Developing a standard method for the assessment of subclonal architecture reconstruction

Niels Burghoorn
TU Delft student 4659252
CCBC at ErasmucMC in Rotterdam

Supervised and reviewed by
Alberto Nakauma, Harmen van de Werken and Roeland van Ham

Januari - July 2020

Abstract

During cancer development, the tumor cell population usually emerges from a single cell ancestor and can therefore be categorized as clonal. As a result of the genomic instability and selection within this clonal population, additional mutations take place and allow for the differentiation of cell lineages. This tumor heterogeneity can be captured by doing clonality analysis which stratifies distinct tumor cell populations into groups, called subclones. The process of estimating the subclonal composition of a tumor requires intricate algorithmic approaches. Due to the growing popularity of clonality analysis, many tools have been made available for the reconstruction of subclonal architectures. The majority of the tools use next generation sequencing data to infer aspects of the subclonal composition and its dynamics. The vastness of available tools calls for an unbiased comparative method between them. Here a novel framework is presented to satisfy this requirement. By integrating a selection of available tools into this framework and testing their response to different simulated data, some of the tool qualities can be identified. The achieved results show this method is able to compare the ***PhyloWGS*** and ***DPClust*** tools using in silico generated tumor samples. Next to this, the framework is capable of analyzing real data. Samples taken from metastatic sites of 8 bladder cancer patients will be discussed.

Contents

1 Introduction

Cancer is a disease originated from mutations in the DNA of human cells. It usually starts with a specific (set of) beneficial mutations (driver mutations) that increase the fitness in a single cell. [1] [2] [3] [4] [5] [6] This cell starts dividing much faster and surviving better and longer than other cells, until a growth in the form of a tumor has developed. All the cells in the resulting tumor are descendants of the original single cell, which makes the tumor a clonal cell population. Furthermore, during tumor growth additional mutations occur due to the unchecked behaviour of the inner mechanisms of cancerous cells. These additional accumulating mutations may give particular cells advantages over others, while competing for space and resources within the tissue environment. Much like the original single cell, a new population may grow from the advantaged cells, contributing to the tumor population. This new population, of even faster dividing cells, is named a subclonal population. Investigating the nature of these dynamics is of interest for dissecting the complexity of cancer itself. Research in this area mainly strives to develop methods that allow for the identification of the major subclones in any tumor sample. This information could then be used to apply more appropriate therapy and to more fundamentally analyze cancer evolution.

Due to increased interest in the tumor's inherent intra-heterogeneity [7], tumor clonality analysis is a recurring subject in cancer studies. [8] This heterogeneity is ascribed to the evolutionary character of cancer cell populations, which are referred to as clones or subclones. This property of tumors has in the last 2 decades resulted into an additional field (”subclonal architecture reconstruction”) within cancer research, after notice was taken of some of the clonal characteristics of cancer. [1] [2] [3] The increase in popularity in this field is accompanied by the rise of next-generation sequencing (NGS), which complements the field well. Namely, NGS allows for accurate and relatively cheap extraction of the cancer's genome. NGS data is gathered in various ways to dissect the underlying mechanisms present in these subclonal architectures. These include: single cell analysis, bulk sample analysis or cross-sectional analyses. [9] The NGS data currently available primarily consist of (multiple) bulk samples from individual patient tumors. These bulk samples are however comprised of a mixture of subclonal tumor populations and healthy tissue, which complicates the identification of individual subclones. The need of separating this data and sorting the mixture of components within a sample has issued the development of subclonal architecture reconstruction methods (or subclonal reconstruction methods). Methods and tools are continually popping up, each one trying to capture some aspect of the clonality analysis in a different light or by different means. [10] [11] [12] [13] [14] [15] [16] [17] [18] [19] [20] [21] [22] [23] [24] [25] [26] [27] [28] [29] [29] [30] [31] [32] [33] [34]

Even though many of these tools prove useful in their respected papers, the abundance of produced tools calls for unbiased comparative analysis. This has also been noticed by the community [35] [36], although no method for comparison of a greater number of these tools is available yet. A possible solution for this problem is discussed in this report. The development of a comparative framework is described, where parameters and variables influencing subclonal dynamics in an *in silico* simulated tumor population can be compared for multiple tools. Also, the use of real data within this framework is illustrated.

The following report will first discuss the problems encountered by subclonal reconstruction methods. Then the main focus of the project is discussed, being the assessment of subclonal reconstruction tools by implementation of the novel framework. This is followed by a demonstration of the framework's features. Lastly, an analysis is done of a real data set, which is comprised of 8 whole genome sequenced patients with multiple biopsies over time that suffered from urothelial cancer. This patient data is investigated for the subclonal compositions of the metastatic sites. The patient analysis is reliant on the fact that the tumor has a clonal origin as discussed above. This can be reassured by previous research on the multifocality and possible clonal origin of urothelial cancers in particular. [4] [5] [6]

2 Subclonal reconstruction

The major subclones present in a tumor after diagnosis have attained a particular prevalence within the population. Much like Darwinian evolution the subclones compete with each other and with the healthy tissue for resources. Awareness of this phenomenon leads to the insight that the evolution of the subclonal composition throughout time - subclonal dynamics - is inherently linked to the tumor's trajectory. Therefore, knowing what subclones are present in a patient's tumor, and how they are going to evolve or adapt to therapy is essential for the improvement of clinical trials. Since the tumor is partly defined by the subclonal composition, knowing these composition features (e.g. fitness) and its dynamics (e.g. therapy response) might reveal information on the cancer's trajectory. More interestingly, this knowledge will provide us with a way to alter the cancer's trajectory by providing suitable treatment. A reconstruction of the subclonal composition, which is unknown at the time of diagnosis, is difficult to accurately reproduce using the contemporary information. Nonetheless, it is possible to do so, as will be described in this section.

By analyzing genetic material from tumor samples, inferences of possible subclonal compositions can be made, since the subclones are uniquely characterized by their DNA sequence. Although several small samples can be taken from the tumor, it is most often the case in the clinic that bulk samples are taken, containing multiple subclonal populations and including some healthy tissue (lowering the purity of the tumor sample). [7] [37] [9] This also means that a single sample is heterogeneous, a combination of subclones. Therefore, each sample should be "deconvoluted" (separating the individual subclones), since it is desired to find the subclonal phylogenetic relations, not the sample relations. [38]

The most recent common ancestor (MRCA) of the final cell population, a single tumor cell, must have expanded quickly during tumor formation in an event called clonal expansion. Similarly, some of the resulting cells in this population will later on have their own expansion events, subclonal expansions. One realistic assumption that can be made about both these expansion events, is that each one is caused by a single or a set of driver mutation(s), and is preceded by a set of passenger mutations. [7] [39] These passenger mutations of the initial clone primarily consist of some of the (non-malignant) mutations, which it has acquired during its lifetime before it picked up the driver mutation(s). Likewise, the passenger mutations found in the subclonal populations stem from somatic mutations. As a consequence of these different expansion events, the distribution of mutations describing the tumor will consist of 2 categories. The first is clonal mutations, which have been inherited by all descendants and therefore are present in the whole tumor cell population. And secondly, there are subclonal mutations consisting of all the mutations prior to a subclonal expansion, thus only spanning a fraction of the tumor cell population. This characteristic of the tumor allows for the deconvolution of the subclones through their mutational distribution. However, these distributions are distorted by the copy number changes, which are mutations that influence the count of genetic information (duplications or deletions of genetic fragments). These copy number changes have to be considered during subclonal reconstruction. Generally, this is how the sequencing data can be related to some of the subclonal features. Inferring of the subclones therefore starts with specific genetic information about the mutations in the tumor sample(s). This information usually consists of either the somatic nucleotide variants (SNVs), the copy number variants (CNVs) or both.

