Chapter 3

Validation of methods

3.1 Introduction

In the previous chapter I have introduced the Battenberg algorithm for calling (subclonal) copy number from whole genome sequencing data, an approach to estimate the CCF values for SNVs and indels, and the DPClust method to infer the subclonal architecture from the distribution of CCF values measured in a single cancer. In this chapter, I will focus on validating the performance of these methods *in silico*. To this end, I have developed a subclonal architecture simulator called SimClone and metrics are introduced to evaluate the performance of a subclonal architecture caller. These metrics have a theoretical lower bound of performance, but a realistic upper bound does not exist (apart from the worst possible score). To set a realistic upper bound for a subclonal architecture caller, I introduce a series of simple, naive methods (termed RandomClone) that produce random subclonal architectures. An edited version of the text and figures describing SimClone, it's simulated data set and RandomClone will appear in the supplement of the PCAWG consensus subclonal architecture calling paper (Yu et al. 2017, manuscript in preparation). The simulated data set will be further used in Chapter 6 to compare the performance of subclonal architecture callers. Figure 3.6 is created by Maxime Tarabichi and is used with permission.

3.2 Simulating subclonality with SimClone

3.2.1 Introduction

SimClone was developed to evaluate the performance of DPClust. It can be used to generate subclonal architectures with underlying data that can test specific scenarios, or to build a set of random samples that can be used to evaluate overall performance. For it to be a

true evaluation it is important for the simulator to generate problems that are as realistic as possible. I have therefore aimed to build SimClone such that it can take high level characteristics of real data as input.

A typical workflow goes as follows: (1) A subclonal architecture is generated in the form of a phylogenetic tree with subclones and their locations. Relationships between nodes (mutation clusters) are created, where each node has a parent and sits at a particular level in the tree (the level of a node is determined as the minimum number of steps required to "walk" from the root of the tree to the node). (2) Each node on the tree is assigned a number of mutations. (3) Then a genome wide copy number profile is simulated, after which (4) mutations are simulated separately for each node, which requires the user to also specify a tumour purity value that is used for all mutation clusters. The user can provide input for steps 1, 2 and 3 to have full control over the solution to be simulated.

3.2.2 Assumptions

Both the mutation and wild type alleles are supported by a number of reads. I assume that the distribution of the number of mutation-supporting-reads takes on the shape of a binomial distribution. To model variation on the total number of reads covering the locus where the mutation has occurred, I assume that the depth can be modelled as a Poisson distribution. The mutation is carried by a number of chromosome copies (multiplicity). The shape of this distribution is partly determined by the copy number profile, that bounds the possible multiplicity states, and by cancer type specific development, i.e. if gains occur late there will be many SNVs on multiple copies, while if gains occur early there will be few. I model multiplicity through a Poisson, and learn the parameter that determines the shape of the distribution from real data. Finally, as a simplification, subclonal mutations cannot be carried by more or less than 1 chromosome copy.

3.2.3 Simulating a tree

A tree consists of nodes and edges, and each node has a parent that is either another node or the root. The tree simulation step generates a tree independently of other sample characteristics such as coverage, cluster sizes, etc. The procedure is provided with a number of nodes to place on the tree and the number of tries allowed to place each node.

The procedure starts by placing a root node, that represents mutations that are clonal. Then, iteratively, new nodes are placed until the required number is reached. Before a new node can be inserted SimClone first selects the new nodes parent, that resides in the tree at a level and in a branch. The level is selected by a draw from a uniform distribution that covers

all levels below the root in the current tree and a node at that level is selected to determine the parent with uniform probability.

For the new node to fit on the tree it must be assigned a CCF value such that the total CCF at the level of insertion does not exceed the CCF of the parent. The possible CCF space is therefore constrained. A further constraint can be placed (this is a user setting) in requiring that the new node position must be at least a minimum CCF away from its parent because clusters that are too close cannot be separated during clustering.

A CCF is then selected for the node between the max possible value and 0 as the CCF of the new node and the node is placed if it doesn't violate any of the constraints. The addition is retried with a new sampling of parameters if no suitable location is found, but the insertion is aborted if no location can be found after a specified maximum number of tries, resulting in one fewer node on the tree. This particular scenario is more likely when large numbers of clusters are requested as placed nodes will constrain the allowed space for new nodes. The user can then opt to either work with fewer nodes, rerun the procedure for an alternative tree or manually add extra nodes afterwards. Alternatively, a function is provided where a custom tree can be created.

3.2.4 Determining cluster sizes

The cluster size determination procedure takes the minimum and maximum total number of mutations in the tumour as input and an optional proportion of those mutations that are clonal. The total number of mutations is drawn from a uniform distribution between the minimum and maximum. I then determine cluster sizes by applying a stick breaking procedure where iteratively a randomly sized chunk is broken of the remaining stick. Each chunk then represents the proportion of total mutations that belong to a cluster. If a minimum proportion of clonal mutations is specified, then the first chunk will be constrained to be at least that specified size.

3.2.5 Simulating mutations

With node locations and sizes determined or provided as input SimClone now simulates the mutations per node. Further input is required in a copy number profile (the copy number simulation procedure is explained in the next section), a tumour purity value, coverage and a multiplicity λ parameter (also explained in the next section). These parameters are sample specific and clusters are therefore simulated independently per sample. Mutations are generated by calculating the expected number of reads reporting the mutation and wild-type alleles. But the multiplicity must first be determined before those can be calculated.

The multiplicity (m_m) is drawn from a Poisson distribution with the provided λ parameter as input.

$$m_m \sim \text{Pois}(\lambda) \tag{3.1}$$

The mutations are randomly assigned to a copy number segment in the provided profile. If the multiplicity is not possible given the major and minor allele of the selected segment I adjust it to the copy number of the major allele.

Then the number of reads per chromosome copy for the tumour (c_t) and normal (c_n) cells are calculated from the total coverage (C), tumour purity (ρ) and tumour ploidy (ψ_t). The total copy number of the normal cells is assumed to be 2:

$$c_t = C\frac{\rho}{\rho\,\psi_t + 2(1-\rho)} \tag{3.2}$$

$$c_n = C\frac{1-\rho}{\rho\,\psi_t + 2(1-\rho)} \tag{3.3}$$

Then expected number of mutant alleles r_m is determined by the multiplicity of the mutation, the mutations fraction of tumour cells (f) and the number of reads per tumour chromosome copy (c_t):

$$\mathbf{E}(r_m) = m_m f c_t \tag{3.4}$$

The expected number of wild type alleles r_w consists of three components: (1) Reads from normal cells (can be zero when the sample is pure and does not contain normal cells), (2) reads from whole chromosome copies from tumour cells that are not carrying the mutation (can also be zero when the copy number is 1+0) and (3) if the mutation is subclonal, an additional number of reads from cells that are not part of the subclone that carries the mutation:

$$\mathbf{E}(r_w) = 2c_n + m_w c_t + m_m(1-f)c_t \tag{3.5}$$

The total number of observed reads are then drawn from a Poisson distribution:

$$r_d \sim \text{Pois}(\mathbf{E}(r_m) + \mathbf{E}(r_w)) \tag{3.6}$$

And the final mutant and wild type alleles are determined by a draw from a binomial distribution:

$$r_m \sim \text{Bin}(r_d, \frac{\mathbf{E}(r_m)}{\mathbf{E}(r_m) + \mathbf{E}(r_w)}) \tag{3.7}$$

3.2.6 Extension to simulating multi-sample cases

The above procedure is already extended to simulate multi-sample cases. During my Ph.D. I have mostly worked with single-sample cases and this thesis contains results on that type of data only. I have therefore opted not to include any multi-sample simulations and validations.

The tree building step can be provided with an additional parameter that specifies the number of samples a multi-sample case should contain. It then simulates mutation clusters with (potentially different) CCF values in all the samples. However, it does require one single node that is clonal in all, but that node does not need to contain any mutations (i.e. to simulate multi-focal tumours). The procedure to simulate mutations for each cluster can take a multi-sample tree as input and it then simulates the mutations belonging to that cluster with CCF values for all the requested samples.

It is currently not possible to simulate multi-sample copy number profiles. One could use the same copy number profile or run the copy number simulator a number of times on the same input data. The method can be adapted in the future to simulate copy number profiles for multi-sample cases where the samples share a number of common alterations.

3.2.7 Simulating copy number

A copy number profile consists of segments and a certain number of copies are available for every segment. SimClone simulates copy number in three steps: (1) it selects a segmentation from a catalogue, (2) then it models the total copy number profile and (3) it breaks down the total copy number into allele specific contributions. Fig. 3.1 shows an example of a real copy number profile (top) and a simulation inspired by it (bottom). The described approach is simulating clonal copy number only.

A segmentation is selected either randomly from a catalogue of real segmentations, or can be chosen as containing only whole chromosome or whole chromosome arm segments. The

total copy number is then modelled through a Poisson distribution, where the λ parameter is learned from a real copy number profile. Before learning the λ parameter I first subtract 1 from the total copy number and after drawing from the learned distribution I add one to the simulated total copy number because the Poisson distribution often draws many zeroes. This means SimClone does not simulate homozygous deletions.

$$n_{tot} \sim \text{Pois}(\lambda) \tag{3.8}$$

Total copy number is drawn from the learned distribution and assigned to randomly selected copy number segments, until the fraction of the genome covered by total copy number distribution looks similar to that of the real tumour (Fig. 3.2 shows the total copy number distributions of the real and simulated profiles shown in Fig. 3.1). Often there is a minor discrepancy between the distribution from real and simulated data inspired by the real sample due to the random assignment of total copy number to segments. The real distribution can often only be obtained by recreating the real profile exactly, which is what SimClone aims to avoid.

