed description of the uses and options see the appropriate README file in the BLAST software directory (see Appendix X).

In order to generate comparison files that can be read into ACT you can use the **blastall** program running either BLASTN (DNA-DNA comparison) or TBLASTX (translated DNA-translated DNA comparison) protocols.

As an example you will run a BLASTN comparison on two relatively small sequences; the pHCM1 and R27 plasmids from *S. typhi*. In principle any DNA sequences in FASTA format can be used, although size becomes an issue when dealing with sequences such as whole genomes of several Mb (see **Exercise 2** in this module). When obtaining nucleotide sequences from databases such as EMBL using a server such as SRS (<http://srs.ebi.ac.uk>), it is possible to specify that the sequences are in FASTA format.

To run the BLAST software you will need an Xterminal window like the one below. If you do not already have one opened, you can open a new window by clicking on the Xterminal icon on the menu bar at the bottom of your screen.



Make sure you are in the appropriate directory (in this example it is BLAST\_Appendix.) You should now see both the new FASTA files for the pHCM1 and R27 sequences in the BLAST\_Appendix directory as well as their respective EMBL format files. (Hint: You can use the **pwd** command to check the present working directory, the **cd** command to change directories, and the **ls** command will list the contents of the present working directory).

When comparing sequences in BLAST, one sequence is designated as a **database** sequence, and the other the **query** sequence. Before you run BLAST you have to format one of the sequences so that BLAST recognises it as a database sequence. **formatdb** is a program that does this and comes as part of the NCBI BLAST distribution.

You will treat pHCM1.dna as the **database** sequence and R27.dna as the **query** sequence

At the Command Prompt type:  
**formatdb -i pHCM1.dna -p F**

Press **Return**

**formatdb** is the database format program

```
Terminal
File Edit View Terminal Go Help
$: formatdb -i pHCM1.dna -p F
```

**-i** designates the input sequence: pHCM1.dna

**-p** designates the sequence type: DNA is F (protein would be T)

Now you can run the BLAST on the two plasmid sequences. The program that you are going to use is **blastall**. In addition to the standard command line inputs we have to add an additional flag (**-m 8**) to the command line so that the BLAST output can be read by ACT. This specifies that the output of BLAST is in one line per entry format (see appendix II).

At the Command Prompt type:

**blastall -p blastn -m 8 -d pHCM1.dna -i R27.dna -o pHCM1\_vs\_R27**

Press **Return**

**tblastx** could be substituted here if a translated DNA-translated DNA comparison was required

**-o** designates the output file: pHCM1\_vs\_R27

**blastall** is the BLAST program

**-p** designates the flavour of BLAST: **blastn** (in this instance a DNA-DNA comparison)

**-m 8** designates the ACT readable output

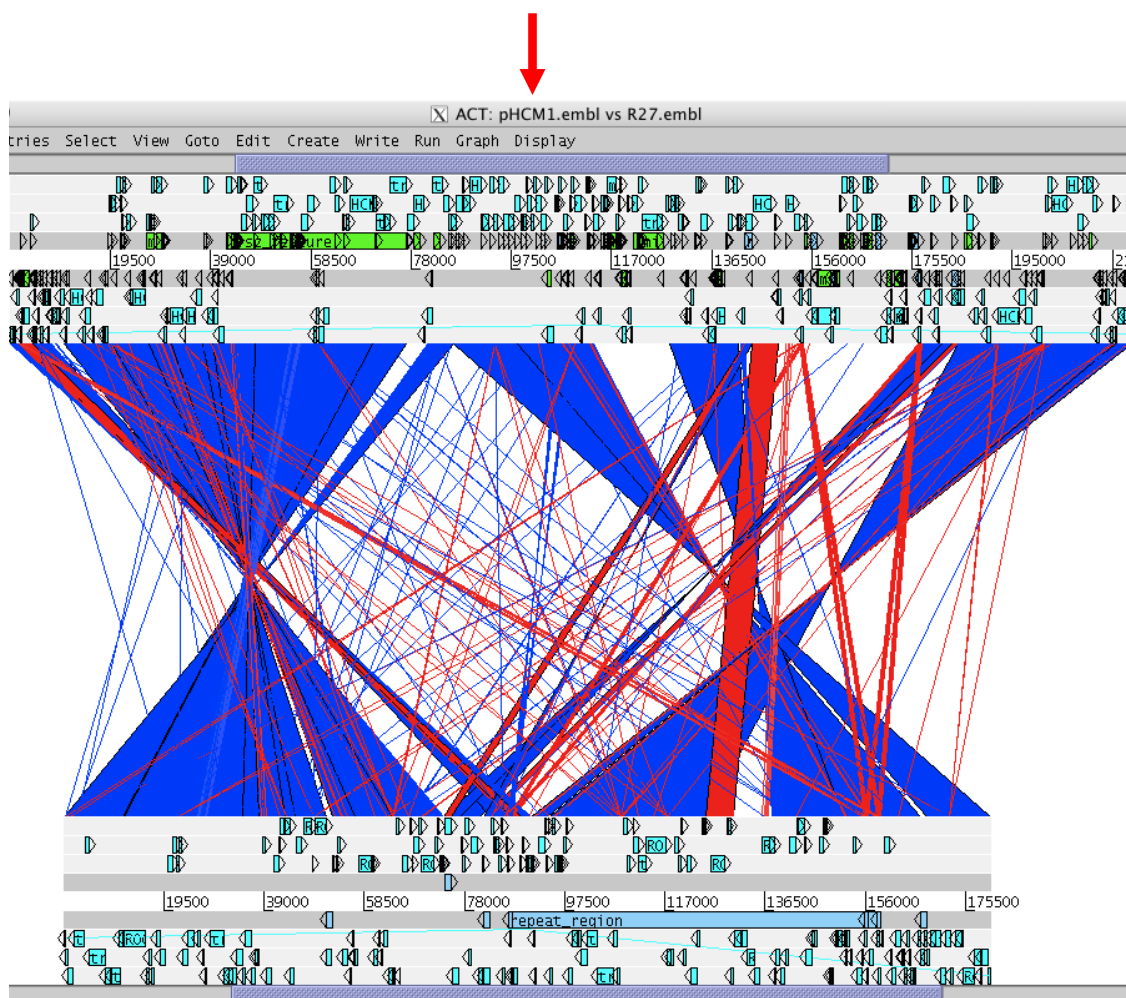
**-d** designates the database sequence: pHCM1.dna

**-i** designates the query sequence: R27.dna

```
Terminal
File Edit View Terminal Go Help
$: blastall -p blastn -m 8 -d pHCM1.dna -i R27.dna -o pHCM1_vs_R27
```



The pHCM1\_vs\_R27 comparison file can now be read into ACT along with the pHCM1.embl and R27.embl (or pHCM1.dna and R27.dna) sequence files.



The result of the BLASTN comparison shows that there are regions of DNA shared between the plasmids; pHCM1 shares 169 kb of DNA at greater than 99% sequence identity with R27. Much of the additional DNA in the pHCM1 plasmid appears to have been inserted relative to R27 and encodes functions associated with drug resistance. What antibiotic resistance genes can you find in the pHCM1 plasmid that are not found in R27?

The two plasmids were isolated more than 20 years apart. The comparison suggests that there have been several independent acquisition events that are responsible for the multiple drug resistance seen in the more modern *S. typhi* plasmid.

## Exercise 2

In the previous exercise you used BLASTN to generate a comparison file for two relatively small sequences (>500,000 kb). In the next exercise we are going to use another program from NCBI BLAST distribution, **megablast**, that can be used for nucleotide sequence alignment searches, i.e. DNA-DNA comparisons. If you are comparing large sequences such as whole genomes of several Mb, the **blastall** program is not suitable. The BLAST algorithms will struggle with large DNA sequences and therefore the processing time to generate a comparison file will increase dramatically.

**megablast** uses a different algorithm to BLAST which is not as stringent which therefore makes the program faster. This means that it is possible to generate comparison files for genome sequences in a matter of seconds rather than minutes and hours.

There are some drawbacks to using this program. Firstly, only DNA-DNA alignments (BLASTN) can be performed using **megablast**, rather than translated DNA-DNA alignments (TBLASTX) as can be using **blastall**. Secondly as the algorithm used is not as stringent, **megablast** is suited to comparing sequences with high levels of similarity such as genomes from the same or very closely related species.

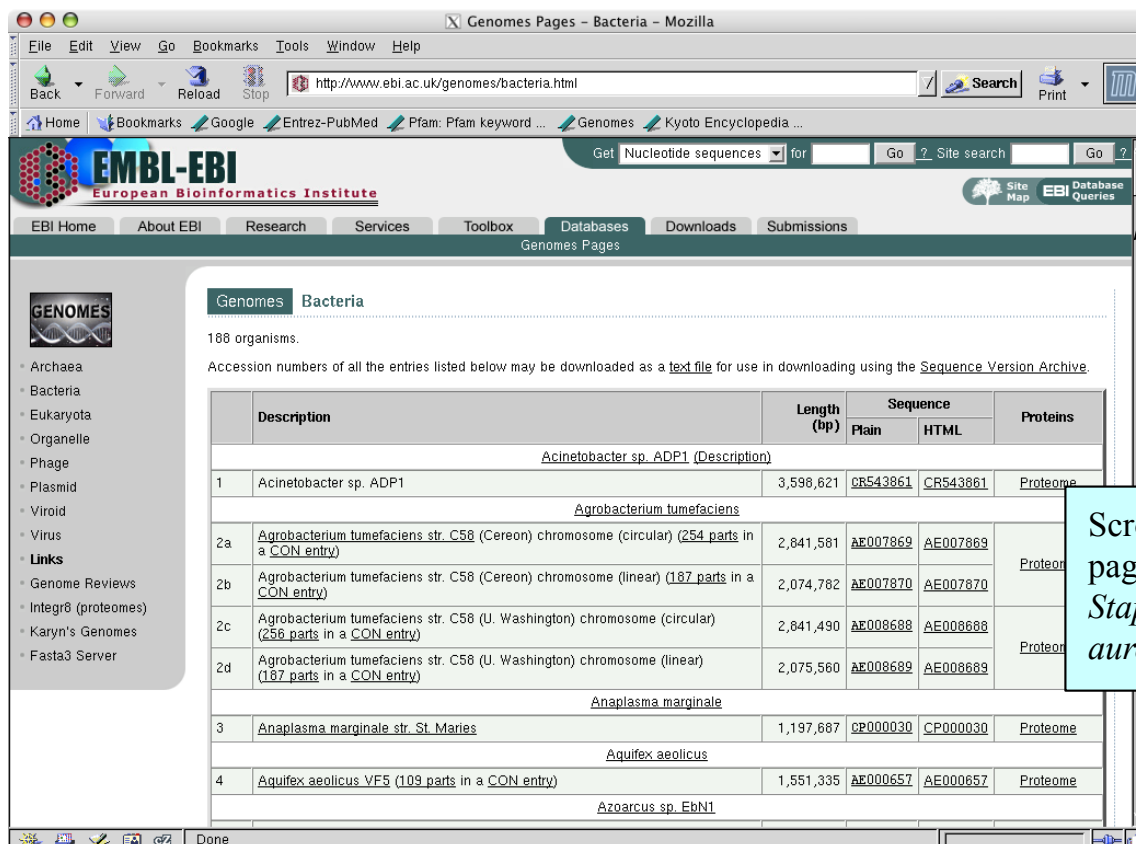
In this exercise you are going to download two *Staphylococcus aureus* genome sequences from the EBI genomes web page and use Artemis to write out the FASTA format DNA sequences for both as before in **Exercise 1**. These two FASTA format sequences will then be compared using **megablast** to identify regions of DNA-DNA similarity and write out an ACT readable comparison file.