SNVs are acquired by both sequencing the tumor sample and a healthy reference sample. After sequencing, any difference in the genetic sequence between the two samples could indicate a mutation - a variant. To make sure that these variants are called truthfully, many reads (high depth/coverage) are obtained from a single variant locus solidifying call confidence. This kind of sequencing where high coverage data is collected (deep sequencing), will - as seen later - also prove useful in another way. It is expected that for some variants loci observed in the tumor sample, they do not cover all the reads (they are not found in the whole tumor sample), since the tumor consists of multiple subclonal populations, each with their own variant signature (group of variants). Utilizing this fact, the number variant calls can be compared to the number of reference calls at the variant loci. By taking the fraction of variant reads of this total, the variant allele frequency (VAF) is calculated as shown in Equation 3.

$$r_h : \quad \text{The number of reads supporting } h. \tag{1}$$

$$\text{VAF}_i : \quad \text{the variant allele frequency at locus } i. \tag{2}$$

$$\text{VAF}_i = \frac{r_{\text{variant at } i}}{r_{\text{reference at } i} + r_{\text{variant at } i}} \tag{3}$$

Doing this for every variant locus, the VAF spectrum is constructed, consisting of the frequency of mutation at each of these loci within the sample taken. When there are many variants with a particular frequency, it can be presumed that these variants are originating from a single subclone. It is expected that the VAFs of every subclonal population to cluster together, because the unifying variant signature consists of the mutations that occurred before its subclonal expansion. Due to sequencing noise and the discrete number of reads attained, the VAFs of the variants associated with a single subclone are spread out. This is illustrated in Figure 2. A clustering method is then used to assign these VAFs to groups. After estimating what the average frequency is for all the groups of variants, this forms an approximation of their subclonal prevalences. Generally, this is how these VAFs can be utilized to find the number of subclones and their prevalences in the tumor population.

Being well on the way of extracting subclonal prevalences, the CNVs complicate the picture. The CNV information consists of the copy number changes for genome fragments of the tumor sample relative to the reference sample. The CNVs also influence the number of variant calls, and therefore distort the VAF spectrum. These changes in copy number are regularly observed in studies investigating intra-cancer heterogeneity and relate to genomic instability of the subclones.[7] [40] As a standalone procedure, similarly to the case of SNVs, these CNVs can be related to a frequency spectrum and can then also be used to infer the subclone prevalences by the same principles.[30]

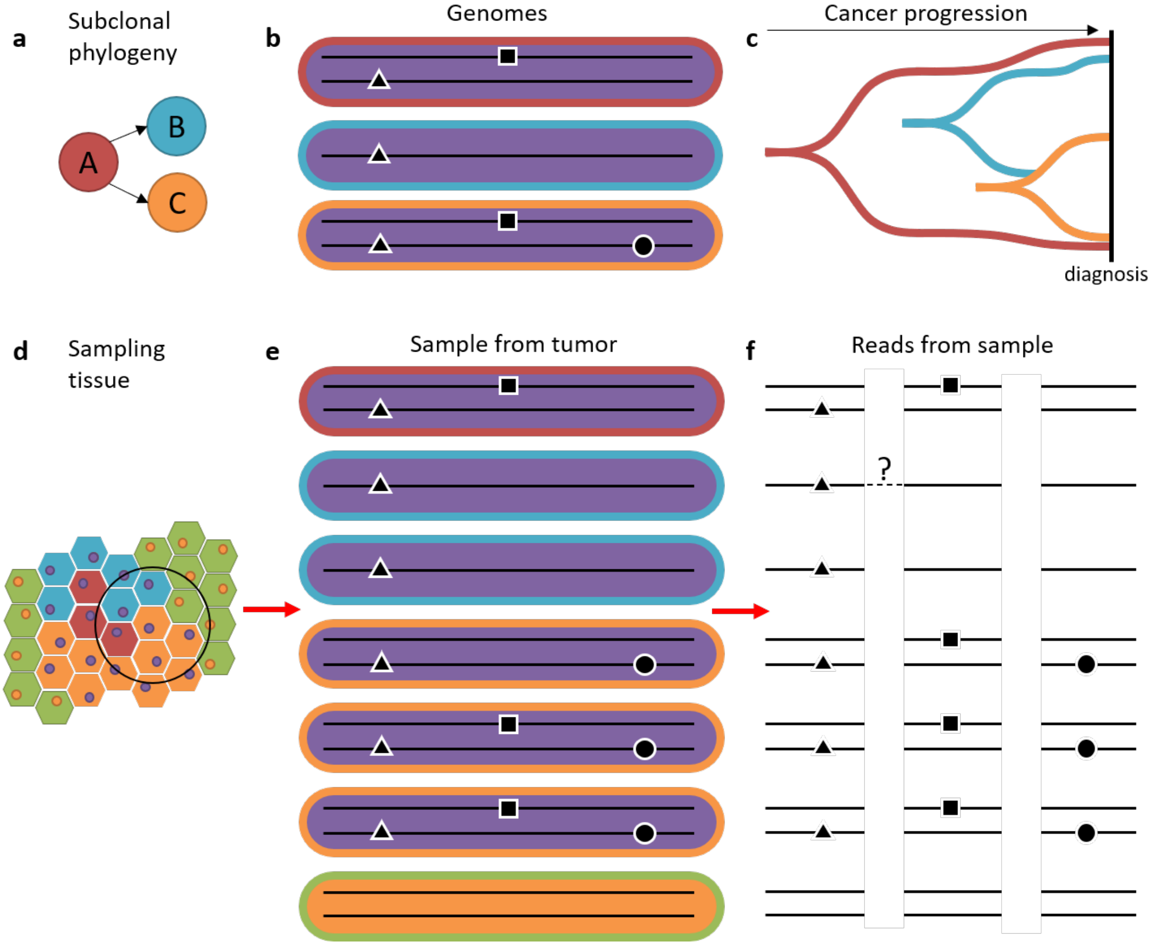


Figure 1: A minimal overview of the emergence of subclonal composition and tumor data collection by sequencing. **a**: The phylogenetic relationship of the subclonal composition, here A is the initial clonal population and B and C are it's descendant subclones. **b**: A schematic of the genomes of the three tumor populations. Subclone B has a CNV for this particular genome fragment. **c**: A fish-diagram of the prevalences of each tumor population. **d**: Schematic of tumor and surrounding healthy tissue. The black circle indicates the sample taken from the tumor. **e**: The cells within the sample taken with their respected genome. **f**: The acquired reads after sequencing of the genomes in the sample. Here no complete assembly of an individual cell genome can be made, since the inter-read relation is lost (indicated by the question mark). *Although it might appear obvious, it has to be kept in mind that this figure does not represent the full complexity of actual subclonal dynamics, it's sampling and the sequencing nuances. Besides, in realistic scenarios the order of magnitude of the displayed elements is orders of magnitude larger, and could be extrapolated for multiple samples and genome fragments.*

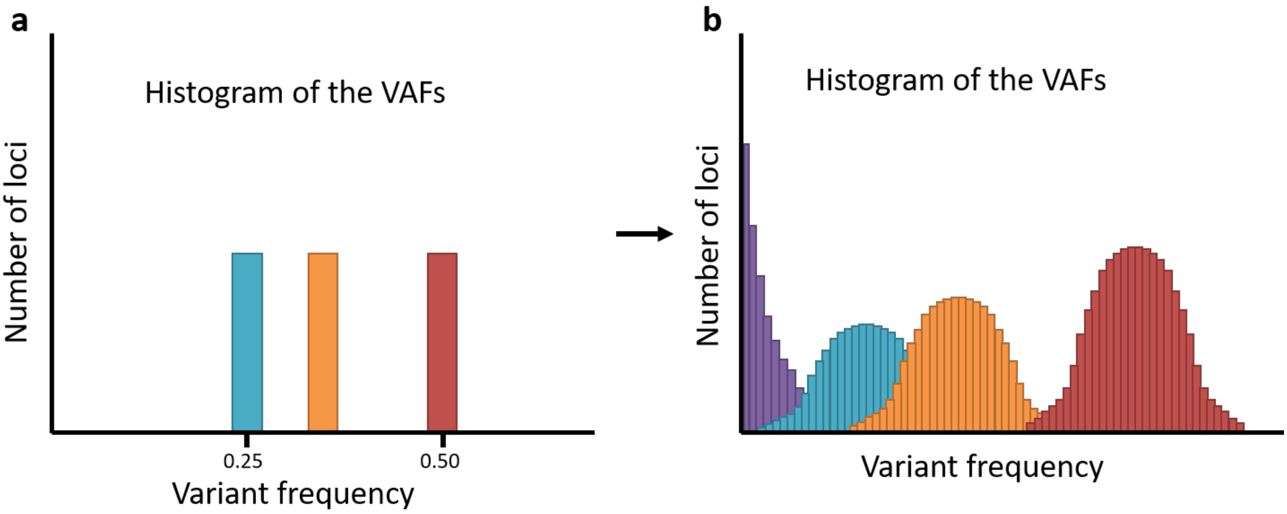


Figure 2: An ideal representation of the VAFs corresponding to the scenario posed in Figure 1. **a**: The idealized peaks expected for the VAFs matching to the subclonal populations present in the sample. No sequencing noise or mutational noise. **b**: The VAF distributions of many mutations corresponding to the subclonal populations in the sample. The mutational noise is shown as the purple tail in the lower part of the spectrum. The sequencing noise (broadness of the peaks) is due to the discrete measurements of variant reads, DNA polymerase infidelity among other vulnerabilities in deep sequencing.