The total copy number is then broken down into separate contributions from two alleles to obtain allele specific copy number by using the multiplicity distribution of the SNVs from the real sample. Multiplicity values are drawn from a Poisson distribution with its λ parameter

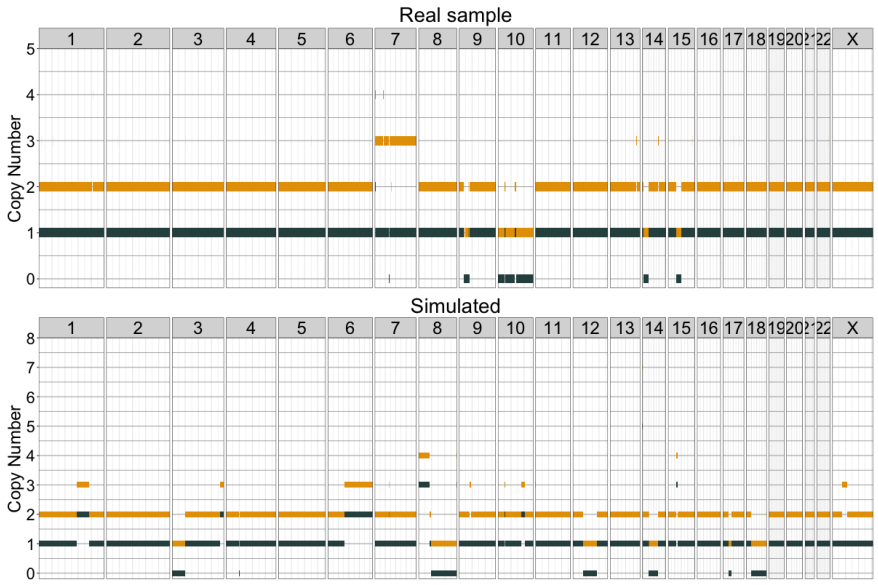


Fig. 3.1 Genome wide overview of a real copy number profile (top) and a simulation that is inspired by the real profile (bottom).

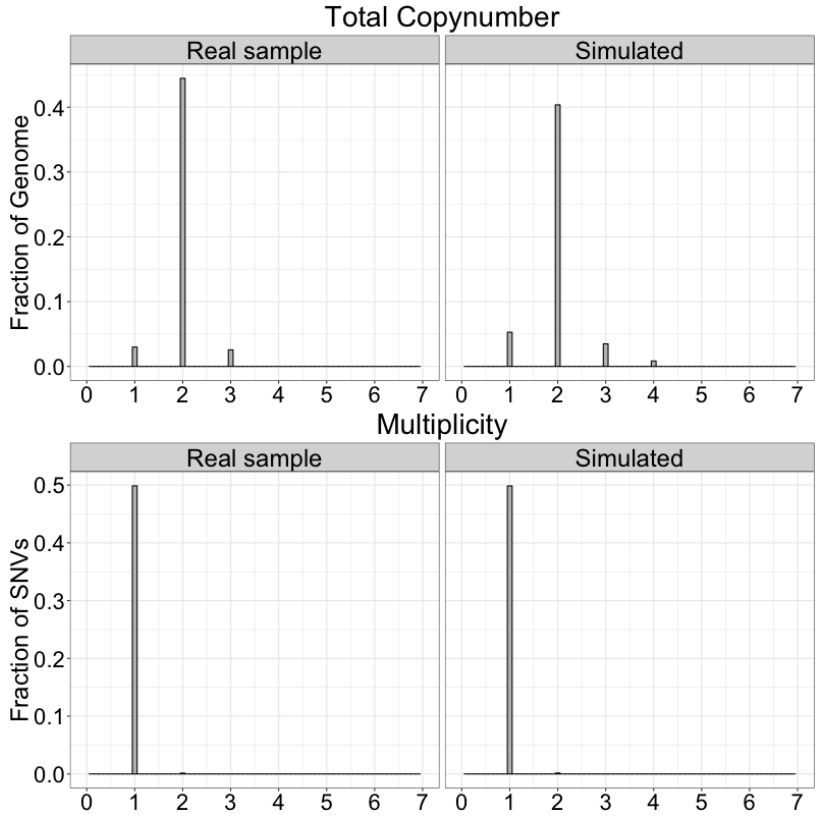


Fig. 3.2 Comparison of real distributions (left) with distributions of simulated data. The figures on the top row show that the copy number states distribution of the simulated data follows that of the real data, but there is a noticeable discrepancy. The algorithm aims to approximate the total copy number states distribution in the real tumour as closely as possible by iterating over the observed copy number states and assigning the state to a randomly selected segment until similar proportions of the genome are covered. But due to the variability in segment lengths it is not always possible to exactly match the real distribution. The multiplicity distribution (bottom) closely resembles that of the real sample.

learned from the real data (Fig. 3.2). A multiplicity value is drawn for each SNV and SNVs are assigned to segments (this assignment is purely for establishing copy number) where the total copy number is greater than or equal to the multiplicity of the SNV. The maximum multiplicity, m_i, assigned to segment i is then used to determine the one allele, $n_{A,i}$.

$$n_{A,i} = \max(m_i) \tag{3.9}$$

The other allele, $n_{B,i}$, is then determined by subtracting the copy number of $n_{A,i}$ from the total copy number $n_{tot,i}$:

$$n_{B,i} = n_{tot,i} - n_{A,i} \tag{3.10}$$

Finally, the major allele for each segment is established as the allele with the highest copy number state, the minor allele is the allele with the lowest copy number state.

This procedure ensures that if many SNVs have a high multiplicity state, then many segments will be created with an allele specific copy number configuration that can support them. If, for example, the multiplicity and total copy number distributions are very similar, then the profile will have many major alleles that closely follow the multiplicity, leading to a profile with much loss of heterozygosity (Fig. 3.3). By randomly distributing copy number states across the genome it becomes possible to create very difficult and truly chaotic profiles to test a methods' limits (Fig. 3.4)

The procedure however doesn't restrict particular copy number states to particular chromosomal areas. That means the actual copy number profile will most likely not resemble the profile used as inspiration. But it should be covered by the same allele specific copy number state combinations in similar proportions, if the same segmentation is used.

A different choice in segmentation can cause a bigger discrepancy between the real and simulated distributions of total copy number and multiplicity. Fig. 3.5 shows three simulations inspired by the same real profile, but with segmentations that are restricted to whole chromosome arms (middle) or whole chromosomes (bottom). The real tumour consists for large parts of normal copy number with a few large and small scale alterations. When restricting segments to whole chromosomes, many segments are too large for a reasonable approximation of the proportion of the genome covered by total copy number. SimClone therefore tends to pick small segments, that reside on the small chromosomes.

In the future it could be interesting to experiment with an additional catalogue of cancer type specific common events to create more biologically accurate copy number profiles. However, currently the aim is to simulate the effect of copy number on the VAF of SNVs, for which it does not matter where on the genome the alterations are placed.

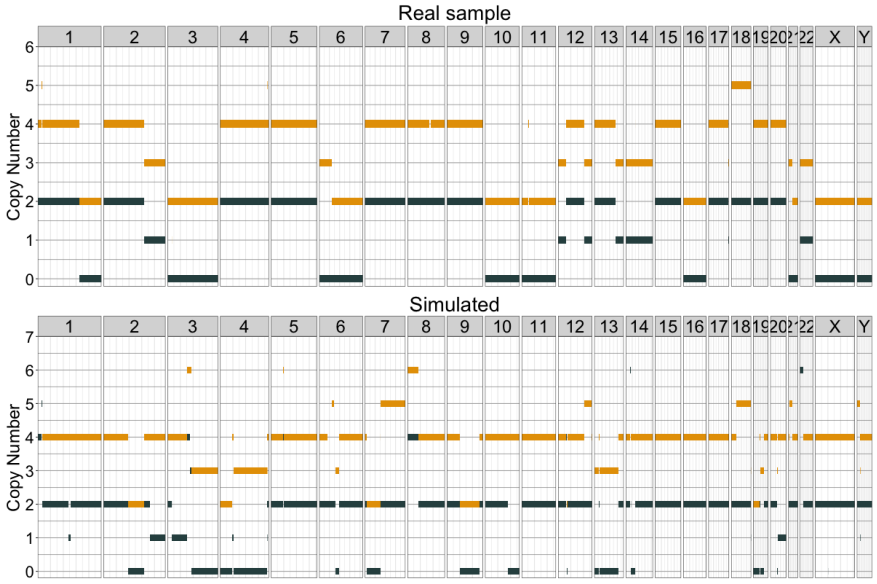


Fig. 3.3 A copy number profile simulation that is inspired by a real tumour with loss of heterozygosity and a whole genome duplication.

