GRAFT (Genomic Rearrangement Assembly For Tumours)

Introduction

These readme notes describe usage of the GRAFT software, written in Matlab with the statistics toolbox. This tool is designed to help examine the timing of mutations in cancer.

The technique requires the following three inputs:

- Allelic copy number segmentation
- Aberrantly mapping paired end read data for these segments
- Point mutation counts in each segment for each possible multiplicity (i.e. zygosity)

The following outputs are then generated:

- A list of possible orders of rearrangements and their type (terminal deletion, interstitial deletion, chromosomal loss, chromosomal gain, tandem duplication, inversion, inverted duplication, breakage-fusion-bridge, translocation)
- A digital karyotype for each order
- The predicted times of when the rearrangements occurred (for events that increase copy number)

More details can be found in:

'Estimation of Rearrangement Phylogeny in Cancer', Greenman et al., Submitted.

GRAFT contains tools to try this. The file GRAFT.zip should contain the following:

- SimRearr.m
- EstRearr.m
- DEL_ID.mat
- Utils_Dir_Rand.m
- Utils_Graph_Comp.m
- Utils_Rearr_Classfy.m
- Utils_Time_Solver.m
- Utils_Dir_LogPdf.m

The main scripts that generate and analyse data are:

SimRearr - This function creates randomly rearranged chromosomes, reporting the rearrangements that have taken place along with their timings.

EstRearr - This function takes a rearranged genome and produces possible sequences of rearrangements that explain the data, along with resultant karyotypes.

The 'DEL_ID.mat' files contain some test data.

The Utils*.m files are just utiliy files.

STEP 1: Data Creation

There are three choices; A) load some test data, B) randomly generate some data or C) construct some real data.

A) Load Test Data

Type the following at the prompt:

```
>>load('DEL_ID.mat')
>>who
```

This produces the following:

```
Your variables are:

G  acn  g  mn_mut_rate  pmd  rearr_chr  rearr_names  rearr_types  segment_lens  times
```

***G{o1,o2,c1,c2}(b1,b2)**	Indexes connections between breakpoints *b1* and *b2* of chromosomes *c1* and *c2* with orientations *o1* and *o2*
***acn{chr_no}**	Allelic copy number for chromosomes *chr_no*
mn_mut_rate	Mean mutation rate per megabase per genomic copy per year
***pmd{chr_no,seg_no}(mult)**	Point mutation counts of multiplicity *mult* for chromosome *chr_no* and segment *seg_no*
rearr_chr{c}(s,1:4)	Rearranged chromosome no *c*, segment *s*, the four entries are chromosome no., parental allele, start position, end position
rearr_names{no}	The name of rearrangement number *no* is given
rearr_types{no}	A numerical classification for the rearrangement *rearr_name{no}*
***segment_lens**	These are the lengths of segments (in kilobases)
times	This contains the times (in years) between rearrangement events

This data is a simulation of the data generated from a **Deletion** and **Tandem Duplication**, in that order. We describe this example in more detail:

Note that only the variables * are generated by experiments and are all that is required to apply the techniques of GRAFT. The other variables can be used to test predicted times.

Example 1

Consider the following simple pedagogic example; there are two copies of a chromosome, one from each parent. One of the chromosomes undergoes an interstitial deletion. This chromosome then undergoes an inverted duplication that bridges the original deletion. This will result in an allelic copy number profile:

Segment	1	2	3	4	5
Major Copy Number	1	2	1	2	1
Minor Copy Number	1	1	0	1	1

To see this data type the following:

```
>>acn{1}

ans =

   1   1   0   1   1

   1   2   1   2   1
```

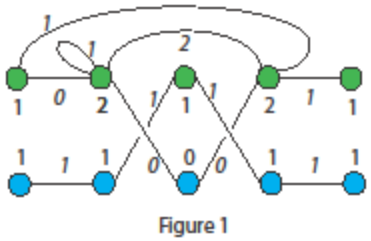
Figure 1

We also have the lengths, in megabases encoded as:

```
>> segment_lens{1}

ans =

   22.9040   21.7730   50.0005    3.2287    2.0978
```

We now have five segmented regions and four breakpoints. Aberrantly mapping reads from new sequencing data will indicate that the 3' end of segment 2 is attached to the 5' end of segment 4, implicating the second and third breakpoints. This is a result of the deletion. We also find that the 5' end of segment 2 is attached to the 3' end of segment 4, implicating the first and fourth breakpoints. These connections are captured with the non zero entries of the G variable:

```
>> G{1,1}
ans =
   0   0   0   1   0
   0   0   0   0   0
   0   0   0   0   0
   1   0   0   0   0
   0   0   0   0   0
```

```
>> G{1,2}
ans =
   0   0   0   0   0
   0   0   0   2   0
   0   0   0   0   0
   0   0   0   0   0
   0   0   0   0   0
```

```
>> G{2,1}
ans =
   0   0   0   0   0
   0   0   0   0   0
   0   0   0   0   0
   0   2   0   0   0
   0   0   0   0   0
```

```
>> G{2,2}
ans =
   0   0   0   0   0
   0   1   0   0   0
   0   0   0   0   0
   0   0   0   0   0
   0   0   0   0   0
```

The resultant allelic graph for this example is displayed above.