In Figure 1, a minimal overview of the problem caused by CNVs is portrayed. Within panel **e** and **f** notice the purity issues generated by the inclusion of healthy tissue. Additionally, the need of relating the variants to the VAFs becomes more apparent, since it is shown how the inter-variant relationships are lost through the process of sequencing. The only way of telling the subclones apart at this stage is by relating the variant loci (with their VAFs) to a subclone. Here the challenge of subclonal reconstruction becomes apparent: trying to infer the information of panel **a**,**b** and **c** from the data in panel **f**. For more reliant estimations, it is important that the VAFs accurately represent the sample's VAFs, and it is here where the high read depth of deep sequencing proves useful once more. Next to this, the sample should be representative of the tumor population in order for any later subclonal architecture estimation to bear relevance over the whole tumor (panel **d**). The CNV distortion of the VAF spectrum can also be seen in panel **e** and **f** in the population B (blue), since its loss of the genomic fragment now would contribute to the VAF of 1 instead of the VAF of 0.5 in its parental population A (red).

A common strategy for subclonal reconstruction is to extract the cancer cell fractions (CCFs) from the information of the SNVs and the CNVs, while also correcting for sample purity. A CCF is defined as the fraction of cancer cells that carry a (set of) variant(s). The earlier mentioned mutational distribution components, which are clonal mutations and subclonal mutations are then assigned part of the CCF spectrum, these are $CCFs = 1$ (present in all cells) and $CCF < 1$ (fraction of cells), respectively. This definition therefore also allows a CCF to represent a single subclone, since subclones in turn are described by a set of variants - a variant signature. These CCF values can be calculated for each SNV after correcting for the CNV and purity information. Acquisition of CCFs is rigorously explained in "*Principles of Reconstructing the Subclonal Architecture of Cancers*".[37]

Within the same paper an introduction of a clustering method is given, known as the Dirichlet Process (DP), which can be used to assign the CCFs to a subclone. This is a Bayesian non-parametric method, used to construct a mixture model (MM) of the subclonal composition. A DP itself can be sampled with various algorithms (each with its own analogy), for instance through the Chinese Restaurant Process (CRP), the Stick-breaking (SB) or imagining a colored Pòlya Urn.[37] [41] [42] The basic idea is that new clusters are assigned to data points on the fly, meaning that new clusters (in this case subclonal populations) are added when the data (CCFs) requires so.

To give a slight overview of the workings of the DP sampling methods, two of the most popular algorithms (and analogies) are shown in Equation 4-11. Sampling some of the subclonal composition parameters such as: number of subclones, mutational load and the subclonal prevalences/fractions (CCFs) can be done using these algorithms. The likelihood that this prior distribution represents the given data can then be calculated. This process can then be implemented with Markov Chain Monte Carlo (MCMC) algorithms (such as Gibbs sampling) for aquiring the posterior distribution, and is (for some tools) fundamental for estimating the subclonal architecture.

$$\text{Dirichlet process}: \; DP(\alpha, G) \tag{4}$$

$$\begin{aligned} &\textbf{Stick breaking} \\ V_i &\sim Beta(1, \alpha) \\ C_k &\sim V_k \prod_{i=0}^{k-1} (1 - V_k) \end{aligned} \tag{5}$$

$$\textbf{Chinese restaurant process}$$

$$\begin{aligned}
c_i^k :& \quad \text{The event that customer } i \text{ joins table } k. && (6)\\
\tilde{\pi}_{[n]} :& \quad \text{The set of partitions (customers assigned to tables).} && (7)\\
n :& \quad \text{The number of customers assigned.} && (8)\\
K :& \quad \text{The number of tables assigned.} && (9)\\
P(c_n^k | \tilde{\pi}_{[n]}) =& \quad \frac{\text{number of customers at table } k}{\alpha + n} && (10)\\
P(c_n^{K+1} | \tilde{\pi}_{[n]}) =& \quad \frac{\alpha}{\alpha + n} && (11)
\end{aligned}$$

Clustering can be done by implementation of a Collapsed Gibbs sampler and extending it for non-parametric applications.[43] A Gibbs sampler is a statistical inference algorithm using random samples generated similar to other MCMC methods. The Gibbs sampler can be used to make a MM of the CCF spectrum, therefore deconvoluting the subclonal variants. For this it uses the conditional probability distributions (CPDs) to approximate a multivariate probability distribution (MPD). This MPD contains the information about the subclonal composition, and will be parameterized by the cluster means $\boldsymbol{\mu} = (\mu_1, ..., \mu_i, ..., \mu_K)$, the cluster weights $\boldsymbol{\pi} = (\pi_1, ..., \pi_i, ..., \pi_K)$ and the variant-cluster assignments $\boldsymbol{z} = (z_1, ..., z_i, ..., z_N)$ where N represents the total number of variants and K is the total number of clusters. The $\boldsymbol{\mu}$ then will form an approximation of the subclonal fractions. Similarly the $\boldsymbol{\pi}$ will relate to the number of mutations of the respected subclone, where the specific assignment of the variants is captured in $\boldsymbol{z}$ (here every $z_i \in \{1, ..., K\}$). By doing many of these Gibbs sample iterations, the outcome will approximate a MPD with parameters close to the real subclonal composition parameters. The main reason that allows the use of CPDs to approximate the MPD is the conditional relation between them, which is shown in Equation 12. An overview of the Gibbs sampling process is shown in algorithm 1.

$$\begin{aligned}
&\text{All parameters of interest } \mu_k, \pi_k \forall\ k \in \{1,\ \dots, K\} \text{ and } z_i \forall\ i \in \{1,\ \dots, N\} \text{ are captured in } \boldsymbol{\gamma} \\
&\boldsymbol{\gamma} = (\gamma_1,\ \dots, \gamma_D) \quad , with\ \ \gamma_{\neg\ i} = (\gamma_1,\ \dots, \gamma_{i-1}, \gamma_{i+1},\ \dots, , \gamma_D) \\
&P(\gamma_i | \gamma_{\neg\ i}) = \frac{P(\gamma_1,\ \dots, \gamma_D)}{P(\gamma_{\neg\ i})} \propto P(\gamma_1,\ \dots, \gamma_D)
\end{aligned} \tag{12}$$

Algorithm 1: Simplified overview of Gibbs sampling algorithm for a MM

Input : Data points $\boldsymbol{x} = \{x_1, ..., x_N\}$, number of clusters K, number of samples *sample_number*

Output: Data point assignments $\boldsymbol{z} = \{z_1, ..., z_N\}$, cluster means $\boldsymbol{\mu} = \{\mu_1, ..., \mu_K\}$, cluster weights $\boldsymbol{\pi} = \{\pi_1, ..., \pi_K\}$

```
$\boldsymbol{z}$ = randomAssignments($\boldsymbol{x}, K$)
$\boldsymbol{\mu}$ = randomMeans($\boldsymbol{\mu}, K$)
$\boldsymbol{\pi}$ = randomWeights($\boldsymbol{\pi}, K$)
for sample in 1:sample_number do
    $\boldsymbol{z}$ = updateAssignments($\boldsymbol{x}, K$)
    $\boldsymbol{\mu}$ = updateMeans($\boldsymbol{\mu}, K$)
    $\boldsymbol{\pi}$ = updateWeights($\boldsymbol{\pi}, K$)
end
```

Algorithm 2: Simplified overview of Collapsed Gibbs sampling algorithm for a DPMM