The genomes that have been chosen for this comparison are from a hospital-acquired methicillin resistant *S. aureus* (MRSA) strain N315 (BA000018), and a community-acquired MRSA strain MW2 (BA000033).

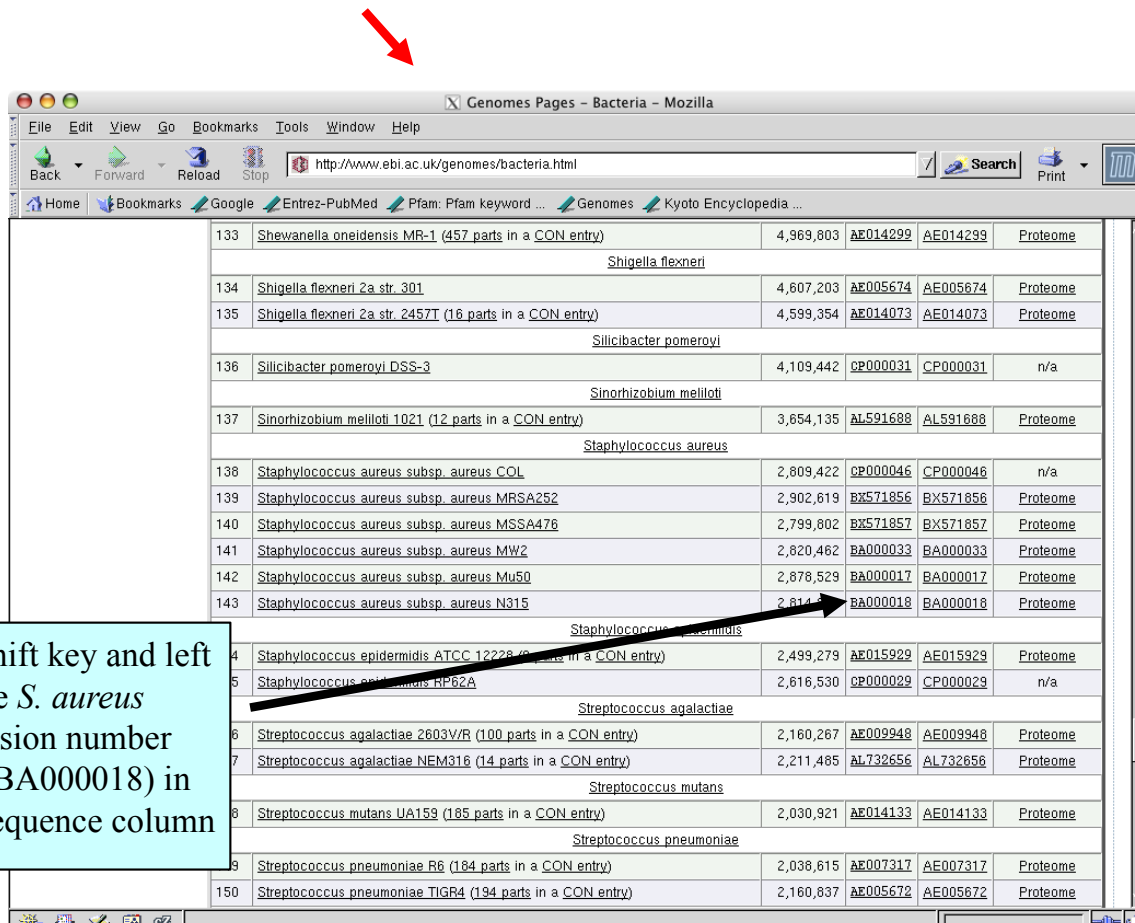


## Downloading the *S. aureus* genomic sequences

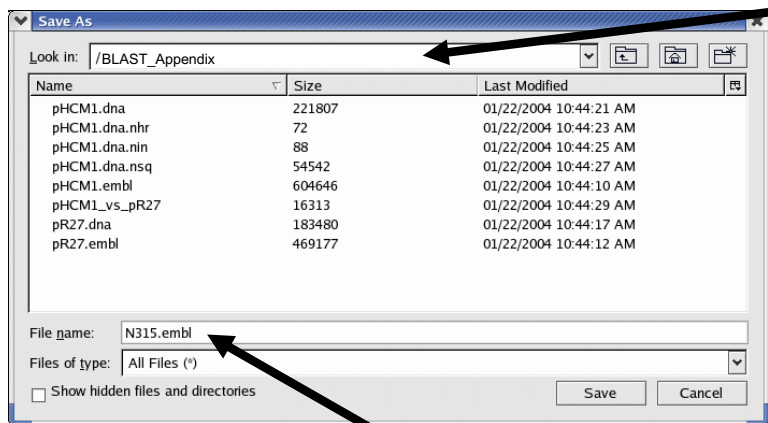
Go to the EBI genomes web page (<http://www.ebi.ac.uk/genomes>) as before in Exercise 2, and click on the **Bacteria** hyperlink



Scroll down the page to the *Staphylococcus aureus* genomes



Press the Shift key and left Click on the *S. aureus* N315 accession number hyperlink (BA000018) in the Plain Sequence column



Save the EMBL sequence in a suitable directory. For example: BLAST\_Appendix

Save the file as N315.embl

Repeat for the *S. aureus* MW2 genome (BA000033). Be careful when choosing the genome to download as there is another *S. aureus* genome entry for strain Mu50 (BA000017). Save as MW2.embl.

Generate DNA files in FASTA format using Artemis for both the genome sequences as previously done in exercise 1.

(Hint: In **Artemis** (each genome requires a separate Artemis window), select **Write, Write All Bases, FASTA format**).

Save the DNA sequences as N315.dna and MW2.dna for the respective genomes.

## Running Blast

In the previous exercise you used the **blastall** program to run BLASTN on two plasmid sequences. As the genome sequences are larger (~2.8 Mb) you are going to run **megablast**, another program from the NCBI BLAST distribution that can generate comparison files in a format that ACT can read (see Appendix II). For a detailed description of the uses and options in **megablast** see the megablast README file in the BLAST software directory (Appendix X).

As before you will run the program from the command line in an Xterminal window.

Like BLAST, **megablast** requires that one sequence is designated as a **database** sequence and the other the **query** sequence. Therefore one of the sequences has to be formatted so that Blast recognises it as a database sequence. This can be done as before using **formatdb**.

We will treat N315.dna as the **database** sequence and MW2.dna as the **query** sequence

At the Command Prompt type:  
**formatdb -i N315.dna -p F**

Press **Return**

```
Terminal
File Edit View Terminal Go Help
$: formatdb -i N315.dna -p F
```

**-i** designates the input sequence: N315.dna

**-p** designates the sequence type: DNA is F (protein would be T)

Now we can run the **megablast** on the two MRSA genome sequences. The default output format is one line per entry that ACT can read, therefore there is no need to add an additional flag (i.e. -m 8) to the command line (see appendix II).

At the Command Prompt type:

**megablast -d N315.dna -i MW2.dna -o N315\_vs\_MW2**

Press **Return**

**megablast** is the program

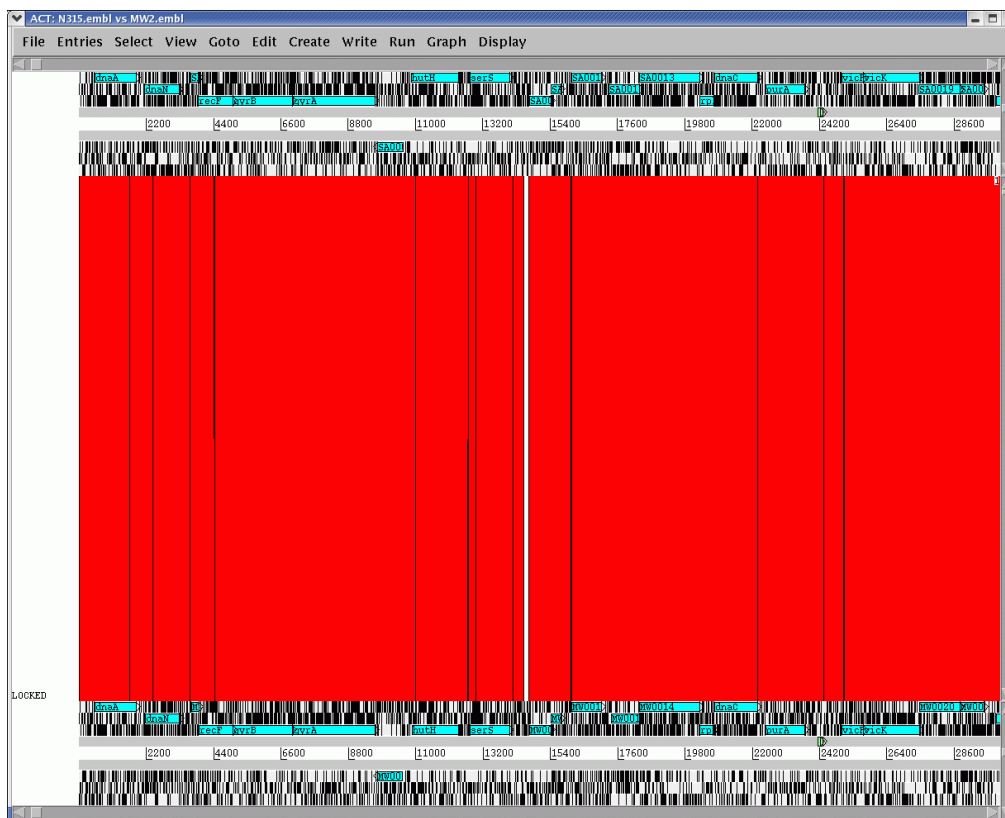
**-o** designates the output file:  
N315\_vs\_MW2

```
Terminal
File Edit View Terminal Go Help
$: megablast -d N315.dna -i MW2.dna -o N315_vs_MW2
```

**-d** designates the database sequence: N315.dna

**-i** designates the query sequence: MW2.dna

The N315\_vs\_MW2 comparison file can now be read into ACT along with the N315.embl and MW2.embl (or N315.dna and MW2.dna) sequence files.