These data were generated by simulation (see below). This simulation assumes that a background point mutation process is present. The resultant counts of mutations and the number of genomic copies are encoded in the pmd variable;

```
>> pmd{1:5}

ans =

   16

ans =

   24    4

ans =

   31

ans =

    3     1

ans =

    1
```

So there is 1 mutation in segment 4 that have two copies in the genome, for example.

It is these four variables that need encoding from any real data. For simulated data we also have the remaining descriptive variables:

The times between the rearrangements, in years:

```
>>times

times =

   2.4247    2.3426    6.3085
```

The mutation rate per year per genomic copy:

```
>> mn_mut_rate

mn_mut_rate =

   0.5000
```

The types of rearrangements are encoded as:

```
>> rearr_names,rearr_types
```

```
rearr_names =

   'deletion'    'inverted duplication'

rearr_types =

    7     2
```

The possible types (and numbers) are Tandem Duplication (1), inverted duplication (2), breakage-fusion-bridge (3), insertion (4), chromosomal gain (5), chromosomal loss (6), deletion (7), translocation (8) and inversion (9).

B) Randomly Generate Data

Similar data to **A)** can be generated with the following function:

```
>>[mn_mut_rate,times,rearr_types,rearr_names,rearr_chr,segment_lens,acn,g,G,pmd] = SimRearr
```

Note that the data generation is random and some modification to this script will be required to generate specific rearrangement combinations.

C) Real Data

If real data is available, construct the four variables indexed with * in **A)**.

STEP 2: Order and Time the Rearrangements

To estimate the possible rearrangements, run the following:

```
>>out = EstRearr(acn,G,segment_lens,pmd)
```

This produces the following output:

```
NoRearrOrders: {{1x2 cell}}
Topology: {{1x1 cell}}
No_ChromComp: {[2]}
ChromComp: {{1x2 cell}}
NoRearrOrderSets: {{1x2 cell}}
RearrChr: {{1x2 cell}}
SegmentalEvolution: {{1x2 cell}}
RearrOrders: {{1x2 cell}}
NoTop: 1
Times: {{1x1 cell}}
Times_ci: {{1x1 cell}}
MutRate: {{1x1 cell}}
MutRate_ci: {{1x1 cell}}
```

These are to be interpreted as follows:

out.NoTop [The number of allelic topologies]

```
>> out.NoTop

ans =

   1
```

out.Topology{top_no}{c}(:,b) [α and β values for breakpoint b and chromosome c]

```
>>out.Topology{1}{1}

ans =

   1   -1   -1    1

   1    1   -1    1
```

out.No_ChromComp{top_no} [the number of different components of the allelic graph]

```
>> out.No_ChromComp{1}

ans =

   2
```

[Note the graph in Figure 1 has two components]

out.NoRearrOrders{top_no}{ChromComp_no}

[the number of possible orders of rearrangements]

```
>> out.NoRearrOrders{1}{2}

ans =

   1
```

out.RearrOrders{top_no}{ChromComp_no}{order_no}

[A viable list of rearrangement orders for the allelic graph component]

```
>> out.RearrOrders{1}{2}{1}

ans =

   2    1

   2    4
```

The first row is used to label each rearrangement, the second classifies the rearrangements from chromosomal loss (-1), chromosomal duplication (0), breakage-fusion-bridge (1), Tandem duplication/Deletion (2), inversion translocation (3), inverted duplication (4), insertion (5), arm loss (6).

out.Times{top_no}{order_no}{chrom_comp_no}

[The estimated times for rearrangements increasing copy number]

```
>> out.Times{1}{1}{2}

ans =

   0.4031    1.0000
```

If we compare this to the cumulative sum of the proportionate times between rearrangements,

```
>> cumsum(times/sum(times))

ans =

   0.2189    0.4304    1.0000
```

We get good agreement with the second time; the inverted duplication event that increased copy number.

Out.MutRate{top_no}{order_no} [This estimates the mutation rate per single genome copy]

```
>> out.MutRate{1}{1}

ans =

   0.4852
```

This is approximately equal to the value of 0.5 used to simulate the data.

Out.RearrChr{top_no}{chrom_comp_no}{order_no}{chr_in_comp_no}

[This describes the aberrant chromosomal segments in genomic order. The four columns are chromosome no., segment no., direction and a copy label for that segment used to construct the evolution matrices. Components with no rearrangements are blank and are thus wild type chromosomes]

```
>> out.RearrChr{1}{2}{1}{1}

ans =

    1    1    1    1
```

```
   1    2    1    1

   1    4    1    1

   1    4   -1    2

   1    2   -1    2

   1    5    1    1
```

This is the order and orientation expected when segment 3 is deleted and segments 2 and 4 form a subsequent inverted duplication.

out.SegmentalEvolution{top_no}{chrom_comp_no}{order_no}{set_no}{rearr_no}{chr_no}{seg_no}

[This describes the evolution matrices for the various chromosome, segment and rearrangement combinations]

```
>> out.SegmentalEvolution{1}{2}{1}{1}{2}{1}{4}

ans =

   1    1
```

This reflects the fact that the fourth segment of single chromosome in the second component of the allelic graph is duplicated.

Limitations

At present this software is just a command line beta version and does not produce any of the graphics in Greenman et al. and cannot deal with data with deleted breakpoints (such as unbalanced translocations). Future versions will address these issues.

Contact

Feel free to play with this software. Any feedback, ideas or modifications are welcome!

cdg@sanger.ac.uk