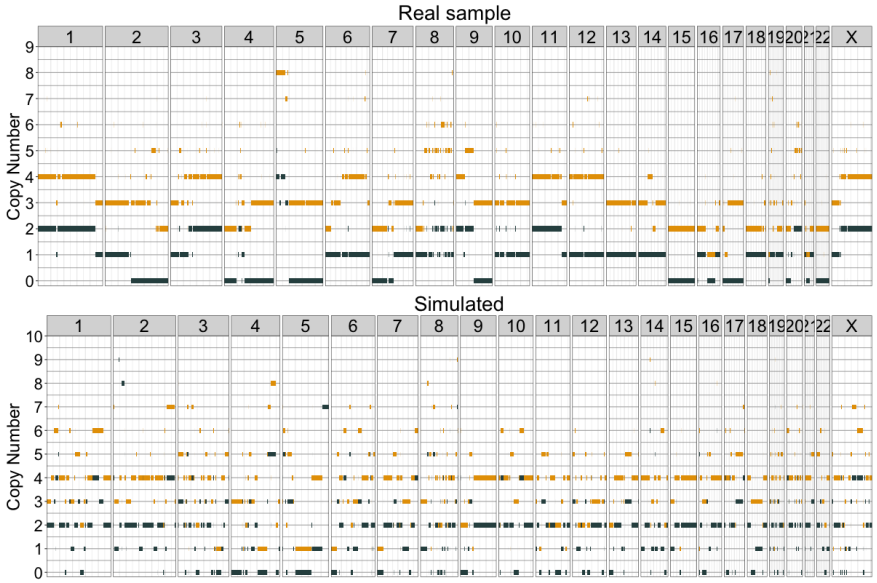
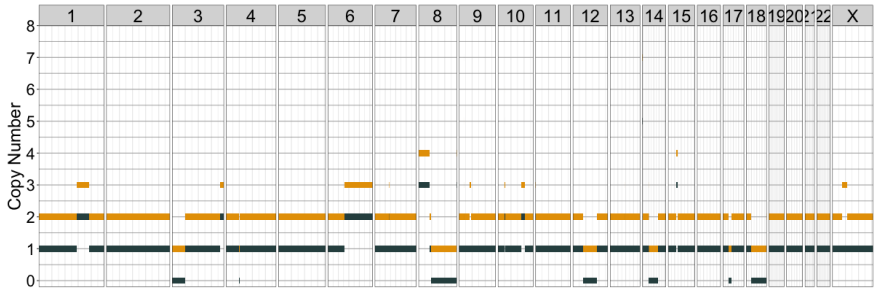
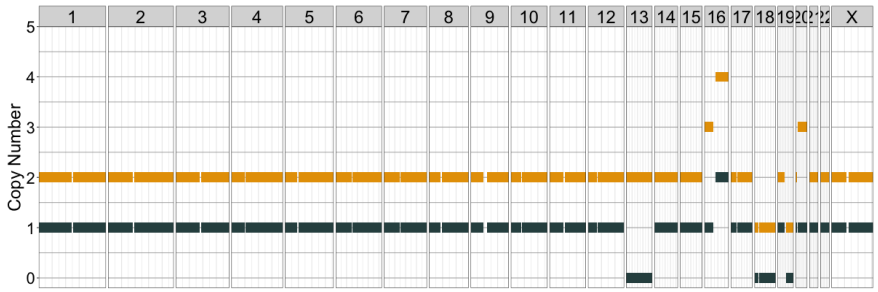


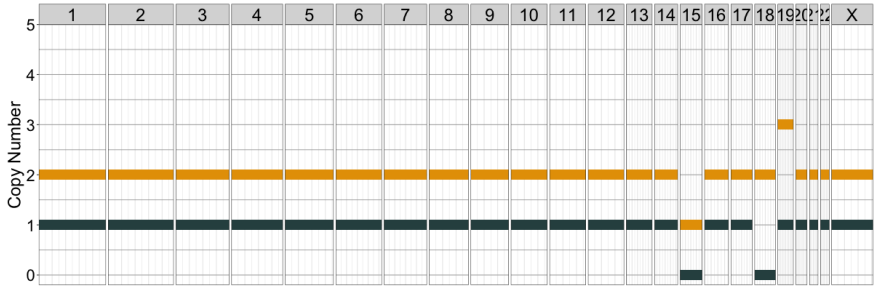
Fig. 3.4 Example of a simulation based on a very fragmented and messy real copy number profile. The random assignment of copy number to segments creates a chaotic simulated profile.



(a) Segments from reference profile



(b) Chromosome arm segments



(c) Whole chromosome segments

Fig. 3.5 Simulations using the same real sample as inspiration, but with different segmentations: (a) The segmentation of the real sample (b) segmentation where each segment is a full chromosome arm and (c) segmentation where each segment is a whole chromosome. The simulator aims to approximate the proportion of the genome covered by certain copy number states. The real tumour (shown in figure 3.1) contains a few large scale and a few small scale alterations, but consists mostly of normal copy number. Due to the large segment sizes in (b) and (c) it therefore tends to place alterations in the smaller segments that reside on smaller chromosomes.

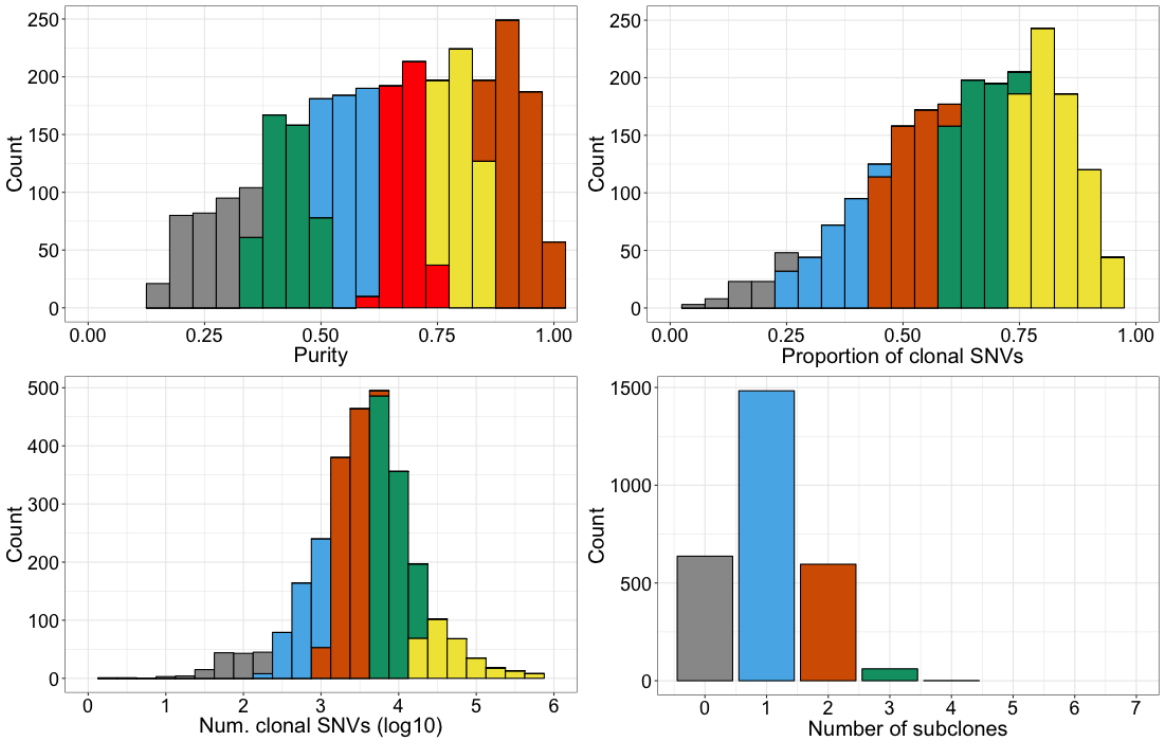


Fig. 3.6 The simulated data set was created as a grid with four axis. Each axis represents a type of measurement that can be obtained from real data. This figure shows the histogram of these four measurements from the PCAWG data and the colours represent bins along each grid axis. A simulated tumour falls somewhere on the grid, which amounts to a combination of 4 bins (one on each axis). The parameters for this sample are then generated by sampling a single value from each of the 4 bins.

3.3 SimClone1000, a validation data set for PCAWG

SimClone was used to simulate 1000 tumours with the aim to evaluate the performance of subclonal architecure callers within PCAWG. The data set consists of 700 unique subclonal architecture simulations and 300 cases where the exact subclonal architecture was simulated a second time on a copy number profile without any alterations. The 300 paired cases allow us to investigate whether subclonal architecture callers perform better without having to adjust for copy number alterations. We created a grid with four key parameters by which tumours vary when considering their subclonal architectures: Purity, the fraction of clonal SNVs, the number of clonal SNVs and the number of subclones. Of the 1000 tumours 36 yielded too few mutations (less than 20) or did not complete the simulation process due to time constraints. The data set therefore contains 964 tumours.

The axis of the grid were determined based on the distribution of each of these values in the PCAWG data set, shown in Fig. 3.6. By applying k-means clustering we obtained 6

purity clusters, 5 clusters for the number of clonal mutations (with one cluster fixed at 10^5 to represent hypermutators) and 5 clusters for the fraction of clonal mutations (one cluster was fixed at 0.995 to represent a typical hypermutator). The grid axis for the number of subclones was determined by creating 4 classes corresponding to 0, 1, 2 and 3+ subclones, where the 3+ category contains tumours with 3 to 7 subclones. A single tumour is then assigned a purity drawn from a bin on the purity axis, a number of clonal mutations from a bin on the clonal mutations axis, etc. This results in combinations of real parameters, but they may not have been observed as a combination. The 6-by-5-by-5-by-4 grid yields 600 unique combinations of parameters, which we extended by sampling another 100 combinations to reach 700.

Copy number profiles were chosen at random from the PCAWG data set. Once a real purity is selected for a particular simulation we also assigned it the copy number profile of the real tumour. We simulated 300 tumours a second time without copy number alterations and therefore only allowed tumours with at least 10% of their genome altered to be included in the grid to not inflate the number of quiet diploid tumours. From the 700 simulations we selected 300 at random for another simulation with normal diploid copy number. A change in ploidy affects the number of reads per chromosome copy, when purity and coverage remain equal, potentially altering the CCF space of the simulation substantially and making the envisaged comparison difficult.

In the regular simulation we, for example, do not have sufficient power to simulate SNVs at a CCF below 0.3. This means that a the distributions of mutations belonging to a subclone at 0.4 CCF will be truncated as some of its mutations cannot be represented by a number of supporting reads greater than 0. When the number of reads per chromosome copy is increased we gain power to simulate subclonal mutations, resulting in a lower CCF cutoff point. That means we can simulate more mutations of the 0.4 CCF subclone, making it potentially easier to correctly identify it by subclonal architecture callers, and rendering a comparison between the diploid and non-diploid simutations uninformative.

We therefore opted to adjust the purity of the non-diploid copy number profile (ρ_n) to correct for the reads per chromosome copy shift when changing the ploidy from the real sample (ψ_n) to create a purity for the diploid simulation ρ_d:

$$\rho_d = \rho_n \frac{2}{\psi_n} \tag{3.11}$$

Subclone positions and sizes were determined as described in the previous section about SimClone. And coverage was fixed to the PCAWG average of 48.46621.