A comparison of the N315 and MW2 genomes in ACT using the **megablast** comparison reveals a high level of synteny (conserved gene order). This is perhaps not unsurprising as both genomes belong to strains of the same species. Using results of comparisons like these it is possible to identify genomic differences that may contribute to the biology of the bacteria and also investigate mechanisms of evolution.

Both N315 and MW2 are MRSA, however N315 is associated with disease in hospitals, and MW2 causes disease in the community and is more invasive. Scroll rightward in both genomes to find the first large region of difference. Examine the annotation for the genes in these regions. What are the encoded functions associated with these regions? What significance does this have for the evolution of methicillin resistance in these two *S. aureus* strains from clinically distinct origins?

## Appendix VI – Generating Artemis comparison files using WebACT

### Introduction

If you do not have access to BLAST software running on a local computer, there is a web resource WebACT (Appendix VII for the URL) that can be used for generating ACT comparison files. WebACT allows you to cut and paste, or upload, your own sequences, and generate ACT readable BLASTN or TBLASTX comparison files. WebACT also has a large selection of recomputed comparison files for bacterial genomes, which can be downloaded along with the EMBL sequence entries and viewed in ACT.

For the purposes of this exercise we are going to focus on the Gram-negative bacterial pathogens *Burkholderia pseudomallei* and *Burkholderia mallei*. Both of these organisms are category B bio-threat agents and cause the diseases Melioidosis and Glanders respectively. The two species are closely related (DNA-DNA identity is >99%, multi locus sequence typing (MLST) predicts that *B. mallei* is a clone of *B. pseudomallei*), however they differ markedly in the environmental niches that they occupy.

*B. pseudomallei* is found in S.E. Asia and northern Australia, and is prevalent in the soil in Melioidosis endemic areas. Inhalation, or direct contact with cuts or breaks in the skin, by soil-borne *B. pseudomallei* is the cause of Melioidosis in humans and higher mammals. In contrast, *B. mallei* is a zoonotic pathogen that is host restricted to horses and cannot be isolated from the environment. Comparative genomic analysis has provided insights into evolution of these two pathogens and the genetic basis for ecological and pathological differences of these two pathogens.

The genomes of these two organisms both consist of two circular chromosomes. Comparisons of the genomes reveals that the genome of *B. pseudomallei* is ~1.31 Mb larger than that of *B. mallei*; 16% of chromosome 1, and 32% of chromosome 2, are unique in *B. pseudomallei* with respect to *B. mallei*.

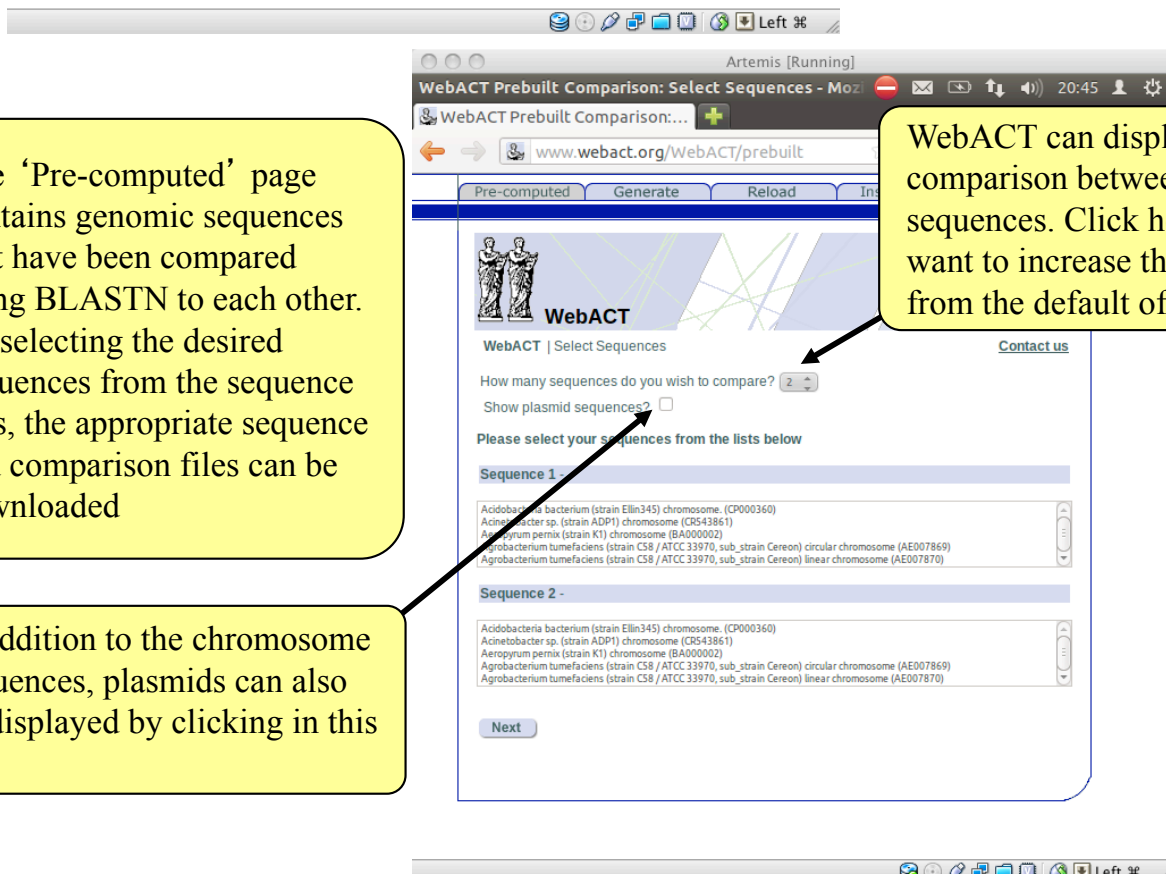
### Aim

You are going to use a web resource, WebACT, to generate a comparison file of the smaller chromosomes of *B. pseudomallei* and *B. mallei*. From the WebACT site you will download a pre-computed ACT comparison comparison file, along with the appropriate EMBL sequence and annotation files, which you will then open in ACT. Using this comparison you can then investigate some of the the genotypic differences that differentiate these closely related pathogens, and look for the basis of structural differences in these chromosomes. We have not provided files for this exercise - you are on your own.

Open up a web browser and go to the URL: [www.webact.org](http://www.webact.org)



Click on the Pre-computed tab.



The 'Pre-computed' page contains genomic sequences that have been compared using BLASTN to each other. By selecting the desired sequences from the sequence lists, the appropriate sequence and comparison files can be downloaded

WebACT can display pairwise comparison between up to 5 sequences. Click here if you want to increase the number from the default of 2.

In addition to the chromosome sequences, plasmids can also be displayed by clicking in this box



You are going to compare the smaller chromosomes of *B. pseudomallei* and *B. mallei*.

Artemis [Running]  
WebACT Prebuilt Comparison: Select Sequences - Mozilla  
www.webact.org/WebACT/prebuilt

Pre-computed Generate Reload Instructions

WebACT | Select Sequences

How many sequences do you wish to compare? 2

Show plasmid sequences?

Please select your sequences from the lists below

**Sequence 1 -**

- Burkholderia pseudomallei (strain K96243) chromosome 1 (BX571965)
- Burkholderia pseudomallei (strain K96243) chromosome 2 (BX571966)
- Burkholderia sp. (strain ATCC 17760 / NCIB 9086 / R18194 / 383) / 383) chromosome 1. (CP000151)
- Burkholderia sp. (strain ATCC 17760 / NCIB 9086 / R18194 / 383) / 383) chromosome 2. (CP000152)
- Burkholderia sp. (strain ATCC 17760 / NCIB 9086 / R18194 / 383) / 383) chromosome 3. (CP000150)

**Sequence 2 -**

- Burkholderia cenocepacia (strain AU 1054) chromosome 2. (CP000379)
- Burkholderia cenocepacia (strain AU 1054) chromosome 3. (CP000380)
- Burkholderia mallei (strain ATCC 23344) chromosome 1. (CP000010)
- Burkholderia mallei (strain ATCC 23344) chromosome 2. (CP000011)
- Burkholderia pseudomallei (strain 1710b) chromosome 1. (CP000124)

Next

In the **Sequence 1** list select *Burkholderia pseudomallei* chromosome 2 (accession number BX571966)

In the **Sequence 2** list select *Burkholderia mallei* chromosome 2 (accession number CP000011)

Once you have selected the sequences click the **Next** button

Artemis [Running]  
WebACT Prebuilt Comparison: Select Regions - Mozilla  
www.webact.org/WebACT/prebuilt

Pre-computed Generate Reload Instructions

WebACT | Select Sequences | Select Region

Do you wish to...

- Set the same range for all sequences?
- Set a different range for each sequence?

Select the sequence range to display

2 sequences selected

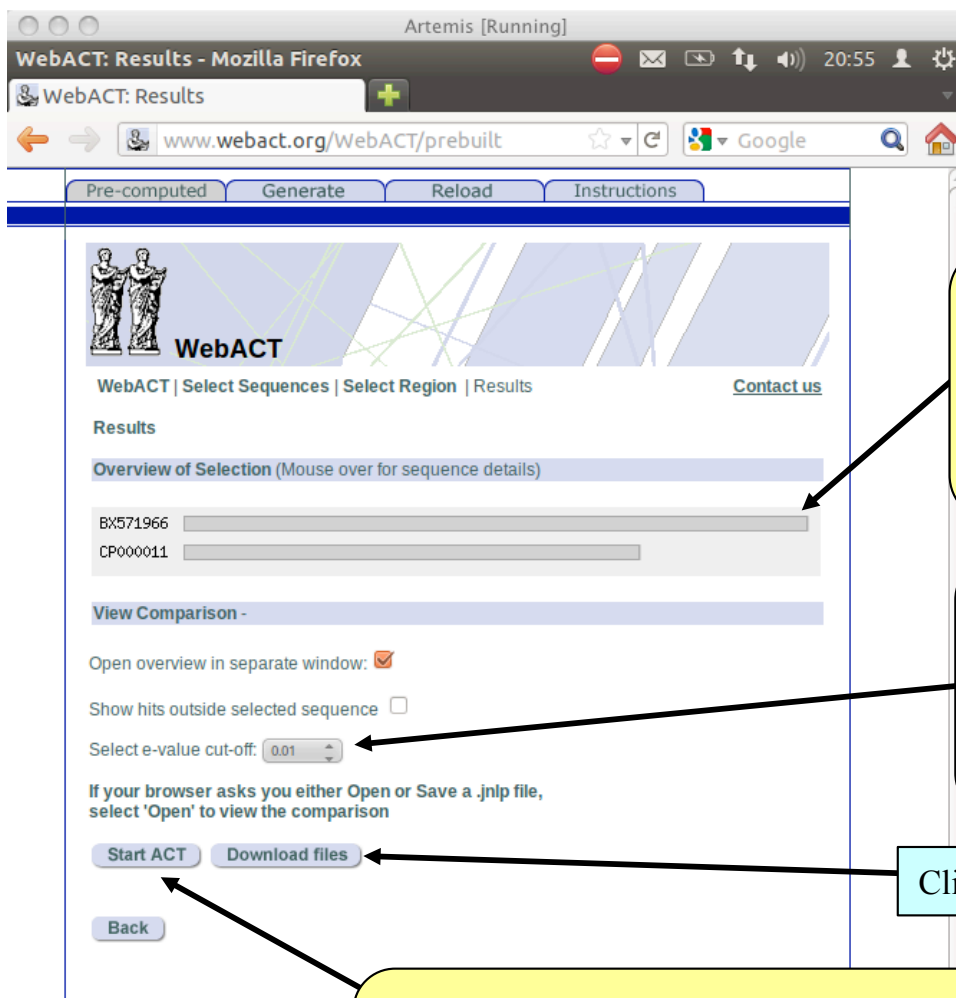
- Full sequence
- gene name -  Browse and 50000 bp of flanking sequence.
- From: 1 To: 100000

Back Next

In this window you can specify the regions in the selected sequences to generate the comparison over. It is possible to query the sequences on gene name or coordinates. The default setting is for the whole sequence, and this is what we want for this exercise as you are going to compare the whole chromosomes.