Input : Data points $\boldsymbol{x} = \{x_1, ..., x_N\}$, initial number of clusters K, number of samples *sample_number*, dispersion parameter α

Output: Data point assignments $\boldsymbol{z} = \{z_1, ..., z_N\}$, cluster means $\boldsymbol{\mu} = \{\mu_1, ..., \mu_{K^*}\}$, cluster weights $\boldsymbol{\pi} = \{\pi_1, ..., \pi_{K^*}\}$, here K^* is determined inside the algorithm

```
$\boldsymbol{z}$ = randomAssignments($\boldsymbol{x}, K$)
$\boldsymbol{\mu}$ = randomMeans($\boldsymbol{\mu}, K$)
$\boldsymbol{\pi}$ = randomWeights($\boldsymbol{\pi}, K$)
for sample in 1:sample_number do
    for $x_i$ in $\boldsymbol{x}$ do
        removePointFromSpectrum($x_i$)
        $z_i$ = getAssignmentFromPredictiveProbability($x_i$, currentSpectrum, $\alpha$)
        addPointToSpectrumInCluster($x_i, z_i$)
    end
end
```

In algorithm 1, it can be seen that the parameters of interest are consecutively updated, finally resulting in a closer representation of the actual parameters. However, the problem remains that the number of clusters, and thus the number of subclones, needs to be known beforehand. This can be solved by making the algorithm generate clusters in a data driven fashion. This is known as non-parametrics, but before this can be implemented, the Gibbs sampler should be transformed into a Collapsed Gibbs sampler. This can be done by integrating out the $\boldsymbol{\mu}$ and $\boldsymbol{\pi}$ parameters. Afterwards, sampling the z_i values from a predictive probability distribution of the existing clusters. After all the variants are assigned, the $\boldsymbol{\mu}$ and $\boldsymbol{\pi}$ values can be retrieved from the clustered values. This method now allows for a flexible number of clusters in the algorithm's outcome. This addition is based on the simple incorporation of a chance that the variant does not belong to any of the existing clusters, but in fact to a new cluster. In practise this chance is defined by the dispersion parameter α.[44] An outline of the DP clustering algorithm derived from the Collapsed Gibbs sampler is shown in algorithm 2.

After the assignment of CCFs to a subclonal cluster, the phylogenetic relation between the subclonal populations can be inferred. This can be achieved by qualitative observation of the specific mutations present in subclones (new sequencing information would be required). However, also probabilistic methods are present, which are based on the "pigeon-hole" principle. This principle uses the fact that the sum of descendant subclonal fractions present must be smaller or equal to the parental subclone fraction. This is true since the subclonal fractions are represented by the variants, which are assumed to be unique as discussed in subsubsection 4.1.2.

3 Project focus

Many tools have been developed to infer the possible subclonal compositions of a tumor using the discussed NGS data. All of these tools have their own advantages and weaknesses. However, they have not been thoroughly compared to each other. This can be attributed to the lack of a golden standard data set and the differences in input data required for each tool. This section will discuss the main aim of the project and stress the importance of comparing these tools.

Knowing the strengths and pitfalls of these tools on a more quantitative level, might reveal useful insights on the dependable and/or flawed aspects/methods employed by these tools. It might also provide us with a more nuanced view of the applicability of certain tools in particular situations. Certain tools might for instance turn out to be more suitable when many variant loci are recorded (WGS), and others in the situation where less variant data points are available (targeted sequencing).

By building a comparative framework in which the tools are fully integrated and can be tested for various aspects, an attempt can be made to solve this problem. The framework will be supported with a simulation capable of simulating data sets for various realistic subclonal trajectories.

3.1 The abundance of tools

A multitude of groups are developing tools and methods that utilize the NGS information to reconstruct aspects of the subclonal architecture. Many have claimed the added significance of their tool and have done comparatative analysis with other popular tools such as *PhyloWGS*[10],*DPClust*,*SciClone*[12] and *PyClone*[14]. Even though many of these tools are indeed of importance to the field, some could be labelled as duplicate work or simply do not outperform the state of the art implementations. The time and effort that has gone into developing tools for subclonal reconstruction has now called for a meta-comparison of the tools themselves. This has also been noticed by the community and therefore effort was put into creating some kind of comparative framework.[35] The next paragraph will briefly discuss their work, since it directly relates to the development of the novel framework to be presented.

In order to better evaluate the analysis of subclonal reconstruction, the problem was split into "subchallenges". Each subchallenge marks a certain step within the process of subclonal reconstruction and is coupled to some feature of a subclonal composition. The subchallenges are as follows:

- SC-1A : estimation of tumor purity
- SC-1B : estimation of number of subclones
- SC-1C : estimation of variant prevalence in the tumor population
- SC-2A : assignment of variant to subclone
- SC-2B : coclustering of mutations (assignment probability of pair of variants to subclone)
- SC-3A : assignment of subclone to phylogentic tree location
- SC-3B : coassignment of subclones (relationships between subclones)

Each group of subchallenges - 1,2 and 3 - asks their own fundamental question to the subclonal architecture, which are "What is the tumor composition?", "What are the mutational characteristics of the subclones?", and "What are the phylogenetic relationships between the subclones?", respectively. This division of the problem was then supported with appropriate evaluation methods while using simulated data, since a golden standard data set is not yet available. The simulation procedure is rather involved and will not be elaborated on. It will however be noted that - given the provided online tools such as the constructed wrapper for BAMSurgeon[45] - this simulation is rather complicated to make use of, mostly due to limited documentation. This intricacy of the created simulation with its resulting framework, together with the need of meta-comparison of the many subclonal reconstruction tools triggered the need for a different framework capable of doing the meta-comparison.

3.2 Novel framework

Unlike the previous framework, the main focus of the novel framework will be on the subclonal reconstruction tools and not on the variant detection methods. This will allow for development of a more straightforward simulation inspired on the actual subclonal dynamics and not on the biological complexity of mutagenesis. In fact, this complexity is usually corrected for by doing CCF calculation and can be omitted by simulating these values more directly. Another advantage of simulating the CCFs is that these allow for fundamentally identical tool input data, and thus provide a way of solely comparing tool performances. The second main difference with the community framework adds to this. Namely, the focus will lay on the application of deep sequenced NGS data, not on the low read depth data reviewed earlier. Lastly, a smaller selection of the subchallenges will be tested for, which are SC-1B,SC-1C and SC-2A.

The differences permit the simulation to be only that of the tumor population, since SC-1A is not tested. Also, the subclonal trajectory can be made very specific to fit particular requested behaviour which could be linked to a certain cancer type. Even though clearly a smaller fraction of the subclonal reconstruction process is explored, it is - in this aspect - a more general approach, since it is applicable to a greater number of tumors. Next to this, different behaviours can be simulated within a particular cancer type, as will be explained later.

What is aimed to achieve with this new framework implementation, is a new simplistic way of testing tools in response to the different subclonal dynamics, with the focus on their most essential estimation features. The features deemed most important were those that answer the three above mentioned subchallenges being: number of subclones, variant prevalences and the variant-subclone assignment. The latter two are for practical reasons combined into the subclonal prevalences or equivalently the subclonal fractions (CCFs). Additionally, run times were recorded, which is mandatory for an inter-tool comparison. The reason for this, contains the fact that it is difficult to keep run times constant. Therefore, these mark the only difference in assigned resource used by the tools. This results in three recorded qualities per estimation: subclone number, subclone prevalences (subclonal fractions as CCFs) and run time.

4 Framework

The framework presented consists of 3 main components:

- Data generation
- Data acquisition
- Tool analysis

The workings of all of these components will be discussed in this section. This will give an overview of the inner workings of the framework. It is primarily programmed using *Python*, but also contains *R* and some *Bash* code. Although the tools are integrated into the framework, the discussion of their inner workings is kept to a minimum in this section. The goal of the framework is to get a more general overview of the tool responses to particular tumor architectures.

To give an indication of the method used here, consider algorithm 3. It shows the global outline of the central script used to do the exploration of tool response within the framework. To use the framework a manual is provided in Supplementary subsection 7.7.