3.4 Metrics to evaluate a subclonal reconstruction

To aid the large scale performance evaluation of tumours on simulated data we developed three metrics around two key descriptions of a subclonal architecture: Clusters (location and size) and assignments of mutations to clusters. The metrics compare a provided subclonal architecture against a known truth. It's also possible to use these metrics to measure how similar a pair of solutions are, which is used later in this thesis to compare performance of subclonal architecture callers.

The overall subclonal architecture can be roughly described by the number of subclones (π) and the proportion of clonal mutations (θ). Eqs. 3.12 and 3.13 capture the absolute difference between solutions k and l. For both metrics, the lower the score, the better.

$$\frac{|\pi_k - \pi_l|}{\pi_k + \pi_l} \tag{3.12}$$

$$\frac{2|\theta_k - \theta_l|}{\theta_k + \theta_l} \tag{3.13}$$

Comparing the cluster locations is more complicated, because the solutions to be compared may not contain the same number of clusters. Instead, we use the mutation assignments. Each mutation is hard-assigned to a cluster, and each cluster has a location. For each mutation i we compare the CP of the assigned cluster between solutions k and l. A small distance across all mutations reflects a good concordance between the solutions. Eq. 3.14 calculates the average difference in CP ($\varphi_{i,k}$ is the cellular prevalence assigned to mutation i by method k), where a lower value is better. The score is divided by the tumour purity (ρ) to correct for purity differences between tumours.

$$\frac{\sqrt{\frac{1}{N}\sum_{i=1}^{N}(\varphi_{i,k} - \varphi_{i,l})^2}}{\rho} \tag{3.14}$$

3.5 A lower bound generated by RandomClone

3.5.1 Introduction

The metrics described above can be used to assess the performance of a subclonal architecture caller, where low scores mean a method is performing well. Often methods will not get the perfect solution and therefore their scores will deviate from the perfect score. What is not clear however is when performance can be described as poor. I reasoned that a caller should be able to outperform a simple random method. Running a random method on the same

data as the caller would then provide an upper bound of what can be considered reasonable performance. To this end I have developed three simple methods that generate random subclonal reconstructions and one method that returns only clonal tumours.

3.5.2 RC – Stick breaking

The stick breaking method starts with drawing a random number between 0 and 6 to determine the number of clusters. It then orders the mutations by their CCF and iteratively breaks a randomly sized chunk of the ordered mutations. Each of these chunks represents a mutation cluster and its location is obtained by taking the mean CCF of the mutations in the chunk. Mutations are automatically assigned by belonging to a particular chunk. The advantage of this approach is that it is more likely to place a cluster where there are large real clusters, but it does also tend to place multiple clusters within large real clusters.

3.5.3 RC – Informed

The downside of the stick breaking approach described above is that it performs a single series of breaks and returns that as a solution. I wondered whether the method could be improved by selecting the best solution from a series of random models. The informed method runs the stick breaking implementation described above 100 times. Contrary to the above approach, the informed method records the size and locations of clusters, but does not record the assignments. It runs the MutationTimer approach (used in Gerstung et al. (2017) to assign mutations), which models each mutation cluster as a beta-binomial and takes into account the size of the cluster. MutationTimer then calculates the proportion of mutations that are poorly explained (i.e. fall in the outermost 5% of the beta-binomial distributions). This proportion is calculated for all 100 runs, after which the run that yields the lowest value is selected as the returned subclonal architecture.

3.5.4 RC – Uniform

The uniform approach is an alternative to stick breaking. It starts with drawing a random number between 0 and 6 to determine the number of clusters to be found. Then that number of draws are made from a uniform distribution that has as minimum the lowest 5% of the mutation CCF space and as maximum the highest 5%. That means cluster locations are drawn from within the CCF space that is occupied by mutations. Mutations are then assigned to clusters by calculating the binomial likelihood per mutation and cluster, and assign to the most likely cluster. This approach does not utilise the shape of the CCF space and therefore

usually places clusters in unexpected locations. As it is nearly always outperformed by the stick breaking approach I omit results from this method.

3.5.5 RC – Single cluster

This approach places a single cluster to explain the data. It can obtain the single cluster by taking the mean mutation CCF, or it can be forced to place a clonal cluster at a CCF of 1. All mutations are assigned to the single cluster.

3.6 Validation of multiplicity calls

DPClust takes as input a fixed multiplicity value for every mutation. That value is obtained during pre-processing using the equations from the previous chapter. Incorrect multiplicity values could artificially alter the CCF space, which DPClust may explain through extra mutation clusters. I therefore used the over 42 million mutations from the 964 simulations to validate the performance of this pre-processing step. Table 3.1 contains the proportion of mutations with a correct multiplicity for four different splits of the data and shows that over 99% of mutations are assigned the correct multiplicity.

The subset of mutations that are gained (i.e. have a multiplicity greater than one) has a lower success rate at just over 93%. For most of these mutations there is not much ambiguity about their multiplicity, but some will fall between two multiplicity states. The addition of binomial noise to the reads supporting the mutation and reference alleles can cause the mutation to shift away from the correct multiplicity. The DPClust pre-processing considers the evidence provided and assigns the most likely multiplicity, which, due to the noise, can be incorrect.

3.7 Assesment of a subclonal architecture through resimulations

In addition to the metrics described earlier I have developed a measurement that aims to capture how well a subclonal architecture explains the raw CCF space it is provided as input. The idea is that if the subclonal architecture called by a method is the true architecture, then their corresponding CCF spaces should be very similar. A distance metric can then be used to calculate the difference, where a small deviation would serve as a good score because the called subclonal architecture describes the observed input data very well.

Evaluation of multiplicity calls across 964 simulations

Type	Total	1st Qu.	Median	Mean	3rd Qu.	S.D.
All	42,536,567	0.9915	0.9992	0.9917	1.0000	0.0186
Gained	2,125,495	0.9195	0.9808	0.9336	0.9987	0.1071
CNA	11,869,768	0.9800	0.9946	0.9840	1.0000	0.0277
CNA & Gained	1,756,994	0.9121	0.9762	0.9311	0.9973	0.1014

Table 3.1 Multiplicity values are compared between the truth and the DPClust calls. Proportions of mutations correct (1st quartile, median, mean and 3rd quartile) are shown for four different splits of the data: All mutations, mutations on more than one chromosome copy (Gained), mutations in regions of aberrant copy number (CNA) and mutations that are in a region of aberrant copy number and are gained (CNA & Gained). The table shows that over 99% of mutations are assigned a correct multiplicity value. Most mistakes are made with gained mutations. This pertains to mutations that fall exactly between two multiplicity values and the binomial noise pushes the mutation away from the correct multiplicity. Even in that scenario over 93% of mutations are assigned the correct value.

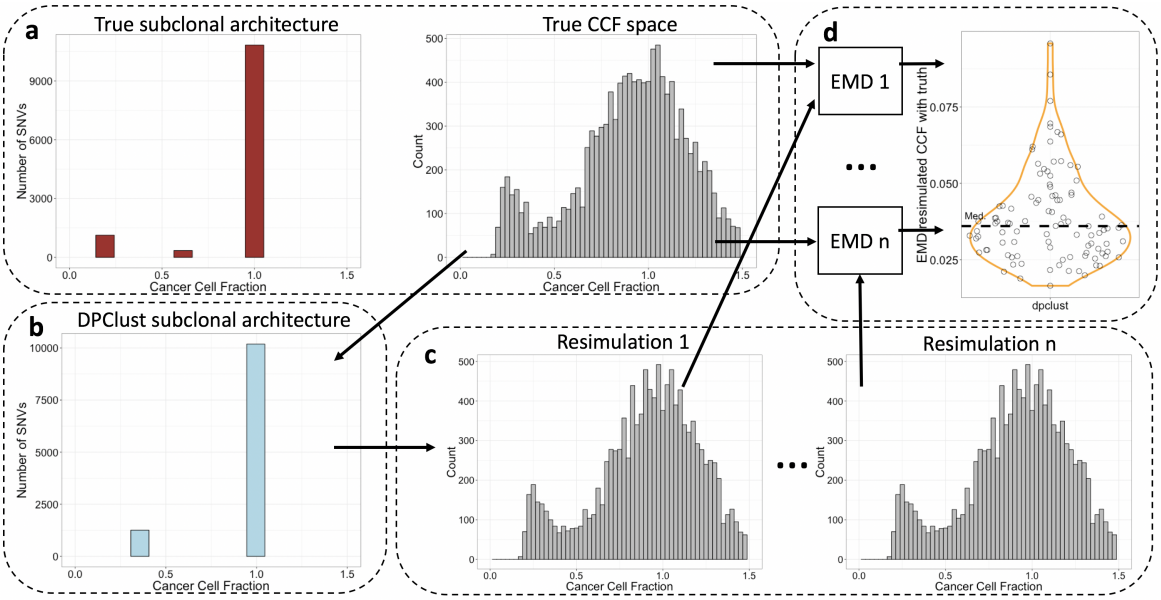


Fig. 3.7 (**a**) SimClone generates a true subclonal architecture and its associated CCF space. (**b**) The mutations that make up the true CCF space are provided as input to DPClust, which returns a subclonal reconstruction. (**c**) That reconstruction serves as input to SimClone for resimulations, which returns a number of CCF spaces. (**d**) The EMD is calculated between each resimulation and the true CCF space that measures how well the DPClust subclonal reconstruction is explaining the true CCF space. The EMD distribution is summarised by the median, resulting in a single similarity value per sample. This similarity value can be compared across methods and across samples.

The approach is illustrated in Fig. 3.7. For a single tumour SimClone returns (among other things) a subclonal architecture with cluster locations and sizes, and the simulated mutations form a CCF space. DPClust takes the CCF space as input and returns a subclonal architecture. That obtained architecture is then fed back into SimClone a 100 times (this process is referred to as *resimulation*), yielding 100 CCF spaces. The earth movers distance (EMD) is then calculcated between each resimulated CCF space and the true CCF space. A summary of the resulting histogram can then be used as a metric of performance.