Click the **Next** button





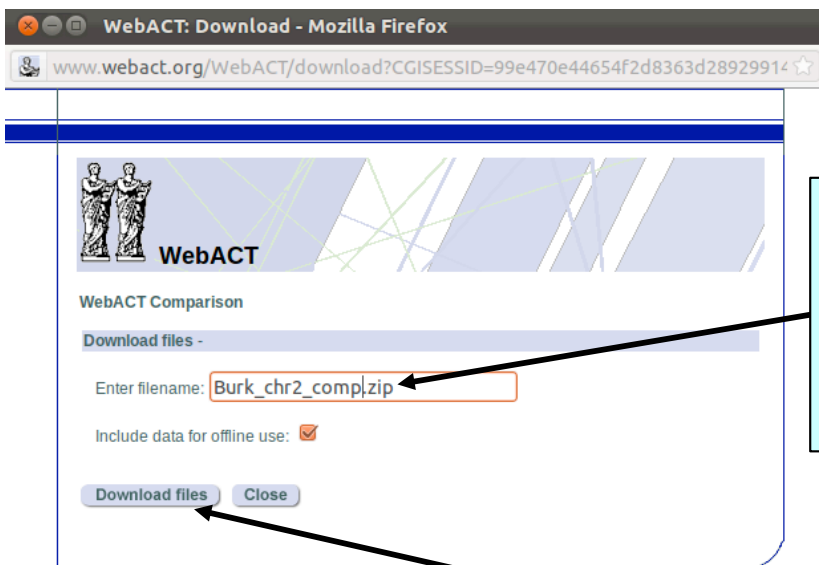
In the **Overview of Selection** you can see a schematic representation of the relative size of the two sequence that have been chosen to be compared.

The Expect (E) value cut-off can be changed in this box. The default value is 0.01, but the range is from 10.0 to 0.0001.

Click the **Download files** button

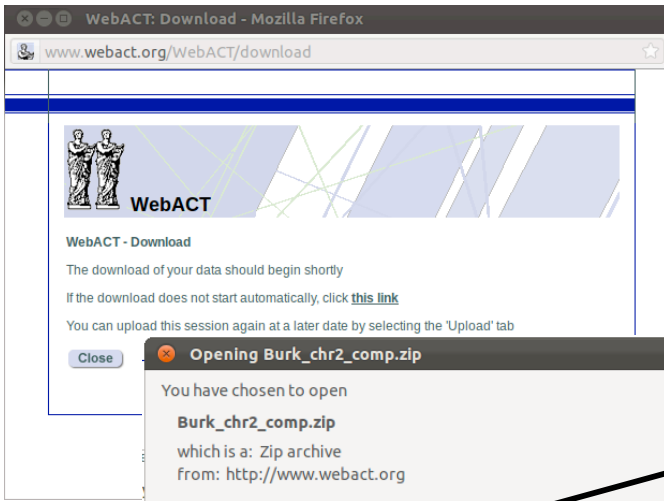
In addition to downloading the comparison files and sequence file it is also possible to view the comparison in a webstart version of ACT. This will run locally on your machine and does not require ACT to be previously loaded, as a webstart version of ACT will be included in the download. You are not going to use this option in this exercise.

The comparison file and sequences files will be contained in a folder. For the ease of downloading the folder is zipped.



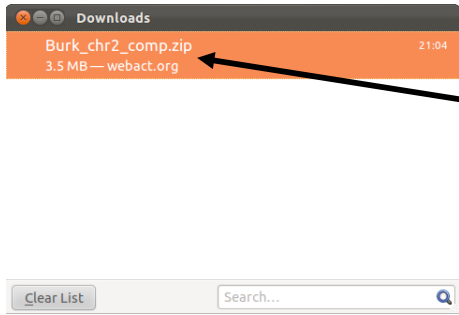
In the filename box you can type the file name of the zip file containing the sequence and comparison files. For this exercise call the file: **Burk\_chr2\_comp.zip**

Click the **Download files** button



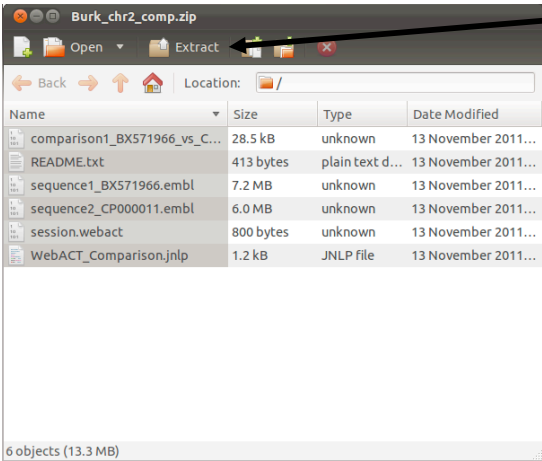
You may get a window appearing asking you what Firefox should do with the Burk\_chr2comp.zip file? Save the file to disk.

Click the **OK** button



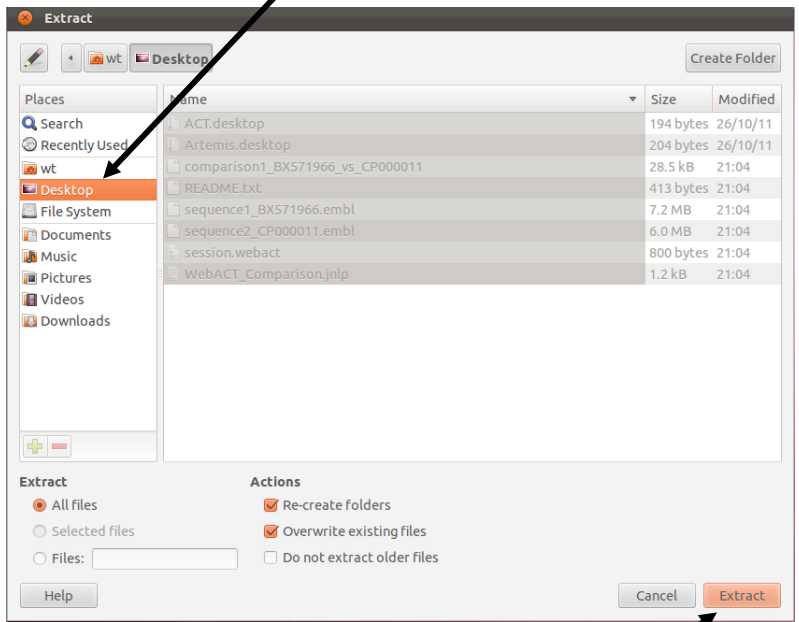
Burk\_chr2\_comp.zip should now be in the **Downloads** directory

To unzip the file, double click with the left mouse button on the file name



Click the **Extract** button

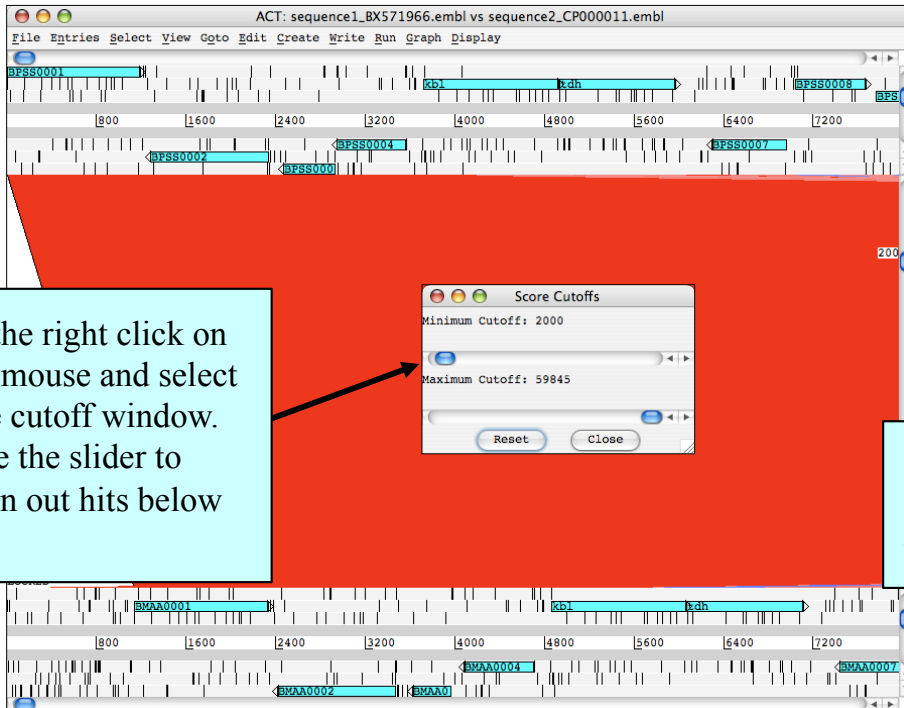
Select a location to extract the files to, such as **Desktop**



Click the **Extract** button

The files contained in the unzipped directory should include: comparison1\_BX571966\_vs\_CP000011, sequence1\_BX571966.embl and sequence2\_CP000011.embl. These are the ACT comparison file and the *B. pseudomallei* and *B. mallei* chromosome 2 EMBL annotation and sequence files respectively.