Algorithm 3: Simplified overview of framework

```
simplified ui.py($\mathcal{T}, \tilde{\theta}$)
Input  : Tools to run $\mathcal{T}$ and parameter space $\tilde{\theta}$ specified by user
Output: Estimations of tool $\mathcal{T}$ for parameter space $\tilde{\theta}$
data = []
for $\theta_i$ in $\tilde{\theta}$ do
    reads = simulateData($\tilde{\theta}$)
    $datapoints_i$ = runTools($\mathcal{T}$, reads)
    data.append($datapoints_i$)
    writeReports(data)
end
writeData(data)
```

4.1 Data generation

The simulation of the data is an essential part of the framework, since it allows for testing the tool response for many different possible subclonal compositions and their trajectories. The simulation should also keep record of the 'true values', used as reference to calculate error estimates for the tools tested. This, while keeping in mind that the data generation will happen within a loop - where the testing the of tools takes place - and therefore is in need of high throughput. The simulation must however not be reduced to the point where an unrealistic representation of the mechanisms involved in subclonal evolution is achieved. Because of these requirements, it was chosen to simulate a population of cells as mutation signatures represented by bit strings.

In order to test the tools for multiple different subclonal trajectory features, a versatile data simulation program was constructed. The simulation will allow for the representation of realistic scenarios, although not too complex in order to reduce computation time. Next to the fact that only the tumor population needs to be simulated, it was chosen to limit to SNVs and disregard CNVs. This eliminates the need for correcting the VAFs for CNVs during the calculation of the CCFs. However, this does put a limit on the tool assessment capabilities. Concretely, the data output of the simulation would only consist of a sequence of mutation presence or absence (binary 1 or 0). Every bit string then represents the mutational signature of a single cell. Consequently taking the mean of all the bit values at a single locus (bit position) over the simulated population, represents the CCF of this locus for the complete tumor sample. Simple calculation of the CCFs is necessary, because a lot of data needs to be converted into this format, as they serve as the elementary values used for determination of many tool inputs.

First, the user specifications of the simulation parameters are used for the initialization of the tumor population to be simulated. Then the new generations of the population are simulated consecutively. In order to mimic the temporal relationship between subclones, the simulation models a time-stepping cell population, which will facilitate the imitation of the evolutionary dynamics of a tumor. The population size is chosen constant, and therefore the cells should be seen as representatives of a complete tumor population. This can be justified by realizing that an actual sequenced tumor samples are also only representatives of the entire population. The number of loci that allow for mutation within each cell is also constant, and could for instance be viewed as the total number of mutable base pairs within the genetic data sequenced (or for example a set of genes in a targeted sequencing panel). During simulation of every generation of cells, the mutations occur randomly (according to user specifications) and are recorded. The population simulation is a Wright–Fisher (WF) model: each cell in the previous generation produces two descendants and this new set of cells is than selected randomly to be a member of the next generation (although some selection characteristics related to the mutations present can be specified by the user). Occasionally a subclonal expansion event takes place. This event subsists of a single cell in the previous generation acquiring a bigger set of mutations and many more daughter cells. Here the additional mutations are included in order to collapse the time needed for passenger mutations to accumulate into a single event. These clonal expansion events are recorded and the cells are tagged such that the fraction of the descendants from this single cell are addressable. The last element that is also recorded is the parental relationships of all cells. The recording of all these population features is essential, since it will serve as a reference - ground truth - to compare against the estimations done by the tools.

Algorithm 4: Simulation and data generation

```
simplified main.py($\theta$)
Input  : Simulation parameter set $\theta$, specified by user
Output: Tumor population samples and population records
$population_0$ = initializePopulation($\theta$)
populations = []
for $t$ in 1:final_time do
    $population_t$,$record_t$ = nextGeneration($\theta$,$population_{t-1}$)
    populations.append($population_t$)
    records.append($record_t$)
end
populations = shuffleWithinGeneration(populations)
writeSamples(populations,$\theta$)
writeRecords(records,$\theta$)
```

When the simulation of the tumor population is complete, the ordering of the cells within a generation is randomized and samples are taken (as specified by the user). These samples are written to *txt-files* as bit strings and will do duty as the only piece of information that is used by the tools. The recorded ground truth of the population dynamics is also written to *txt-files* in convenient formats. Optionally, a log of the total simulation will be outputted. The details captured in the simulation log can be specified as well - ranging from a broad overview of the subclonal expansion events, to a visual representation of every cell in every population mutation with complete phylogenetic tree.

Accompanying the simulation scripts are some scripts that can be used to summarize, visualize and write a *PDF-report* of the output data (among which is a fishplot[46] of the tumor architecture). More information about these specifics in found in Supplementary subsection 7.7.

4.1.1 Simulation parameters

The simulation parameters should be representative of real-world tumor compositions, and so should their dynamics. Since a tumor's subclonal dynamics depend very much on the cancer type and also vary from patient to patient, many simulation parameters can be fully defined by the user to simulate the desired behaviour. In order to account for some of the biological stochasticity, part of the simulation workings are directed by random events, which to some degree can be specified. These events, include the mutation events and subclonal expansion events, which are undetermined for the locus index and cell index respectively. Most of the simulation's parameters are considered in the following section, supported by graphical representations in the form of a fishplot of their population dynamics.

Before considering the individual parameters, a remark should be made on a subset of the parameters. These include the:

- number of subclones
- subclone expansion size
- subclone phylogenetic relations
- selective advantages
- subclone emergence timings

This subset contains the highly impacting features for the resulting subclonal composition. Therefore testing any individual setting within this subset would require testing for all relevant value combinations of the others within this subset. This range of combinations vastly exceeds the computer resources available, and are therefore most of the time not considered as individual classifications for which performance is tested. Knowing that these parameters possibly play a significant role in the estimation performance of any tool, they are mostly kept to some default or tested for all permutations when possible (e.g.: there are only 6 possible phylogenies for a composition with 3 subclones). This subset is, on the other hand, easily graphically displayed using fishplots as done below.

The first obvious specification is the number of subclones to be simulated. Looking at some of the simulation outputs in Figure 3, it can be noticed how specification of a different number of subclones influences the resulting tumor composition.

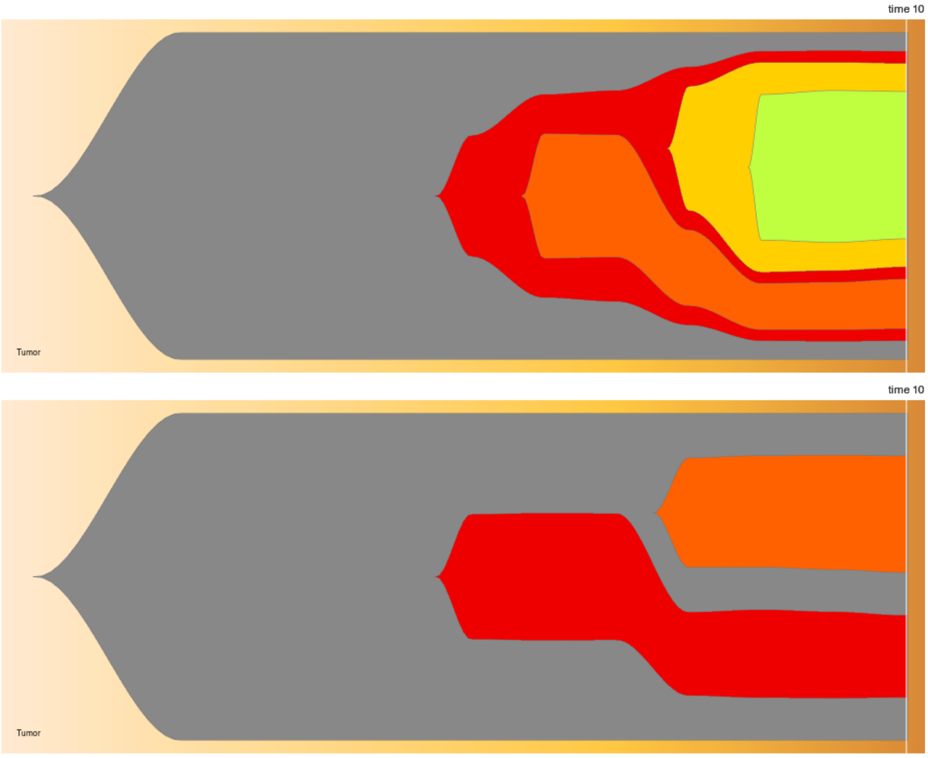


Figure 3: Fishplots showing the subclonal composition over time within a tumor population. The upper panel shows the result when 4 subclones where specified, the lower panel is the result when 2 subclones where specified.