When provided with a subclonal architecture SimClone simulates mutations with binomial noise on the number of supporting reads and Poisson noise on the coverage. It is therefore expected that a pair of resimulations also differ, hence 100 resimulations are performed for every called subclonal architecture by DPClust, the RandomClone methods and for the truth. The distribution of EMDs obtained from resimulating the truth can then be compared with the EMDs from the DPClust and RandomClone solutions.

Figure 3.8 shows the EMD distributions for a selected simulated tumour for the truth, DPClust and the three RandomClone methods, while table 3.2 contains the raw results and scores. It shows that the stick variant of RandomClone comes closest to the expected number of subclones and DPClust does best on the fraction of clonal mutations. The RMSE scores are very close, suggesting the CCFs of the clusters to which mutations are assigned in general are close to the expected value. The median EMD (dashed horizontal line) tells us that DPClust explains the CCF space best, followed by the informed RandomClone variant.

To summarise the EMDs (for a whole data set comparison in the next section), relative to the variation obtained from resimulating the truth, I then calculate the following score:

$$1 - \frac{1}{n}\sum_{i=1}^{n} I(e_t, e_m) \tag{3.15}$$

This score is obtained by sampling n pairs of values with replacement from the truth and from one of the methods' EMD distributions and determining whether the EMD of the method (e_m) is greater than the EMD from the truth (e_t). The index function in eq. 3.15 returns a 1 when e_m is greater than e_t. n is set to 1000.

If the method has returned a subclonal architecture that explains the CCF space very well, then a score of 0.5 is returned. A value greater than 0.5 means the method has not perfectly described the true CCF space, with a higher value meaning a bigger discrepancy. Finally, a lower value means the method is better at explaining the true CCF space than resimulations of the truth do (i.e. the method is overfitting on the input data).

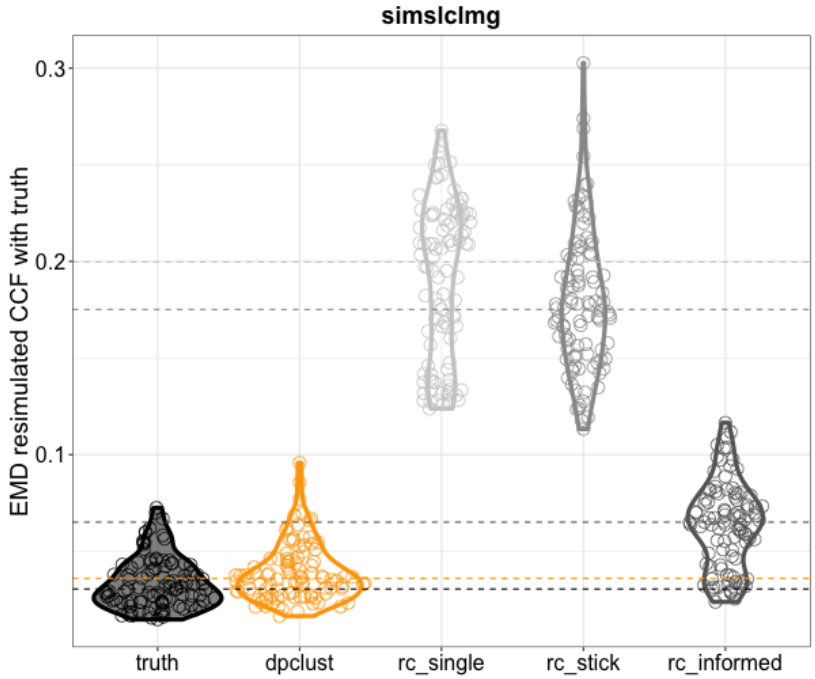


Fig. 3.8 Earth movers distance (EMD) of the CCF space between the truth and resimulations of solutions found by DPClust and the randomclone methods. Resimulating the truth provides a lower bound of what is obtainable, while the random methods provide an upper bound. The similar median (dashed lines) EMD that of the truth suggests DPClust has found a solution that explains the true CCF space quite well.

Evaluation of performance on sample simslclmg

Method	Calls		Scores			
	Num. subcl.	Frac. clonal	Num. subcl.	Frac. clonal	RMSE	EMD
Truth	4	0.196				0.030
DPClust	2	0.235	0.491	0.197	0.007	0.036
RC single	0	0.000	0.999	1.000	0.011	0.199
RC stick	3	0.000	0.221	0.931	0.006	0.175
RC informed	1	0.312	0.822	0.592	0.008	0.065

Table 3.2 A comparison of the scores on simulated tumour simslclmg reveals that the scores capture different characteristics of the reported solutions. RandomClone-stick, for example, comes closest to the true number of subclones and therefore receives the best score in that category. However, it assigns very few mutations to the clonal cluster and therefore attains a poor fraction of clonal mutations score.

3.8 Validation of DPClust

Figure 3.9 shows the outcome for DPClust and the RandomClone methods for the three PCAWG scores and the resimulation metric on the SimClone1000 data set. For the three

PCAWG metrics a lower score means better, with zero being perfect. DPClust comfortably outperforms the three random methods on number of subclones, fraction of clonal mutations and mutation assignments. The resimulation score is expected to be 0.5 when DPClust finds a perfect solution. A value higher than 0.5 represents a drop in performance, while values below 0.5 can be considered overfitting. DPClust also outperforms the random methods on the resimulation score. DPClust is evaluated on the three PCAWG scores against 10 other subclonal architecture callers in Chapter 6.

The scores indicate that DPClust performs well. However, it does not always find the exact true subclonal architecture. Figure 3.10 is an attempt to explore where the differences lie. In nearly half the tumours DPClust calls the correct number of subclones and in those cases the proportion of clonal mutations is close to the truth, indicating cluster locations have been called in roughly the correct locations. For the other cases DPClust nearly always undercalls the number of subclones. Where undercalling occurs, DPClust often calls a larger proportion of mutations clonal. This suggests it merges a nearby subclone into the clone.

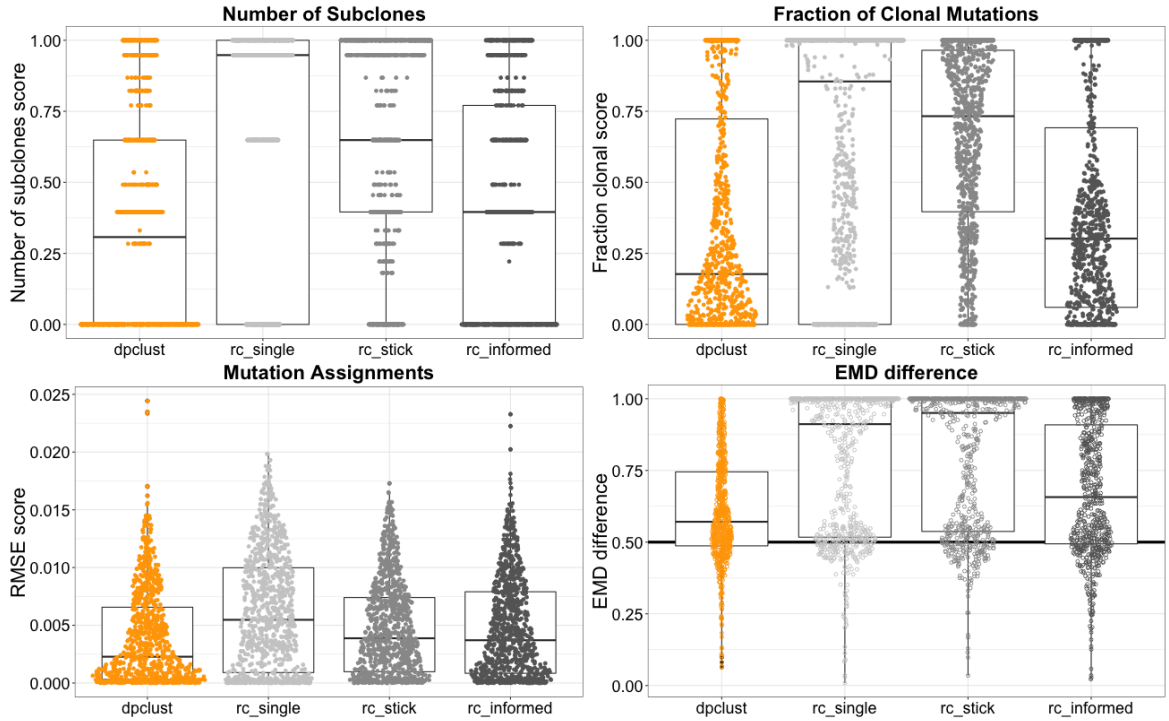


Fig. 3.9 General overview of the four scores obtained on the SimClone1000 comparing DPClust against the the truth and performance by a random method. The best score is 0 in the first three metrics and 0.5 in the fourth. All four scores show that DPClust easily outperforms the random callers and does well on calling the fraction of clonal mutations (top right), the mutation assignments (bottom right) and explaining the true CCF space (bottom left). The discrepancy in the number of called subclones is further explored in figures 3.10 and 3.11.