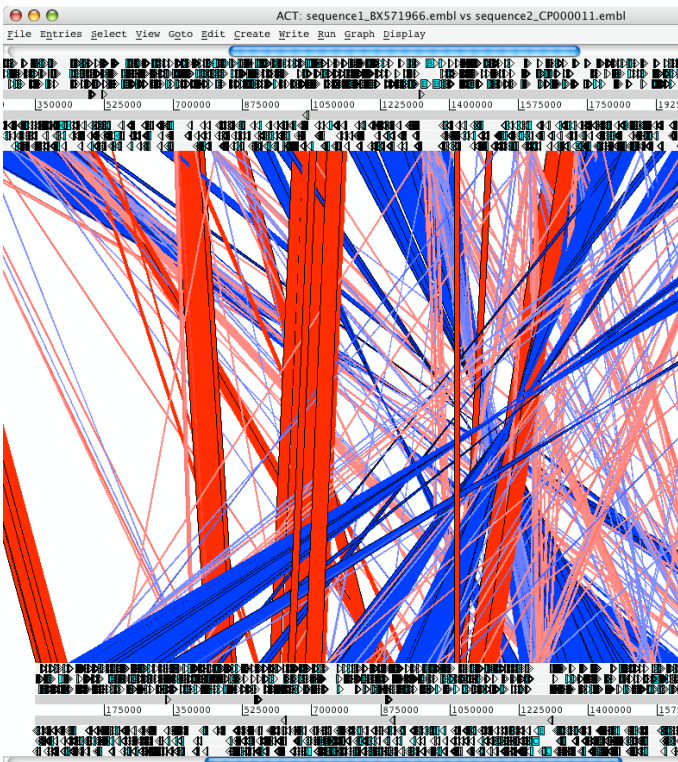
Open up ACT, and load up the comparison (comparison1\_BX571966\_vs\_CP000011) along with the two EMBL sequence and annotation files (sequence1\_BX571966.embl and sequence2\_CP000011.embl). If you get a warnings window asking if you want to read warning, click **No**.



Use the right click on your mouse and select score cutoff window. Move the slider to screen out hits below 2000

Move the slider to 200 to show only BLASTN matches greater than 200

Now remove the stop codons for both entries, and then zoom out you will see the overall conservation of the structure of the small chromosomes is poor.



If you were to look at the comparison for the large chromosomes you would see a similar picture. The lack of conservation is the result of intra-chromosomal rearrangements. What do you think caused this? Zoom into the regions on the edge of the rearranged matches and look at the annotation in the *B. mallei* chromosome.

What is the function of the CDSs consistently found in these regions. Are there matches in the *B. pseudomallei* chromosome?

Try selecting CDSs in *B. pseudomallei* that match these regions and look how many matches there are in *B. mallei*. Are these regions repeated throughout the chromosome?

If you have time, you may want to generate, and view in ACT, comparisons for your own sequences. If you do not have any loaded on your workshop computer, why not try and download some. Sequence in various formats can be cut and pasted, or up loaded onto the WebACT site. In addition, if you know the accession number of the sequence that you want to compare, you can use that. As the web site will have to run BLAST to generate your comparison file, you may want to limit the size of the sequence that you submit for this exercise to <100 kb. The the web site can handle larger sequences, but it will just take longer.

The image shows a screenshot of the WebACT website interface in a Mozilla Firefox browser window. The browser title is 'WebACT: Enter Query' and the address bar shows 'www.webact.org/WebACT/generate'. The page has a navigation bar with tabs: 'Pre-computed', 'Generate', 'Reload', and 'Instructions'. The 'Generate' tab is selected. Below the navigation bar is the WebACT logo and the text 'WebACT | Enter Query'. There is a 'Contact us' link in the top right. A dropdown menu asks 'How many sequences to you wish to compare?' with the number '2' selected. Below this is a checkbox for 'Send e-mail notification on job completion?' and an 'e-mail address:' input field. The main section is titled 'For each sequence below, please either paste a sequence, upload a sequence file or enter an EMBL or Refseq Accession number i.e. NTCAD19MR'. There are two sequence input sections, 'Sequence 1 -' and 'Sequence 2 -'. Each section has three options: 'Paste sequence (raw, EMBL or FASTA format)' (selected), 'Upload File (raw, EMBL or FASTA format)' (with a 'Browse...' button), and 'Enter an EMBL or Refseq Accession number'. At the bottom, there is a 'Blast Search Options [show]' link, a 'Submit' button, and a 'Clear' button. Several callout boxes with arrows point to these elements: 'Clicking on the 'Generate' tab will take you to this page' points to the 'Generate' tab; 'Number of sequences to compare' points to the dropdown menu; 'Cut and paste sequence' points to the text input area of Sequence 1; 'Upload file' points to the 'Browse...' button; 'Type accession number' points to the input field for Sequence 1; 'Click here for BLAST options, such as changing from the default BlastN to TblastX, and altering the BLAST cutoffs' points to the 'Blast Search Options [show]' link; and 'Once you added the relevant sequence information, submit your query. The comparison file or files are down loaded as shown in the example, and can them be loaded in to ACT.' points to the 'Submit' button.

Clicking on the 'Generate' tab will take you to this page

Number of sequences to compare

Cut and paste sequence

Upload file

Type accession number

Click here for BLAST options, such as changing from the default BlastN to TblastX, and altering the BLAST cutoffs

Once you added the relevant sequence information, submit your query. The comparison file or files are down loaded as shown in the example, and can them be loaded in to ACT.

## Appendix VII: Useful Web addresses

### Major Public Sequence Repositories

DNA Data Bank of Japan (DDBJ)	<a href="http://www.ddbj.nig.ac.jp">http://www.ddbj.nig.ac.jp</a>
EMBL Nucleotide Sequence Database	<a href="http://www.ebi.ac.uk/embl">http://www.ebi.ac.uk/embl</a>
Genomes at the EBI	<a href="http://www.ebi.ac.uk/genomes">http://www.ebi.ac.uk/genomes</a>
GenBank	<a href="http://www.ncbi.nih.gov/Genbank">http://www.ncbi.nih.gov/Genbank</a>

### Microbial Genome Databases Resources

Sanger Microbial Genomes	<a href="http://www.sanger.ac.uk/Projects/Pathogens">http://www.sanger.ac.uk/Projects/Pathogens</a>
GeneDB	<a href="http://www.genedb.org">http://www.genedb.org</a>
Institute Pasteur GenoList databases <i>Including: SubtiList, Colbri, TubercuList, Leproma, PyloriGene, MypuList, ListiList, CandidaDB.</i>	<a href="http://genolist.pasteur.fr">http://genolist.pasteur.fr</a>
Pseudomonas Genome Database	<a href="http://www.pseudomonas.com">http://www.pseudomonas.com</a>
Clusters of Orthologous Groups of proteins (COGs)	<a href="http://www.ncbi.nlm.nih.gov/COG">http://www.ncbi.nlm.nih.gov/COG</a>
ScoDB ( <i>S. coelicolor</i> database)	<a href="http://streptomyces.org.uk">http://streptomyces.org.uk</a>
GenProtEC	<a href="http://genprotec.mbl.edu">http://genprotec.mbl.edu</a>

### Protein Motif Databases

Prosite	<a href="http://www.expasy.ch/prosite/">http://www.expasy.ch/prosite/</a>
Pfam	<a href="http://pfam.sanger.ac.uk">http://pfam.sanger.ac.uk</a>
BLOCKS	<a href="http://blocks.fhcrc.org">http://blocks.fhcrc.org</a>
InterPro	<a href="http://www.ebi.ac.uk/interpro/">http://www.ebi.ac.uk/interpro/</a>
PRINTS	<a href="http://umber.sbs.man.ac.uk/dbbrowser/PRINTS/">http://umber.sbs.man.ac.uk/dbbrowser/PRINTS/</a>
SMART	<a href="http://smart.embl-heidelberg.de">http://smart.embl-heidelberg.de</a>

### Protein feature prediction tools

TMHMM Transmembrane helices prediction	<a href="http://www.cbs.dtu.dk/services/TMHMM-2.0/">http://www.cbs.dtu.dk/services/TMHMM-2.0/</a>
SignalP Prediction Server	<a href="http://www.cbs.dtu.dk/services/SignalP/">http://www.cbs.dtu.dk/services/SignalP/</a>
PSORT protein prediction	<a href="http://psort.ims.u-tokyo.ac.jp/form.html">http://psort.ims.u-tokyo.ac.jp/form.html</a>

### Metabolic Pathways and Cellular Regulation

EcoCyc	<a href="http://ecocyc.org/">http://ecocyc.org/</a>
ENZYME	<a href="http://www.expasy.ch/enzyme/">http://www.expasy.ch/enzyme/</a>
Kyoto Encyclopedia of Genes and Genomes (KEGG)	<a href="http://www.genome.ad.jp/kegg">http://www.genome.ad.jp/kegg</a>
MetaCyc	<a href="http://metacyc.org/">http://metacyc.org/</a>

### Miscellaneous sites

NCBI BLAST website	<a href="http://www.ncbi.nlm.nih.gov/BLAST/">http://www.ncbi.nlm.nih.gov/BLAST/</a>
EBI FASTA website	<a href="http://www.ebi.ac.uk/fasta33/index.html">http://www.ebi.ac.uk/fasta33/index.html</a>
The tmRNA website	<a href="http://www.indiana.edu/~tmrna/">http://www.indiana.edu/~tmrna/</a>
tRNAscan-SE Search Server	<a href="http://selab.janelia.org/tRNAscan-SE/">http://selab.janelia.org/tRNAscan-SE/</a>
Rfam	<a href="http://rfam.sanger.ac.uk/">http://rfam.sanger.ac.uk/</a>
Codon usage database	<a href="http://www.kazusa.or.jp/codon/">http://www.kazusa.or.jp/codon/</a>
GO Gene Ontology Consortium	<a href="http://www.geneontology.org/">http://www.geneontology.org/</a>
Artemis homepage	<a href="http://www.sanger.ac.uk/Software/Artemis/">http://www.sanger.ac.uk/Software/Artemis/</a>
ACT homepage	<a href="http://www.sanger.ac.uk/Software/ACT/">http://www.sanger.ac.uk/Software/ACT/</a>
WebACT	<a href="http://www.webact.org/WebACT/home">http://www.webact.org/WebACT/home</a>
Double ACT	<a href="http://www.hpa-bioinfotools.org.uk/pise/double_act.html">http://www.hpa-bioinfotools.org.uk/pise/double_act.html</a>
Glimmer	<a href="http://cbeb.umd.edu/software/glimmer/">http://cbeb.umd.edu/software/glimmer/</a>
EasyGene	<a href="http://www.cbs.dtu.dk/services/EasyGene/">http://www.cbs.dtu.dk/services/EasyGene/</a>
String	<a href="http://string.embl.de">http://string.embl.de</a>
EMBOSS	<a href="http://emboss.sourceforge.net/">http://emboss.sourceforge.net/</a>

## Appendix VIII: Prokaryotic Protein Classification Scheme used within the PSU

This scheme was adapted for in-house use from the Monica Riley's protein classification (<http://genprotec.mbl.edu/files/Multifun.html>).

More classes can be added depending on the microorganism that is being annotated (e.g secondary metabolites, sigma factors (ECF or non-ECF), etc).