Secondly, the expansion size of subclones can be fully determined, or be randomly assigned from a normal distribution given a user-specified mean and standard deviation. It affects the growth of the subclone during its expansion event. The effects of varying this parameter are shown in Figure 4.

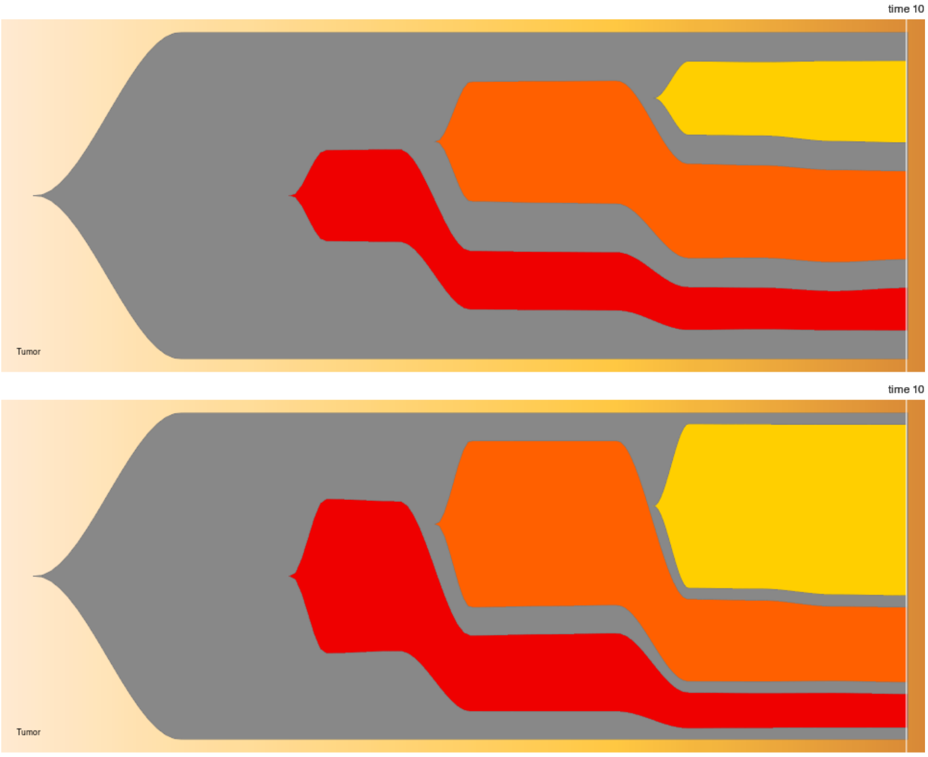


Figure 4: Fishplots showing the subclonal composition over time within a tumor population. The upper panel shows the result when the expansion rate of a subclone was indicated to be around 0.3, the lower panel is the result when the expansion rate of a subclone was indicated to be around 0.5.

Next up, the phylogenetic relation of the subclones can also be fully indicated or left up to chance. Fully linear phylogenies can be specified or any other phylogenetic tree, when indicated. This feature is shown in Figure 5.

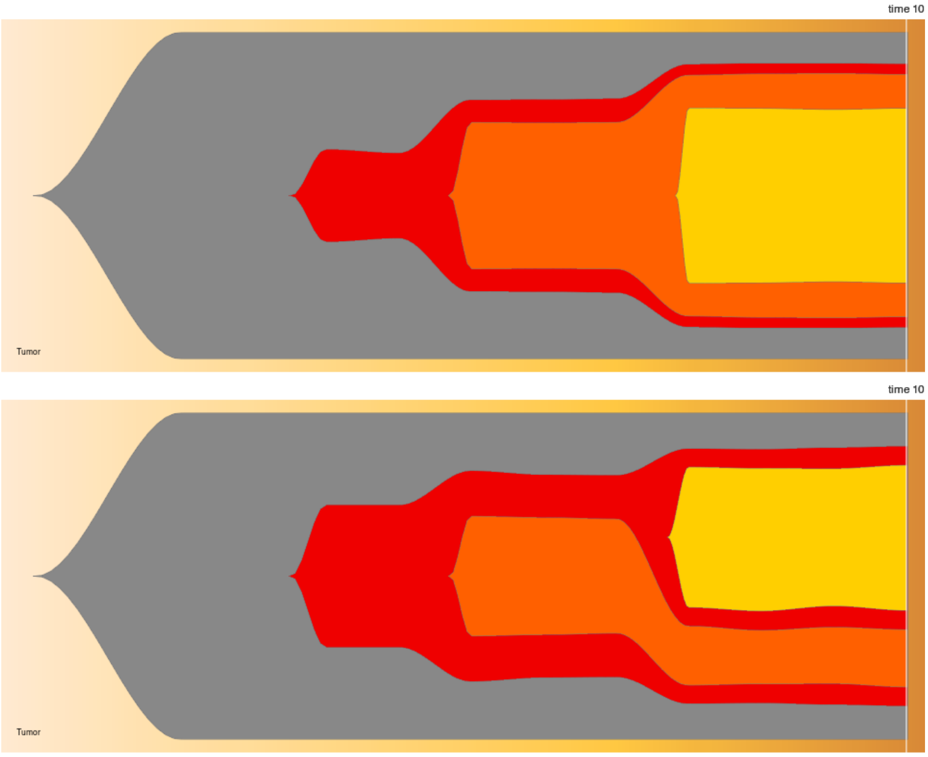


Figure 5: Fishplots showing the subclonal composition over time within a tumor population. The upper panel shows a completely linear phylogeny, the lower shows the result for a particular phylogeny specified.

The fourth characteristic alterable is the time of emergence of every subclone. For practical reasons, this is limited to 1 subclone per time point. However, if close succession of subclones is desired, then simply increasing the time scale of the whole simulation will suffice. Two simulations are shown in Figure 6, that illustrate this setting.

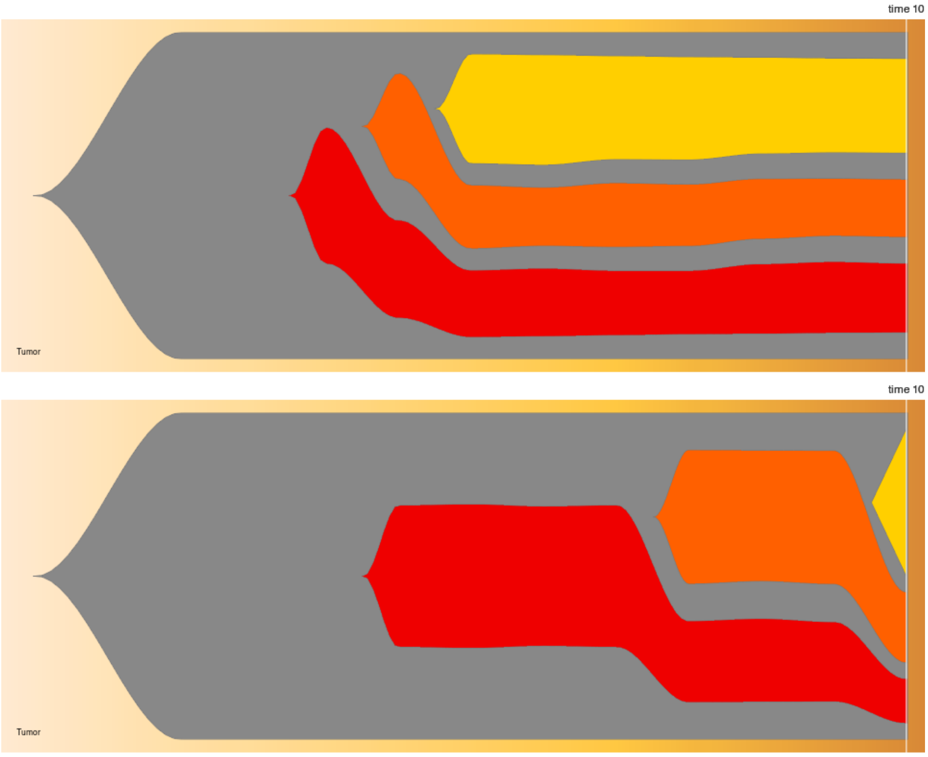


Figure 6: Fishplots showing the subclonal composition over time within a tumor population. The upper panel shows a result where the subclones where chosen to emerge at times 2,3 and 4, the lower panel is the result when the subclones where chosen to emerge at times 3,6 and 9.