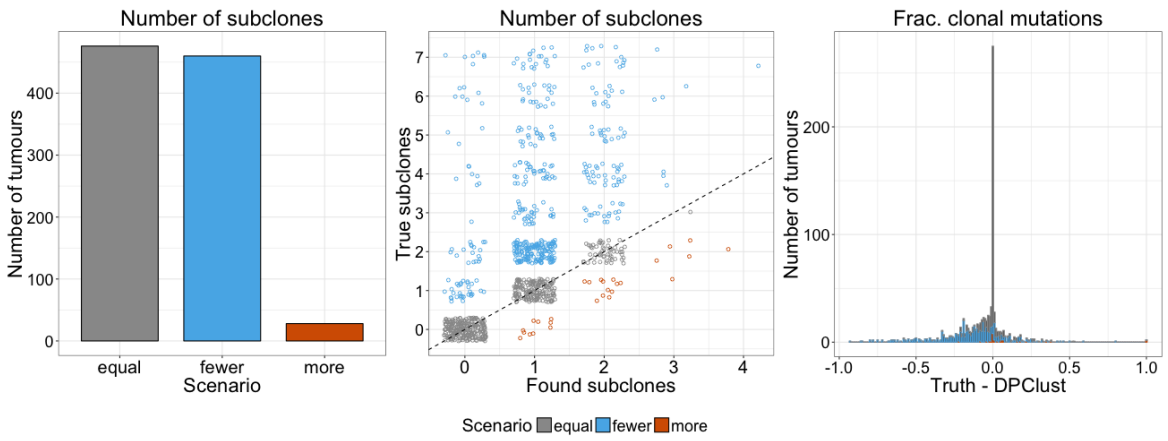


Fig. 3.10 The trend across the 964 simulated tumours is that DPClust calls the correct number of subclones in nearly half of the cases, in the other half it nearly always undercalls by one or more subclones (left). Undercalling occurs in two major scenarios: Cases where in reality there are two subclones, but DPClust calls a single subclone and cases where the true number of subclones is 3 or more (middle). This affects estimates of the fraction of clonal mutations, where in cases where DPClust undercalls it often overestimates the fraction of clonal mutations (right). Combined these results suggest DPClust can be considered conservative in its statements about the amount of subclonality found in a data set. An explanation where the number of subclones discrepancy comes from is explored in Fig. 3.11.

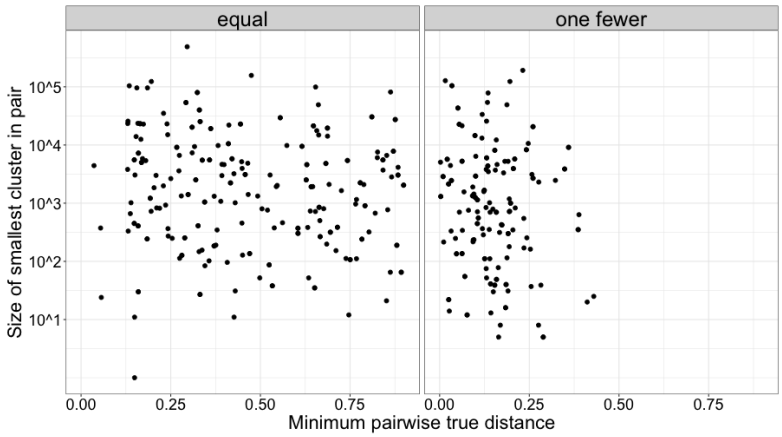


Fig. 3.11 The results in Fig. 3.10 suggest that subclones are merged in about half of the cases within the SimClone1000 data set. This figure compares the distance between the closest pair of clusters (x-axis) and the size of the smallest subclone within that pair (y-axis) for cases where DPClust finds the correct number of subclones (left, clonal tumours omitted) and where it finds a single subclone where two are expected. The data show that merging of clusters occurs when a pair of clusters are within 0.25 CCF of each other, regardless of their size.

Figure 3.11 shows this phenomenon, with on the x-axis the CCF difference of the two closest mutation clusters and on the y-axis the number of mutations that belong to the smallest cluster of the selected pair of clusters. When comparing these data between cases where DPClust finds the correct number of subclones (clonal tumours are excluded) and cases where it finds a one subclone where two are expected, it is clearly visible that in the latter category merging occurs frequently when the distance between a pair of clusters goes below 0.25 CCF. The size of the cluster (and the size difference, data not shown) plays little to no role in the separability of clusters.

DPClust assumes each of these clusters is its own statistical distribution influenced by binomial noise. In this merging scenario a pair of clusters are significantly overlapping and DPClust cannot find sufficient evidence of there being two clusters. It therefore takes the conservative approach and calls one fewer subclones, protecting against overstating the amount of heterogeneity.

These analysis explain that discrepancy between the truth and the DPClust solutions can be explained by clusters that are too close to separate. It shows that tumour heterogeneity estimates based on DPClust are reliable, yet a conservative underestimate of the true amount of heterogeneity.

3.9 Validation of assignments of gained mutations

A gained mutation (i.e. a mutation on more than 1 one chromosome copy) is assumed to be clonal by the described procedure to establish a mutation's multiplicity in the previous chapter. It is the maximum parsimony explanation when, for example, a mutation appears to be reliably carried by two chromosome copies, and the local copy number allows for this to occur (there are two copies of at least one of the two alleles). In such a scenario one would expect DPClust to assign this mutation to the clonal cluster, but DPClust is not constrained and can assign a gained mutation to a subclone. I therefore set out to investigate how often this occurs.

Across the SimClone1000 data set an average 10% of mutations are gained (Fig. 3.12, top and middle). Nearly all these gained mutations are correctly assigned, with an average of 94% of mutations assigned to the clone (Fig. 3.12, bottom). However, in some samples this percentage is much higher, in some cases nearly all gained mutations are incorrectly assigned. Furthermore, samples in which a very high proportion of gained mutations are assigned incorrectly contain a large number of mutations overall.

Investigation revealed that in many cases clearly clonal mutations (i.e. mutations with a CCF near or greater than 1) were assigned to a subclone (sample sim01bxzd is provided

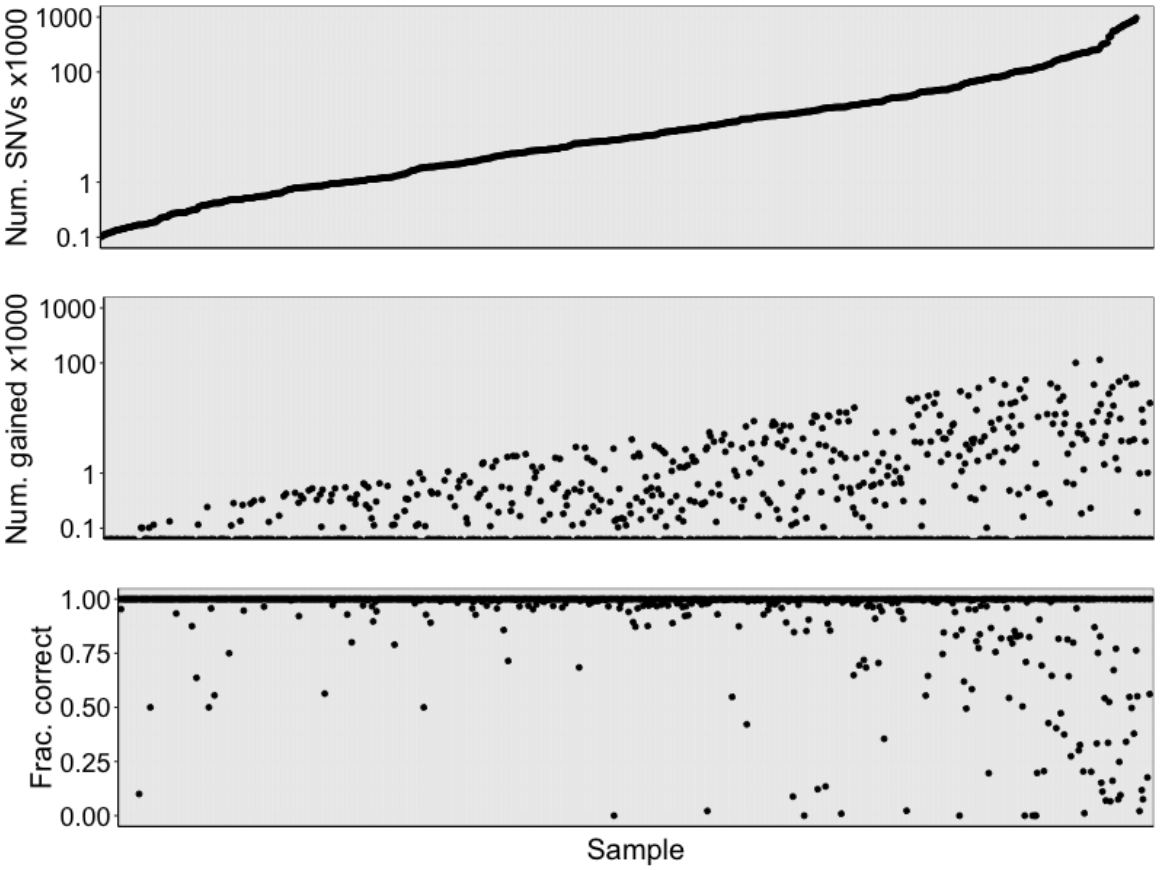


Fig. 3.12 The figure shows the total number of mutations (x1,000) at the top, the number of gained mutations (x1,000) in the middle and the fraction of those gained mutations that are correctly assigned to the clone at the bottom. A high proportion of incorrectly assigned mutations is concentrated in the samples with a high number of mutations, which lead to the exploration of a number of DPClust runs with different parameters.

as an example in Fig. 3.13), often forming a clear cluster around the location of the clone. Combined with the observation that samples with poor performance often contain many mutations, I postulated that these represent cases where the MCMC chain had not yet converged. At each MCMC iteration a mutation is assigned to the most likely cluster and after all iterations are complete these assignments are amalgamated into the most likely call. If the chain has not yet converged it is still in a state of flux, where mutation assignments can be volatile, leading to an increased likelihood of assignment to an incorrect cluster.

I therefore performed additional runs on a number of samples that show a poor assignment performance. I increased the total number of iterations and also altered the number of iterations that are discarded as burn-in, leading to the following combinations: 1,250 iterations with 250 burn-in (default), 2,000 iterations with 1,000 burn-in, and to runs of 5,000 with 4000

burn-in and 10,000 with 9,000 burn-in. The proportion of incorrectly assigned mutations for sample sim01bxzd (Fig. 3.13) directly decreases from 16% to 0.4% when increasing to 2,000 iterations and 1,000 discarded as burn-in. This behaviour is consistent across nearly all selected samples (Fig. 3.14), highlighting that the number of iterations the MCMC chain is run for by default is too short and should be increased to at least 2,000 and 1,000 iterations should be used for burn-in.