- 0.0.0 Unknown function, no known homologs
- 0.0.1 Conserved in Escherichia coli
- 0.0.2 Conserved in organism other than Escherichia coli
- 1.0.0 Cell processes
  - 1.1.1 Chemotaxis and mobility
  - 1.2.1 Chromosome replication
  - 1.3.1 Chaperones
- 1.4.0 Protection responses
  - 1.4.1 Cell killing
  - 1.4.2 Detoxification
  - 1.4.3 Drug/analog sensitivity
  - 1.4.4 Radiation sensitivity
- 1.5.0 Transport/binding proteins
  - 1.5.1 Amino acids and amines
  - 1.5.2 Cations
  - 1.5.3 Carbohydrates, organic acids and alcohols
  - 1.5.4 Anions
  - 1.5.5 Other
- 1.6.0 Adaptation
  - 1.6.1 Adaptations, atypical conditions
  - 1.6.2 Osmotic adaptation
  - 1.6.3 Fe storage
- 1.7.1 Cell division
- 2.0.0 Macromolecule metabolism
- 2.1.0 Macromolecule degradation
  - 2.1.1 Degradation of DNA
  - 2.1.2 Degradation of RNA
  - 2.1.3 Degradation of polysaccharides
  - 2.1.4 Degradation of proteins, peptides, glycoproteins
- 2.2.0 Macromolecule synthesis, modification
  - 2.2.01 Amino acyl tRNA synthesis; tRNA modification
  - 2.2.02 Basic proteins - synthesis, modification
  - 2.2.03 DNA - replication, repair, restriction./modification
  - 2.2.04 Glycoprotein
  - 2.2.05 Lipopolysaccharide
  - 2.2.06 Lipoprotein
  - 2.2.07 Phospholipids
  - 2.2.08 Polysaccharides - (cytoplasmic)
  - 2.2.09 Protein modification
  - 2.2.10 Proteins - translation and modification
  - 2.2.11 RNA synthesis, modif., DNA transcrip.
  - 2.2.12 tRNA
- 3.0.0 Metabolism of small molecules
- 3.1.0 Amino acid biosynthesis
  - 3.1.01 Alanine
  - 3.1.02 Arginine
  - 3.1.03 Asparagine
  - 3.1.04 Aspartate
  - 3.1.05 Chorismate
  - 3.1.06 Cysteine
  - 3.1.07 Glutamate
  - 3.1.08 Glutamine
  - 3.1.09 Glycine
  - 3.1.10 Histidine
  - 3.1.11 Isoleucine
  - 3.1.12 Leucine
  - 3.1.13 Lysine
  - 3.1.14 Methionine
  - 3.1.15 Phenylalanine
  - 3.1.16 Proline
  - 3.1.17 Serine
  - 3.1.18 Threonine
  - 3.1.19 Tryptophan
  - 3.1.20 Tyrosine
  - 3.1.21 Valine

**Appendix VIII (cont):**

- 3.2.0 Biosynthesis of cofactors, carriers
  - 3.2.01 Acyl carrier protein (ACP)
  - 3.2.02 Biotin
  - 3.2.03 Cobalamin
  - 3.2.04 Enterochelin
  - 3.2.05 Folic acid
  - 3.2.06 Heme, porphyrin
  - 3.2.07 Lipoate
  - 3.2.08 Menaquinone, ubiquinone
  - 3.2.09 Molybdopterin
  - 3.2.10 Pantothenate
  - 3.2.11 Pyridine nucleotide
  - 3.2.12 Pyridoxine
  - 3.2.13 Riboflavin
  - 3.2.14 Thiamin
  - 3.2.15 Thioredoxin, glutaredoxin, glutathione
  - 3.2.16 biotin carboxyl carrier protein (BCCP)
- 3.3.0 Central intermediary metabolism
  - 3.3.01 2'-Deoxyribonucleotide metabolism
  - 3.3.02 Amino sugars
  - 3.3.03 Entner-Doudoroff
  - 3.3.04 Gluconeogenesis
  - 3.3.05 Glyoxylate bypass
  - 3.3.06 Incorporation metal ions
  - 3.3.07 Misc. glucose metabolism
  - 3.3.08 Misc. glycerol metabolism
  - 3.3.09 Non-oxidative branch, pentose pathway
  - 3.3.10 Nucleotide hydrolysis
  - 3.3.11 Nucleotide interconversions
  - 3.3.12 Oligosaccharides
  - 3.3.13 Phosphorus compounds
  - 3.3.14 Polyamine biosynthesis
  - 3.3.15 Pool, multipurpose conversions of intermed. metab.
  - 3.3.16 S-adenosyl methionine
  - 3.3.17 Salvage of nucleosides and nucleotides
  - 3.3.18 Sugar-nucleotide biosynthesis, conversions
  - 3.3.19 Sulfur metabolism
  - 3.3.20 Amino acids
  - 3.3.21 other
- 3.4.0 Degradation of small molecules
  - 3.4.1 Amines
  - 3.4.2 Amino acids
  - 3.4.3 Carbon compounds
  - 3.4.4 Fatty acids
  - 3.4.5 Other
  - 3.4.0 ATP-proton motive force
- 3.5.0 Energy metabolism, carbon
  - 3.5.1 Aerobic respiration
  - 3.5.2 Anaerobic respiration
  - 3.5.3 Electron transport
  - 3.5.4 Fermentation
  - 3.5.5 Glycolysis
  - 3.5.6 Oxidative branch, pentose pathway
  - 3.5.7 Pyruvate dehydrogenase
  - 3.5.8 TCA cycle
- 3.6.0 Fatty acid biosynthesis
  - 3.6.1 Fatty acid and phosphatidic acid biosynthesis
- 3.7.0 Nucleotide biosynthesis
  - 3.7.1 Purine ribonucleotide biosynthesis
  - 3.7.2 Pyrimidine ribonucleotide biosynthesis
- 4.0.0 Cell envelop
  - 4.1.0 Periplasmic/exported/lipoproteins
  - 4.1.1 Inner membrane
  - 4.1.2 Murein sacculus, peptidoglycan
  - 4.1.3 Outer membrane constituents
  - 4.1.4 Surface polysaccharides & antigens
  - 4.1.5 Surface structures
- 4.2.0 Ribosome constituents
  - 4.2.1 Ribosomal and stable RNAs
  - 4.2.2 Ribosomal proteins - synthesis, modification
  - 4.2.3 Ribosomes - maturation and modification
- 5.0.0 Extrachromosomal
  - 5.1.0 Laterally acquired elements
    - 5.1.1 Colicin-related functions
    - 5.1.2 Phage-related functions and prophages
    - 5.1.3 Plasmid-related functions
    - 5.1.4 Transposon-related functions
    - 5.1.5 Pathogenicity island-related function
- 6.0.0 Global functions
  - 6.1.1 Global regulatory functions
- 7.0.0 Not classified (included putative assignments)



## Appendix IX: List of colour codes

- 0** (white) - Pathogenicity/Adaptation/Chaperones
- 1** (dark grey) - energy metabolism (glycolysis, electron transport etc.)
- 2** (red) - Information transfer (transcription/translation + DNA/RNA modification)
- 3** (dark green) - Surface (IM, OM, secreted, surface structures)
- 4** (dark blue) - Stable RNA
- 5** (Sky blue) - Degradation of large molecules
- 6** (dark pink) - Degradation of small molecules
- 7** (yellow) - Central/intermediary/miscellaneous metabolism
- 8** (light green) - Unknown
- 9** (light blue) - Regulators
- 10** (orange) - Conserved hypo
- 11** (brown) - Pseudogenes and partial genes (remnants)
- 12** (light pink) - Phage/IS elements
- 13** (light grey) - Some misc. information e.g. Prosite, but no function

## Appendix X: List of degenerate nucleotide value/IUB Base Codes.

**R = A or G**

**S = G or C**

**B = C, G or T**

**Y = C or T**

**W = A or T**

**D = A, G or T**

**K = G or T**

**N = A, C, G or T**

**H = A, C or T**

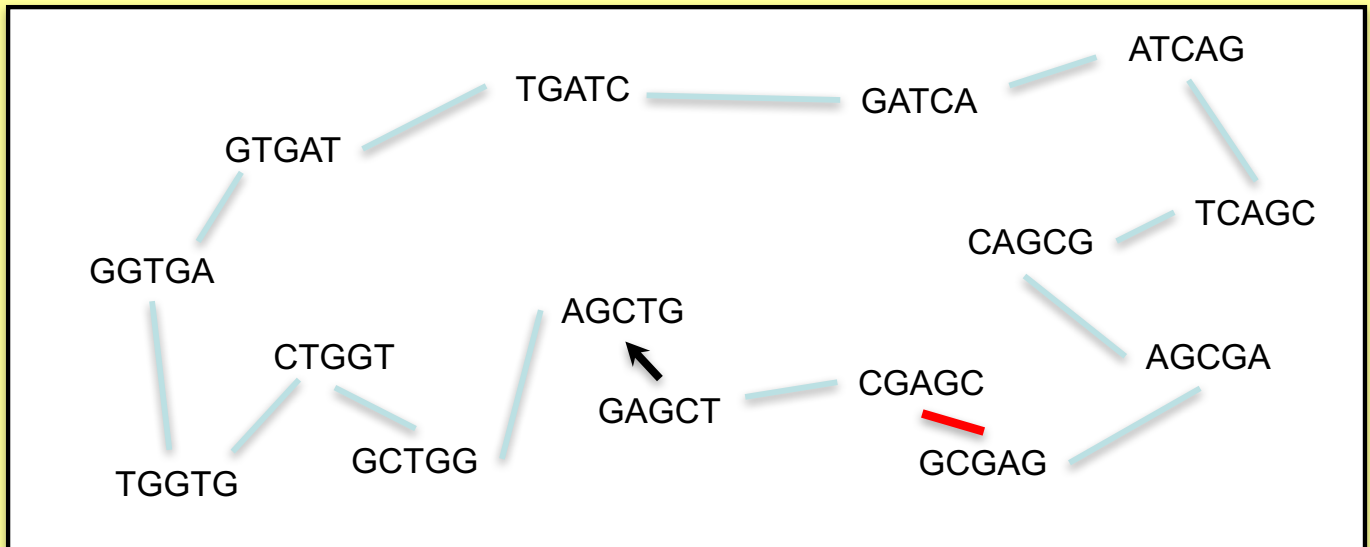
**M = A or C**

**V = A, C or G**

## Appendix X - Assembly

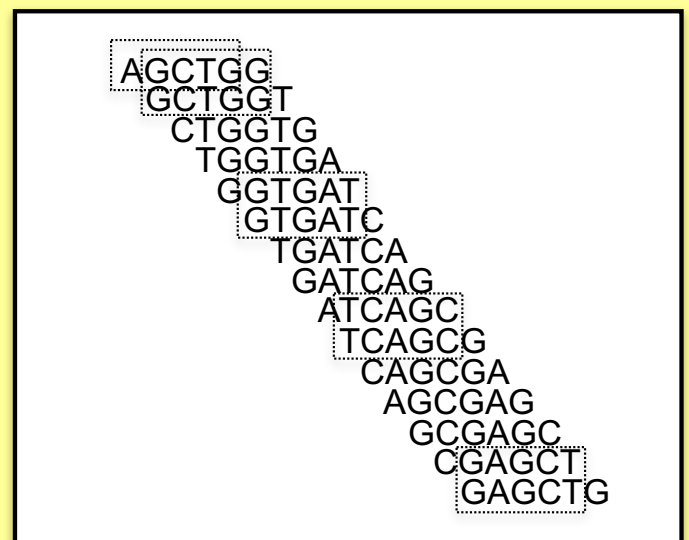
Here we present the solution for the de Bruijn graph exercise, as well as the code for the PERL scripts we mentioned in the assembly module.