The last parameter in this subset, representing the strength of negative or positive selection of subclones, comes in two forms. Firstly, a reward/penalty for any mutation can be given (this also covers random mutations, which is a later discussed setting). Secondly, the subclones can be specified to get a (numer of) driver mutation(s), which consists a specific number of loci with different selective rewards (or penalties). These rewards and penalties are taken into account when the cells in the next generation are selected from the descendants of the previous generation (see algorithm 4, being a WF model). Two different behaviours are shown in Figure 7.

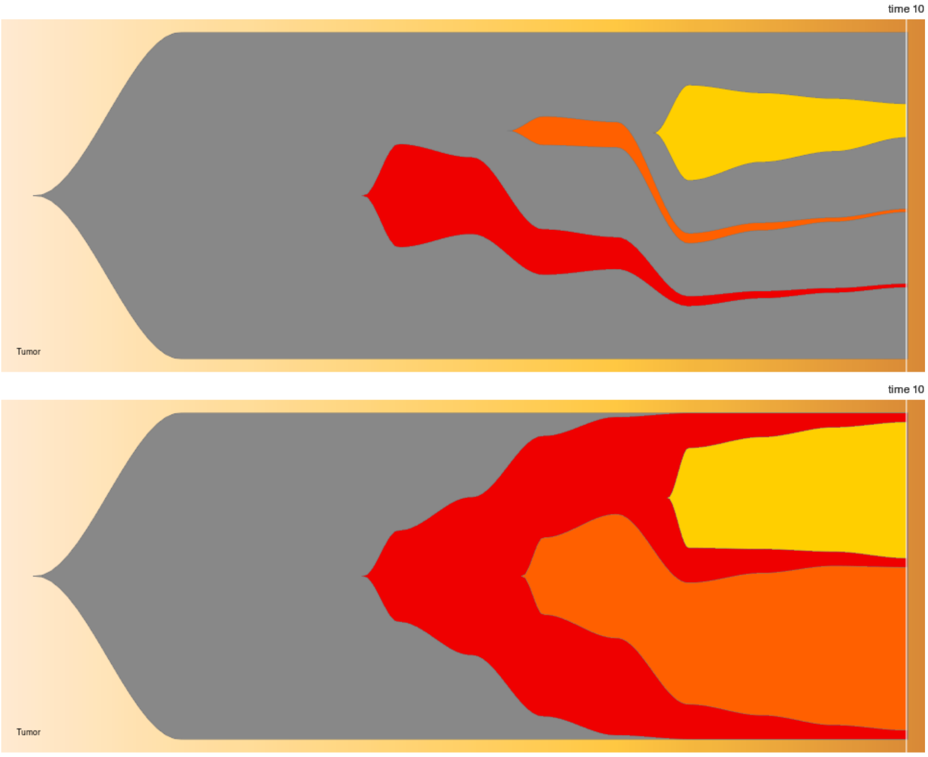


Figure 7: Fishplots showing the subclonal composition over time within a tumor population. The upper panel shows an example where the subclones penalized for their driver mutations (negative selection), the lower panel is the result when each subclone was awarded a selection advantage upon attaining their driver mutation (positive selection).

The above subset of parameters, although being fully determinable, can also be left unchecked. This results in simulating random evolutionary behaviour. In this case, driver mutations mark the emergence of a subclone. By changing the number loci being classified as driver mutations and nudging the mutation parameters, some interesting tumor populations can be generated.

Next to the 5 features discussed above, others which are concerned with the sampling and sequencing of a tumor or with the simulation implementation are also adjustable. These include the:

- initially mutated fraction of clonal population
- random mutation rate of each cell
- mutation load of a subclone
- population size
- number of loci within a cell (mutable genome size)
- sequencing noise
- tumor sampling times

These parameters play an important role in the simulation, even though they do not have the same degree of influence on the subclonal composition that is generated. Their role is mainly related to the CCFs spectrum data that is eventually achieved. These parameters can be tested more independently from each other, as will be discussed later.

The first parameter - the initially mutated fraction - is related to the mutational signature of the single cell ancestor of the tumor. This setting together with the 'number of loci' setting determines the number of mutations within the initial clonal population. An important trait of the tumor, since it establishes the assurance of detection of the parental clonal population.

The second parameter - the random mutation rate or "mutation noise" - bears relevance in the recreation of realistic spectra. This parameter dictates shape of the low frequency tail which is frequently seen in VAF spectra[15] (see Figure 2), as well as the uncontrolled mutational behaviour of cancerous cells. Although the former is primarily thought to be a characteristic of data processing artefacts and the latter a biological phenomenon, these both result in the detection of low frequency variants. These elements were therefore chosen to be governed by a single simulation parameter as to not complicate even more.

Thirdly, there is the subclonal mutation load, which regulates the amount of new random mutations assigned to any subclonal expansion event. Mainly in order to speed up simulation time, this feature was introduced. Without it, many rounds of random mutations need to precede the subclonal expansion, however currently these are all collapsed into one event. Consequently, this setting is responsible for the possibility of detection of the concerning subclone, since it gives an indication of the subclone's presence in the CCF spectrum. Stochastic as well as deterministic behaviour can be specified by the user to determine the fraction of loci to be mutated during each subclonal expansion event.

The setting 'population size' is also more implementation focused, but also has bearing in the accuracy of the CCFs generated. Since the population represents the tumor as a percentage, this setting might seem redundant, and can indeed be kept at a fixed value for most purposes. Nonetheless, it shapes the discretization of the CCFs, greatly influencing the run time of the simulation. Considerations on this particular setting should be application dependent, for the tool estimation performance testing 200 is the default that is used.

The number of loci within each cell in the tumor population is related to the number of mutations (mutation load) that is desired. This setting is especially interesting during examination of the tool's performance behaviour with regard to the infinite sites assumption (next subsection). Additionally, it can serve as the number of SNVs that need to be simulated, as it will for the remainder of the report. It has not left the notice that these loci can be used for mutations other than SNVs, which could be an added component in future analyses.

The sixth parameter, sequencing noise, is introduced in a simplistic and rather artificial manner. Because of this, the parameter should be tweaked with care. As will be explained in the subsection 4.2, this parameter has a big effect on the legitimacy of the generated data.

The last parameter discussed, the sampling times, are rather straightforward. This setting indicates at which times during the simulation a sample must be extracted. These samples consist of the same information as the general (last) sample of the tumor population. This information is essentially the CCF spectrum (used for tool input) of the population accompanied by the ground truth (used for tool estimation error determination).

4.1.2 Infinite sites assumption

In pursuit of resembling real world cases with the simulation output, some effort was done to make sure that multiple simulation parameters were within a realistic range. These include the mutation rate of actual cells[47], however the inter-cancer variability makes statically "correct" settings unattainable. One of the aspects that can be considered, is also one of the central assumptions employed by many tools. This concerns the hypothesis that the same mutation doesn't happen in parallel subclone lineages. This phenomenon was investigated by a framework utilizing single-cell sequencing to quantitatively investigate this "infinite sites assumption" (ISA).[48] To elaborate, the ISA essentially implies a mutation event on any single locus does not happen more than once. Therefore, the assumption states that for every unique locus no two parallel mutations will take place, and thus no sister subclones can have the same mutations, given these mutations are not found in their common ancestor (see the formulation in Equation 13).

Given subclonal composition V, E with subclones $v_i \in V$, where S_i denotes the set of mutated loci with individual loci $s_i^j \in S_i$ within this subclone.
The ancestral relations are given by $e_i^j = \{v_i, v_j\} \in E_i$ and cover all child,parental,grand-parental,grand-child,etc. (non sister) relationships between of subclone v_i.