More iterations did not have the desired effect on all samples. Figure 3.15 shows one such cases, sample sim6zrlr0 as an example. It contains a vast subclone, that is truncated on one side, while it engulfs the clonal cluster on the other side (top row). In such an extreme case it becomes difficult to separate a pair of clusters, which causes a good number of clonal mutations to be assigned to the subclone (middle and bottom). in a scenario where clusters are within each others space it will be difficult to reliably assign mutations. A further adjustment should be made that prohibits gained mutations to be assigned a subclone.

3.10 Validation of Battenberg

The SimClone1000 data set cannot be used to assess the performance of Battenberg, because SimClone simulates just the final copy number profile and not the underlying data. For this validation I therefore introduce a separate data set that contains manually created subclonal architectures and copy number profiles that have subsequently been embedded into a BAM file using BAMsurgeon (Ewing et al., 2015). These tumours have been created for the Somatic Mutation Calling - heterogeneity (SMC-het) project that aimed to evaluate performance of subclonal reconstruction algorithms, where Battenberg copy number profiles were provided to all participants. The organisers of SMC-het used the high coverage NA12878 (Zook et al., 2016) and the previously sequenced parents of NA12878 (NA12891 and NA12892) as part of the 1000 Genomes project (1000 Genomes Consortium, 2012) to obtain the maternal and paternal genome that make up the genome of NA12878, which allows for creation of haplotype correct copy number profiles (details will be available in Salcedo et al. (2017, manuscript in preparation)).

In total 50 tumours were designed by the SMC-het team, inspired by real tumours reported in literature and from PCAWG. The copy number profiles were limited to whole chromosome alterations, as this is a requirement for BAMsurgeon, and subsequently 50 BAMs were generated. For the purposes of this validation I have excluded 8 cases that were designed as subclonal architecture corner cases as these all have the exact same copy number profile.

Figure 3.16 shows a comparison of the expected and measured raw data (BAF and logR, top panels), total estimated copy number and the cancer cell fraction estimates of subclonal

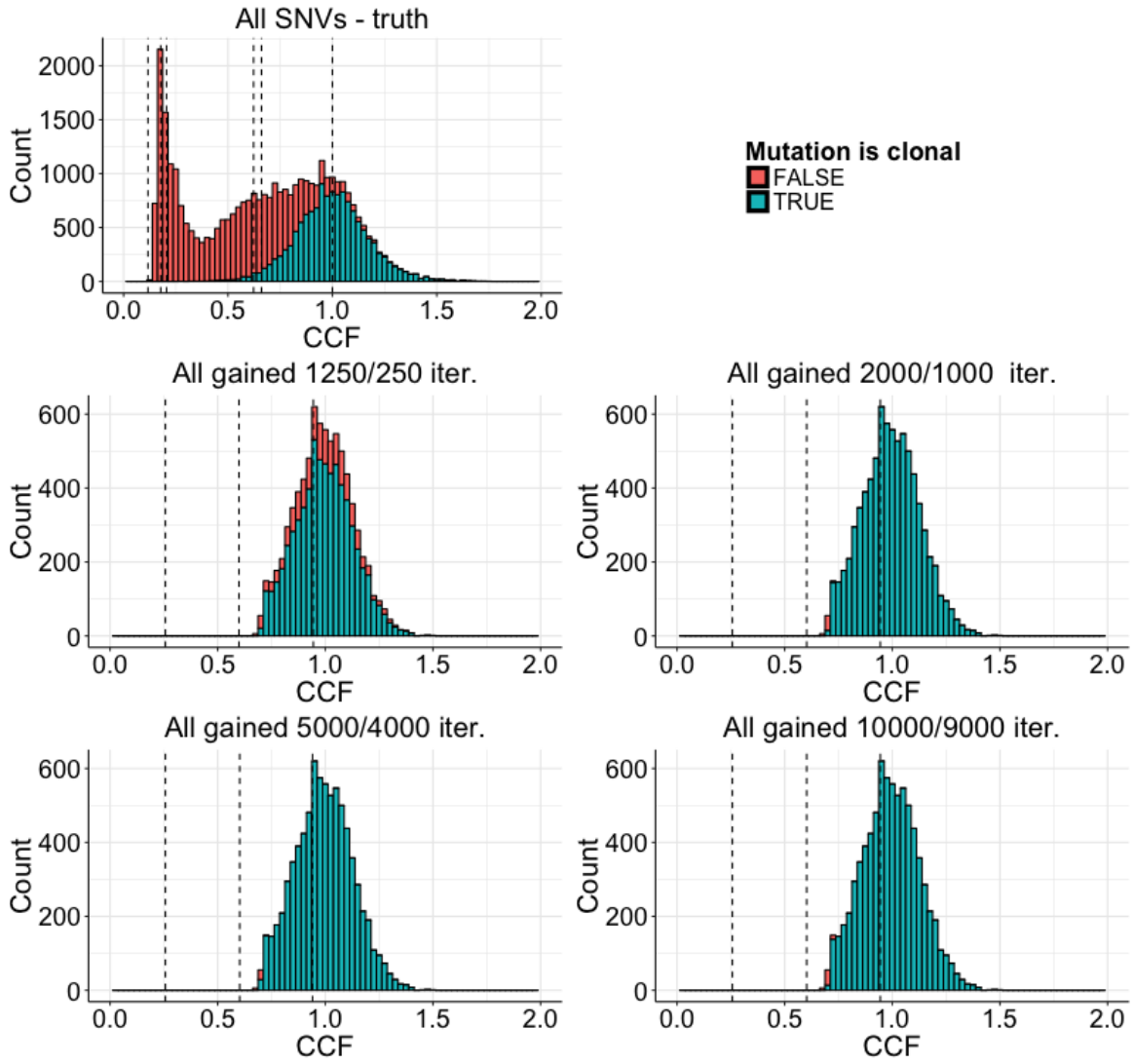


Fig. 3.13 Overview of the truth (top) and the output of multiple DPClust runs (middle and bottom rows) with varying numbers of MCMC iterations and burn-in. The top row shows the full true CCF space where bars are coloured to represent clonal and subclonal mutations, the black dashed lines represent the true cluster locations. The middle and bottom rows show all gained mutations that are clonal and offers a breakdown of these mutations into whether they are correctly assigned to the clone in green or incorrectly assigned to the subclone in red, while the dashed lines represent the found cluster locations. The figure shows that increasing the parameters to 2,000 iterations and 1,000 burn-in yields a reduction in the number of incorrectly assigned mutations.

copy number segments, where each dot represents a segment. Both BAF and logR show very high R^2 values, with only a few segments just of the diagonal. This means that the

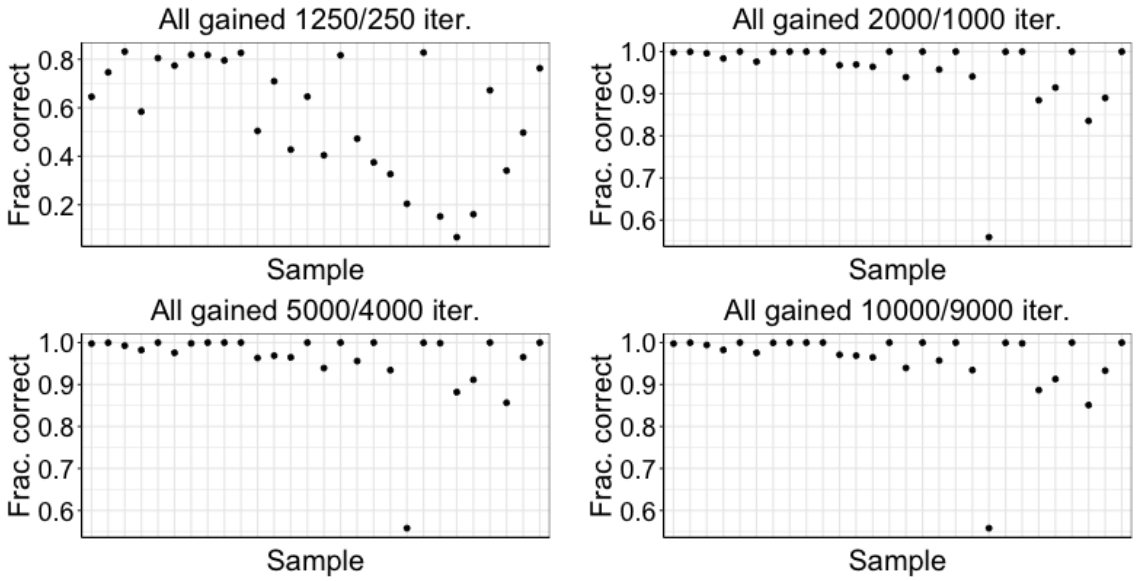


Fig. 3.14 The fraction of incorrectly assigned gained mutations across a number of selected samples showing poor performance. Three additional DPClust runs were performed beyond the original (1,250 iterations and 250 burn-in): 2,000 iterations and 1,000 burn-in, 5,000 iterations and 4,000 burn-in and 10,000 iterations with 9,000 burn-in. The results show that increasing the number of iterations yields considerable improvement, therefore the number of iterations should be increased.

Battenberg phasing and logR creation and correction steps are performing well and are able to adequately recreate the data that goes into copy number calling.

The total copy number estimates for these segments also show a very high correlation with the expected values (bottom left in Fig. 3.16), albeit slightly lower than the correlations obtained on the BAF and logR. One major source of discrepancy is caused by Battenberg calling clonal copy number, where subclonal copy number was expected. This occurs because Battenberg performs a t-test on the BAF and calls subclonal copy number only when the observed BAF is significantly different from the expected BAF of a clonal copy number state. A comparison of the CCFs (bottom right in Fig. 3.16) of the subclonal segments reveals that this affects a low proportion of all subclonal segments and most of these have low expected CCF values, which means the BAF is very similar to that of the closest clonal state.