The exercise is on page 2 of the assembly module. The first step of the solution would be to generate all the k-mers from the reads. For example the k-mers of length 5 for the read GCGAGC are GCGAG and CGAGC. Those two k-mers will be nodes in the de Bruijn graph, and moreover, will be connected (bold red edge). Doing this for all the reads, generates the following graph:



It is not always easy and might be confusing which node to connect. Remember, the concept of k-mers and the de Bruijn graph are needed to be able to process the large amount of short reads generated by the sequencing machines.

The graph can also be represented as a multiple alignment, as shown on the right hand side. All reads are aligned against each other. The dotted boxes are examples of k-mers.



Now just follow the path through the graph. Starting at the arrow, the first k-mer is GAGCT, so this would be the start of our contig. The graph indicates the next k-mer AGCTG. So we add a G to the contig. The next k-mer is GCTGG. The new letter is another G. Doing this for the whole graph, we get: **GAGCTGGT**GATCAGC. As you see, the graph is circular. So depending where you start, you get a different contig! If you do a six frame translation, you might see which is a good starting point for the contig.

## PERL: Find read pairs that map too far apart

For some applications it would be useful to know whether read pairs map too far apart or whether they don't map pointing to each other. This could be an indication of mis-assemblies, but also duplications or rearrangements, which are looking for when comparing sequences of different strains.

To find read pairs (RPs) that map too far apart we just need columns 2 and 9 from the BAM file (mapping flag and insert size), and a PERL one-liner. We successively make the query more and more complex, until we find the mis-assembly. Please keep in mind that this is advanced programming! It should give you an idea how useful programming could be.

Assuming your BAM file is called `IT_onDenovo.bam` and you want to list RPs that map more than 2000bp apart:

```
$ samtools view IT_onDenovo.bam | perl -nle 'my ($read,$flag,$ref,$pos,$mappingQual,$cigar,$mateRef,$matePos,$insertSize,$seq,$seqQual,$other)=split(/\t/); if($insertSize>2000){print}' | head
```

Here is also a shorter version, using an array (not as readable):

```
$ samtools view IT_onDenovo.bam | perl -nle 'my @ar=split(/\t/); if($ar[8]>2000){print}' | head
```

There is a lot of output. Many read pairs map all over the place. We would like to bin those into chunks of 1kb, and then list of the most abundant:

```
$ samtools view IT_onDenovo.bam | perl -nle 'my ($read,$flag,$ref,$pos,$mappingQual,$cigar,$mateRef,$matePos,$insertSize,$seq,$seqQual,$other)=split(/\t/); if($insertSize>2000){print int($pos/1000)."\t".int($matePos/1000)}' | sort | uniq -c | sort -rn | head
```

This does look more complex! In the output the first column is the number of RPs that connect the first bin (2<sup>nd</sup> column) with the second bin (3<sup>rd</sup> column). For example `418 1360 1373` means that 481 RPs connect the region 1360000-1361000 of genome with the regions 1373000-13731000 of the genome. The list shows us that in the subtelomeric regions many RP map far apart!

The following command ignores the subtelomeric ends, by excluding 75kb at each end.

```
$ samtools view IT_onDenovo.bam | perl -nle 'my ($read,$flag,$ref,$pos,$mappingQual,$cigar,$mateRef,$matePos,$insertSize,$seq,$seqQual,$other)=split(/\t/); if($insertSize>10000 && $pos>75000 && $matePos < 1300000){print int($pos/1000)."\t".int($matePos/1000)}' | sort | uniq -c | sort -nr
```

The third line `21 81 323` shows us our mis-assembly. What are the other entries?

We are fully aware that this is a quite complex piece of code, and just used as a one liner. It uses the LINUX commands `sort` and `uniq`. But keep in mind that this command can find you all mate pairs mapping too far apart for any bam file (if you adjust the `insertSize` parameter for your data)!

## Getting non mapping reads and their mates

Here is an example of how to get the mates of non mapping reads. It is a good example of PERL one-liners.

First we are going to get reads that don't map with PERL. The original command is:

```
$ samtools view -f 0X4 IT.Chr5.bam | head
```

In PERL this would be:

```
$ samtools view IT.Chr5.bam | perl -nle 'my ($read,$flag)=split(/\t/); if ($flag & 0x4) {print }' | head
```

Now we need to get the reads where the mate is not mapped. Looking at the samtools manual:

Bit	Description
0x1	template having multiple segments in sequencing
0x2	each segment properly aligned according to the aligner
0x4	segment unmapped
0x8	next segment in the template unmapped
0x10	SEQ being reverse complemented
0x20	SEQ of the next segment in the template being reversed
0x40	the first segment in the template
0x80	the last segment in the template
0x100	secondary alignment
0x200	not passing quality controls
0x400	PCR or optical duplicate

The 0x8 tells if the mate pair is not mapped. So if the read is not mapping (0x4) or the mate is not mapping (0x8) then print the sam line into a file:

```
$ samtools view IT.Chr5.bam | perl -nle 'my ($read,$flag)=split(/\t/); if ($flag & 0x4 or $flag & 0x8) {print }' | sort > NonmappingReadsPlusmate.sam
```

This file can now be used in VELVET for *de novo* assembly as explained in the Assembly module.

We hope that this illustrates the power of PERL one-liners!

## Appendix XI Splice site information

Gene	No.	Exon	Intron	Exon	Size (bp)
41-3	1	GAA	<b>GTACACA</b> . . CCTTCTTTTTCCATATTTAG	CAA	152
	2	AAT	<b>GTTAAAA</b> . . . TTTTTTTTTTTAAACTTAG	CCG	208
	3	GAG	<b>GTAAGAA</b> . . . ATTCATTATATATTTATAG	GGA	86
	4	TCG	<b>GTATGGA</b> . . . TTTTGAAATACTTCCTCAG	TTA	152
	5	ACT	<b>GTAATAT</b> . . TTTTTTTTTTTATTTCCTAG	ATG	112
	6	CAG	<b>GTAATA</b> . . ATAATGACATTTTGATACAG	ATT	120
	7	AAT	<b>GTACATT</b> . . TTATTTTATTATTATTTATAG	AAA	81
	8	TAG	<b>GTATTTG</b> . . ATATTTTTTACTTATGATAG	TTA	96
RhopH3	1	AGG	<b>GTAATAT</b> . . TTTATTTTATTTTTTTTTTA	TTT	150
	2	GGA	<b>GTAAGAG</b> . . TTTTATTATTATTATTGTAG	TCC	442
	3	GGA	<b>GTAAGAG</b> . . TTTTATTATTATTATTGTAG	TCC	199
	4	CAG	<b>GTAYGCT</b> . . TTTAATTTTTTTTCCCTCA	TCA	160
	5	AAA	<b>GTAAGAA</b> . . TATTTTTTTTACAATTTTAG	TTC	206
	6	AAG	<b>GTAAAAG</b> . . TTTTTTTTTTTTGTTCAG	TTT	142
RNA pol III	1	CAG	<b>GTACATA</b> . . TTTTTTTTTTTTTTTTTTAG	GTG	158
	2	CAA	<b>GTAATTA</b> . . TATATTTTATTTTTTCTTAG	GTT	113
	3	TAC	<b>GTTAGTT</b> . . TTTTTTTTTTTTTTTTTTAG	TGG	169
	4	ATT	<b>GTAAGTT</b> . . TATTTTTTTTTTTTTTTTTTAG	TGA	112
SERA	1	TGT	<b>GTAAGAA</b> . . TTGTCATTATTTTTTTTTTAG	GTG	158
	2	AAA	<b>GTATAAA</b> . . TTTATTTATTTTTTTTTTAG	ATA	175
	3	CAG	<b>GTAATA</b> . . TTTTAATTTTTTTGTTTTAG	AAA	129
SERP H	1	CTG	<b>GTTTGTC</b> . . CATATATTTCTTTATTTTAG	ATA	345
	2	AGA	<b>GTAAAAA</b> . . TTTCTTATATTTTCTTTTAG	GTG	92
	3	CTG	<b>GTTTGTC</b> . . CATATATTTCTTTATTTTAG	ATA	116
Ag15	1	ATG	<b>GTAAGAG</b> . . TATTTTTGATACCTTTATAG	AGT	214
	2	AAA	<b>GTAATTA</b> . . CAATCATATTAACACAAAAG	ATG	280
PfGPx	1	GAG	<b>GTATACA</b> . . TTATTATCCCTTGCTTTAG	ATC	208
	2	TCG	<b>GTTAGTA</b> . . TATTTATCATTTTTTTCCAG	ATG	168
Calmodulin	1	GAA	<b>GTAATC</b> . . TTTTTTATTTTCTCATTAG	CTA	480
PfPK1	1	TAG	<b>GTGTGTT</b> . . TCATTACATTTTACCTTAG	GAT	101
MESA	1	TTA	<b>GTAAGTT</b> . . CGTAATATATTTTTTTTTTAG	GAT	122
Aldolase	1	ATG	<b>GTAAGAA</b> . . TATTTTTATATTTTTTTTTTAG	GCT	452
KAHRP	1	AAC	<b>GTAAGTT</b> . . TTATTTTTTTTTTTCATATAG	TGC	430
GBPH2	1	TTG	<b>GTATGCC</b> . . TTTGTATTATTTAATTTTAG	AAT	157
GBP	1	TTG	<b>GTATG</b> . . . TGTGTATTGTTTATTTTTTAG	AAT	179
FIRA	1	TGT	<b>GTAAGGA</b> . . TTTTATATTTTTTCTTTAG	CGA	175
GARP	1	AAG	<b>GTAACAA</b> . . TATATGTATTTTTTTTTTAG	TGC	214