A comparison of the CCF values reveals a strong correspondence between the observed and expected output (bottom right in Fig. 3.16). There is however a larger discrepancy than observed on the other three measures. In part this is caused by the aforementioned segments that are fit with clonal copy number. The data appears on a slightly discrepant diagonal, where the observed CCF is consistently higher than the expected. This effect is most likely explained by deviations in the purity estimate. The correlation between observed

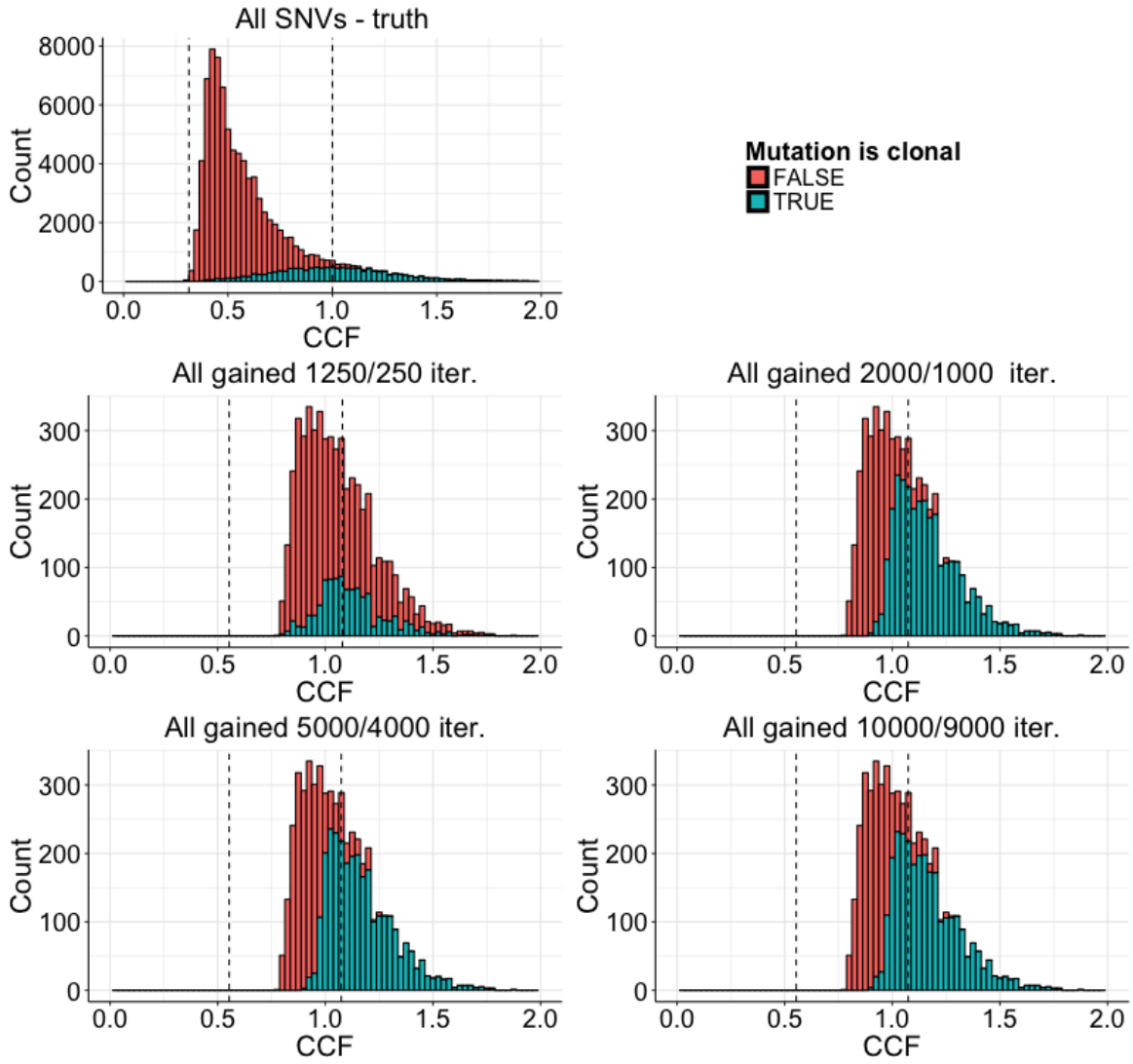


Fig. 3.15 Overview of a sample for which increasing the DPClust parameters had no effect on the number of incorrectly assigned mutations. This particular sample contains a very large and very broad subclone that contains the clone in its tail (top). In this scenario the mutations within the two clusters are split incorrectly, leading to a consistent number of gained mutations assigned to the subclone. A post-hoc step to reassign these mutations would resolve the issue, as there always is considerable uncertainty for mutations in between two subclones.

and expected CCF values is still strong however, but this result shows the CCF estimates of subclonal copy number are under the influence of some variation.

Finally, the bottom panels of Fig. 3.16 suggest that the Battenberg purity estimate may play a role in whether estimates of total copy number and CCF deviate from the diagonal. Fig. 3.17 contains the purity estimates for all samples and shows that Battenberg systematically

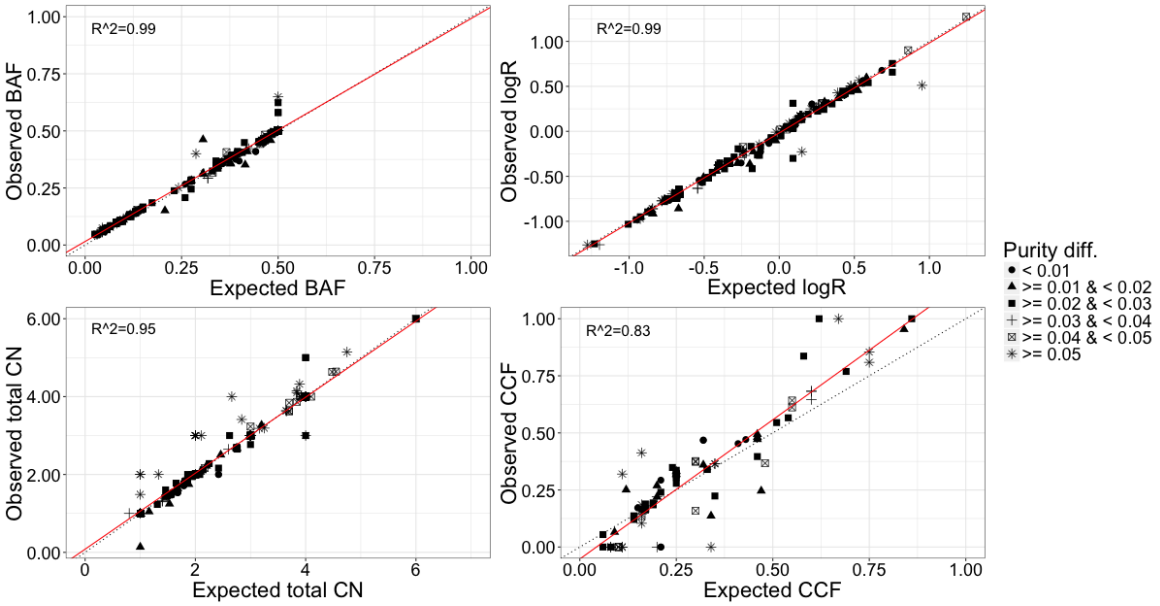


Fig. 3.16 The correlations between observed and expected BAF and logR per segment (each dot represents a segment) are very high, suggesting the Battenberg pipeline does well at obtaining the raw data required for copy number fitting (top panels). These correct raw values result in high correlations on the total copy number estimates per segment (bottom left) and strong correlations on the CCFs of subclonal copy number segments (bottom right). One source of discrepancy are cases where Battenberg calls clonal copy number, where subclonal was expected, due to the BAF not being significantly different from the closest clonal state. The slight bias in the bottom right panel is most likely explained by the deviations observed in purity estimates.

underestimates the purity (left panel). However, when the simulated BAF is replaced by the true BAF (while keeping the simulated logR), Battenberg calls the correct purity (right panel). This shows that the fitting algorithm works correctly and the small deviations of the BAF (the simulated BAF is on average 0.003 lower than it should be) are responsible for the purity discrepancy.

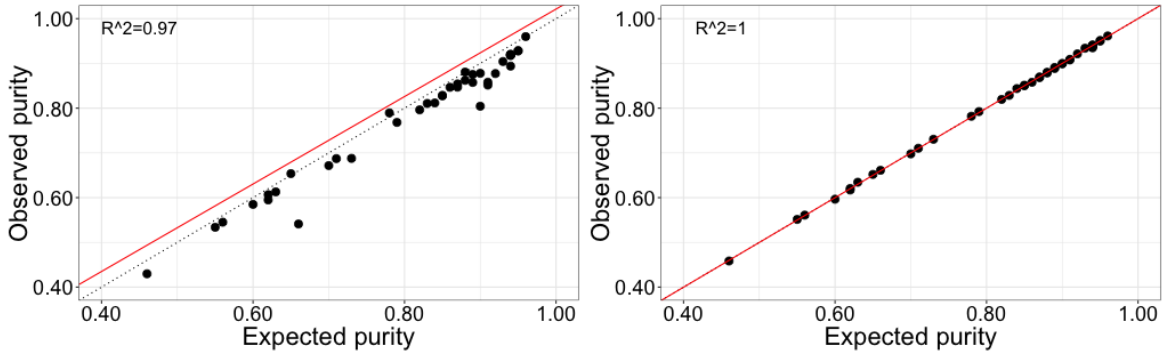


Fig. 3.17 A comparison of Battenberg purity calls (y-axis) from the simulated data with the true purity (x-axis) shows that Battenberg systematic underestimates (left). A run of the Battenberg fitting using the true BAF and simulated logR however shows that the fitting algorithm works as expected and yields the almost exact purity values (right). This means the very small deviations of the simulated BAF are responsible for the offset and shows how sensitive Battenberg is to correct BAF data.