↑  
Donor motif

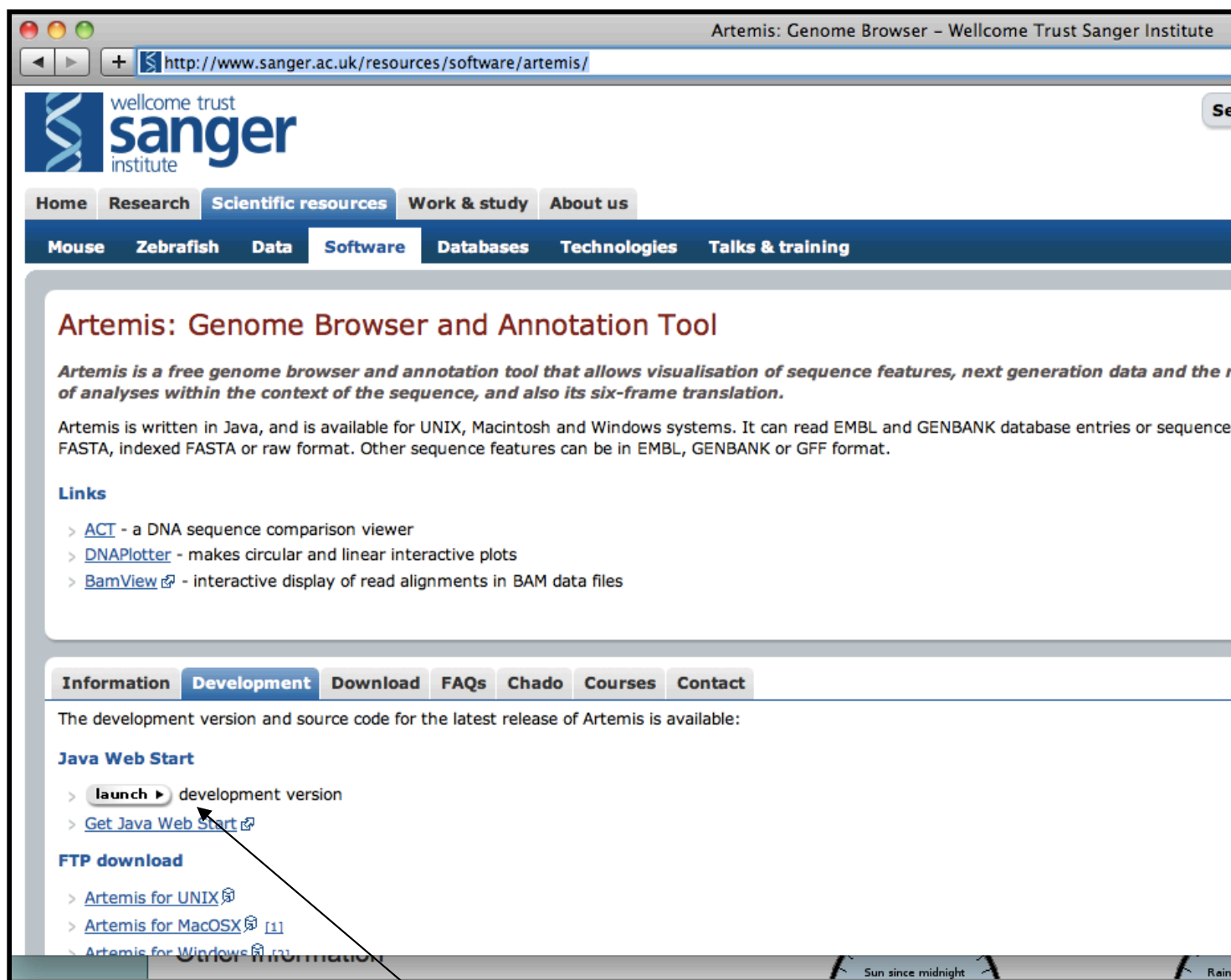
↑  
Acceptor motif

The splice acceptor and donor sequences for several *P. falciparum* genes: adapted from Coppel and Black(1998). In "Malaria:Parasite Biology, Pathogenesis and Protection", I.W. Sherman (ed.); ASM Press; Washington DC; pp185-202

## Appendix XII Running Artemis from the Web

To work this Artemis you don't necessary have to work with it from the VM. It can be run from the web:

<http://www.sanger.ac.uk/resources/software/artemis/>



The screenshot shows a web browser window with the URL <http://www.sanger.ac.uk/resources/software/artemis/>. The page header includes the Wellcome Trust Sanger Institute logo and navigation tabs: Home, Research, Scientific resources, Work & study, About us. Below this is a secondary navigation bar with tabs: Mouse, Zebrafish, Data, Software, Databases, Technologies, Talks & training. The main content area features the title "Artemis: Genome Browser and Annotation Tool" and a description: "Artemis is a free genome browser and annotation tool that allows visualisation of sequence features, next generation data and the of analyses within the context of the sequence, and also its six-frame translation." It also states: "Artemis is written in Java, and is available for UNIX, Macintosh and Windows systems. It can read EMBL and GENBANK database entries or sequence FASTA, indexed FASTA or raw format. Other sequence features can be in EMBL, GENBANK or GFF format." A "Links" section lists: > [ACT](#) - a DNA sequence comparison viewer, > [DNAPlotter](#) - makes circular and linear interactive plots, > [BamView](#) - interactive display of read alignments in BAM data files. A secondary navigation bar includes: Information, Development, Download, FAQs, Chado, Courses, Contact. The "Development" tab is active, showing the text: "The development version and source code for the latest release of Artemis is available:". Under "Java Web Start", there is a "launch" button with a right-pointing arrow next to the text "development version". Below this are links: > [Get Java Web Start](#). Under "FTP download", there are links: > [Artemis for UNIX](#), > [Artemis for MacOSX](#), and > [Artemis for Windows](#). A black arrow points from the "launch" button to a text box below the screenshot.

Click on 'launch', accept and wait a bit...  
Next you can load sequence, bam files etc.

# Appendix XIII Exploring the Sequence Read Archive

## Exercise

You can access the sequence read archive through the following sites:

<http://www.ncbi.nlm.nih.gov/sra>

<http://www.ebi.ac.uk/ena>

It's possible to download transcriptome data and other next generation sequencing data as follows. These instructions are given in the form of an exercise to help make it more interesting:

Go to the following website: <http://www.ebi.ac.uk/ena> and type in the search box:

RNA-seq, Plasmodium falciparum

Now follow the step-by-step instructions to download this data.

The screenshot shows the ENA website interface. The search bar contains 'RNA-seq Plasmodium falciparum'. The search results are displayed in a grid format. The 'Nucleotide Sequences' category shows 1 result. A callout box points to this category with the text 'Click on 'Nucleotide Sequences''. Below the search results, the 'Abstract' section is visible, showing navigation links for 'SRA Sample', 'SRA Submission', 'SRA Run', and 'SRA Experiment'. A callout box points to the 'SRA Run' link with the text 'Click on the link 'SRA Run''.

**Search for RNA-seq Plasmodium falciparum in All the EBI**

Expand all Collapse all

Genomes	0	Molecular Interactions	0
Nucleotide Sequences	1	Reactions & Pathways	0
Protein Sequences	0	Protein Families	0
Macromolecular Structures	0	Enzymes	0
Small molecules	0	Literature	0
Gene Expression	0	Ontologies	0
		EBI Web Site	0

**Abstract**

Navigation

↑ SRA Sample:	<a href="#">ERS000415-ERS000431</a>
↑ SRA Submission:	<a href="#">ERA000119</a> Sequence Read Archive submission submitted by The Wellcome Trust Sanger Institute
↓ SRA Run:	<a href="#">ERR006177-ERR006193</a>
↓ SRA Experiment:	<a href="#">ERX001045-ERX001061</a>

Attributes Top

ENA-SPOT-COUNT	112215691
ENA-BASE-COUNT	9299091736





A list of all the RNA experiments will come up. Click on the red arrow to expand the window.



**SRA Experiment Record : ERX001045** : Illumina Genome Analyzer II sequencing of Plasmodium falciparum 3D7 16 hr cDNA-50

View: [XML](#) Download: [XML](#)

Study : ERP000069 : Plasmodium falciparum RNA-Seq in Blood Stage [More details](#)

Sample : ERS000416 : [More details](#)

**Taxonomic classification**

<b>Organism</b>	Plasmodium falciparum 3D7
<b>Taxonomic identifier</b>	36329

**Library**

Name : 16\_hr\_3D7\_cDNA-50\_1  
 Source : NON GENOMIC  
 Selection : cDNA  
 Paired : Orientation :  
 Nominal length :  
 Nominal sdev : 0.0

**Spot descriptor**

Read 0: Class : Application Read  
 Type : Forward  
 Base coordinates : 1

Read 1: Class : Application Read  
 Type : Reverse  
 Base coordinates : 55

**Platform** : ILLUMINA : Illumina Genome Analyzer II

**Processing** : Base caller : Solexa primary analysis  
 Quality scorer : Solexa primary analysis

**Runs** : [ERR006186](#)

**Submission** : ERA000119

Click on 'Runs'



**SRA Run Record : ERR006186** : Sequence Read Archive run by The Wellcome Trust Sanger Institute

View: [XML](#) Download: [XML](#)

Study : ERP000069 : Plasmodium falciparum RNA-Seq in Blood Stage [More details](#)

Sample : ERS000416 : [More details](#)

**Taxonomic classification**

<b>Organism</b>	Plasmodium falciparum 3D7
<b>Taxonomic identifier</b>	36329

**Submitter** : The Wellcome Trust Sanger Institute

**Experiment** : ERX001045

**Submission** : ERA000119

\*For Aspera download, please [download and install Aspera Connect](#)

**Submitted files**

ftp  
[Download](#)

**Generated files**

ftp  
[ERR006186\\_1.fastq.gz](#)  
[ERR006186\\_2.fastq.gz](#)

Now you can download the RNA-Seq data.



## Appendix XIV

Here is compilations of the programs that we find useful.

### Artemis & Act

<http://www.sanger.ac.uk/science/tools/artemis>

<http://www.sanger.ac.uk/science/tools/artemis-comparison-tool-act>

### Mappers

SMALT: <http://www.sanger.ac.uk/science/tools/smalt-0>

BWA: <http://sourceforge.net/projects/bio-bwa/files/>

BOWTIE: <http://bowtie-bio.sourceforge.net/index.shtml>

TopHat: <https://ccb.jhu.edu/software/tophat/index.shtml>

### SAMTOOLS & BCFtools

<http://samtools.sourceforge.net/>

<https://samtools.github.io/bcftools/bcftools.html>

### Assemblers

Velvet: <http://www.ebi.ac.uk/~zerbino/velvet/>

ABYSS: <http://www.bcgsc.ca/platform/bioinfo/software/abyss>

SOAPdenovo: <http://soap.genomics.org.cn/soapdenovo.html>

### Tools for automatic finishing / Annotation transfer

ABACAS: <http://sourceforge.net/projects/abacas/files/>

IMAGE: <http://sourceforge.net/projects/image2/>

iCORN: <http://sourceforge.net/projects/icorn/files/>

PAGIT: <http://www.sanger.ac.uk/science/tools/pagit>