The ISA states:

$$P(\forall s^j) << 1 \Rightarrow \neg \exists \{s_i^j, s_k^j\} \; \forall \{v_i, v_k : i \neq k, v_i \notin e_k \in E_k\} \tag{13}$$

However, there are strong indications that tumors do violate this central assumption present in many tools.[48] Still, effort was made to optionally conform the data simulation to the ISA, for adequate testing of the tools. In order to quantify violation of the ISA within the simulated data a small analysis was conducted. The main factor considered was the highest number of mutations (mutation load) attainable, given a certain level of confidence that the ISA would not be violated. The mutation load is a crucial element of the data, since this is representative of the information available for subclonal reconstruction. A low mutation load would quickly form a bottleneck for accurate reconstruction (as is later shown), but too many mutations intuitively violate the ISA. Figure 8 shows an overview of the relevant dynamics of this mutation load considering a 95% certainty of not violating the ISA. Here the mutation events are modelled as random variables from a binomial distribution and the resulting probability of not violating the ISA can be calculated as seen in Equation 14a-f. The previously mentioned mutation rate is here referred to as ”mutation chance”, since the chance of mutation is regarded in this model.

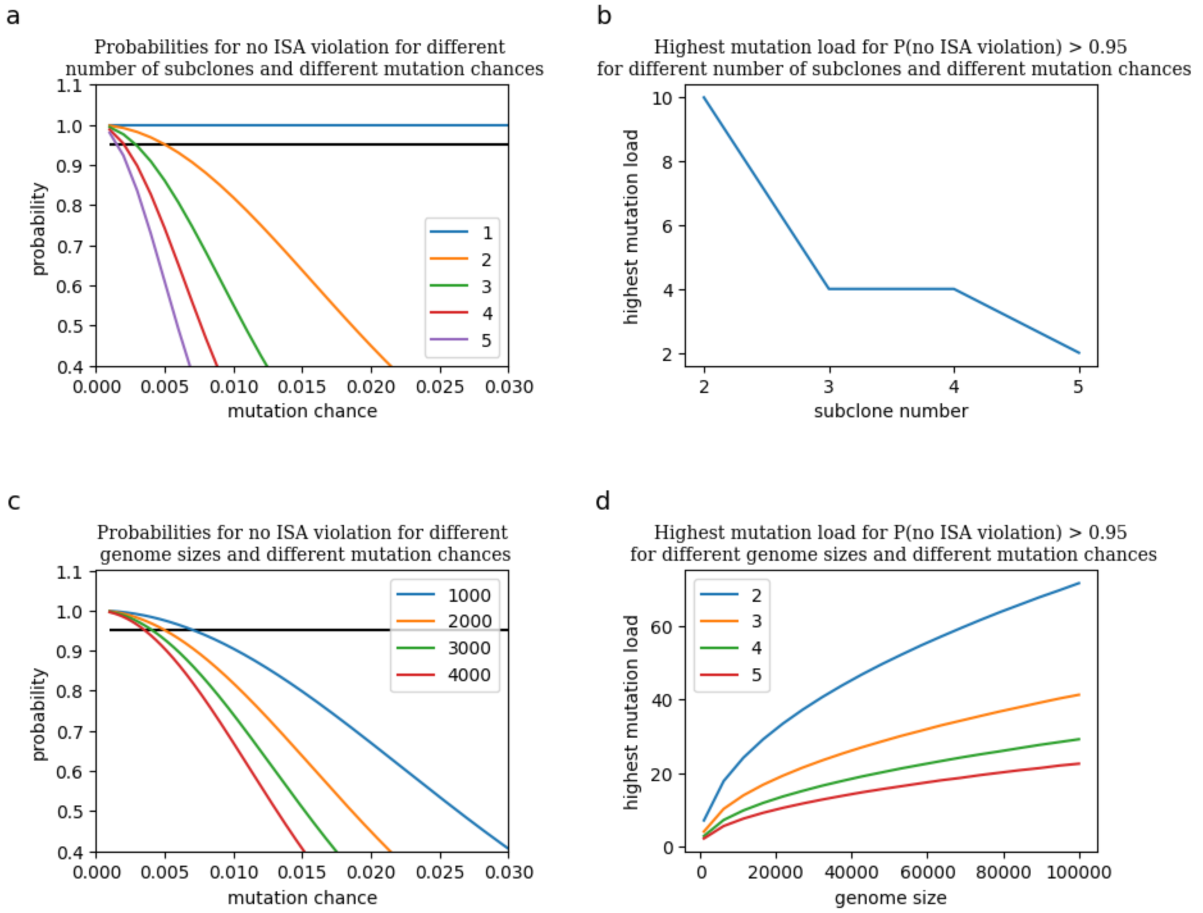


Figure 8: Plots showing the properties of the infinite sites assumption (ISA). **a**: Plot of mutation chance of 2000 loci vs the probability of not violating the ISA for different number of subclones. The horizontal black line indicates 95% probability, which is used as a minimum in panel **b**. **b**: Per number of subclones the highest mutation load, having 95% certainty of not violating the ISA for 2000 loci. Here the highest mutation load is calculated using the highest mutation chance above the 95% mark times the number of loci. **c**: Plot of the mutation chance of different numbers of loci vs the probability of not violating the ISA for 2 subclones. The horizontal black line indicates 95% probability, which is used as a minimum in panel **d**. **d**: Per number of loci available the highest mutation load, having 95% certainty of not violating the ISA. Again the highest mutation load is calculated using the highest mutation chance above the 95% mark times the number of loci.

To some degree, it can be made sure that the simulation will not violate the ISA. As can be seen in Figure 8 in panel **c** and **d**, decreasing the mutation chance and increasing the genome size will lead to a higher mutation load, while keeping the probability of violating the ISA to a minimum. Nonetheless, this proved to extend the simulation's execution time to dramatically, therefore a direct implementation into the simulation was added. This extension of the simulation makes sure that multiple independent mutations on the same locus in parallel subclonal expansion events was removed. Now the subclonal expansion events are implemented to not have overlap in their mutation signatures resulting from the expansion itself. This will not only help test the quantitative performance of a wider range of tools, but also allow for smaller genome sizes to be simulated.

$$A_x : \quad \begin{array}{l}\text{The event where for locus index } x \text{ there} \\ \text{is no more than 1 mutation across } K \text{ subclones} \\ \text{in a cell population with genome sizes } M\end{array} \tag{14a}$$

$$P(\mathrm{A}) = P(x \leq 1) = P(x = 0) + P(x = 1) \tag{14b}$$

$$= (1-p)^K + \binom{K}{1} p(1-p)^{K-1} \tag{14c}$$

$$P(\text{No ISA violation}) = P(A_0 \cap A_1 \cap ... \cap A_M) \tag{14d}$$

$$= P(A_0) \times P(A_1) \times ... \times P(A_M) \tag{14e}$$

$$= P(A_0)^M \tag{14f}$$

4.2 Data acquisition

After the data is simulated, certain steps are required before the tools can be executed. The data generated in the simulation requires some specific processing and formatting, after which it is ready to serve as input for the tools.

The first step in this process is the acquisition of the CCFs of the population from the earlier discussed *txt-files* containing the mutation bit strings. Since these bit strings represent the mutation profiles of the cells in the tumor population only and contain no CNVs, no further corrections need be to made with regard to the purity of the samples. The purity for all following simulations can assumed to be a 100%, because other parameters will be tested. The average number of mutations of each loci now simply represents a frequency of mutation for this loci, since the mutation is defined as value 1 and wild type as 0. This frequency of mutation can also be seen as the fraction of cells within the tumor population carrying this mutation/variant. This will form to be the spectrum of CCFs for the sample. Note that this is done for multiple samples (separately).

The CCFs with a value of 0 are removed since these carry no variant information. The CCFs with value of 1 on the other hand indicate that the associated variants belong to the common population ancestor. These are kept, together with all the other CCFs > 0. Now, a normally distributed random number is added to the CCFs, serving as an substitute for the sequencing noise. In order to get realistic CCFs spectra, this 'sequencing noise' parameter must be combined with the 'mutation noise' parameter, where the latter is responsible for the low frequency/fraction variants. The low frequency variants derived from low frequency subclones, are a problem in the clinic as their presence is usually associated therapy resistance, since this form of subclones are hard to detect and this group can be comprised of a therapy-resistant species. It is important to nudge these parameters towards realistic cases, since these form the main bottleneck achieving realistic data.[49] By applying the noise in this particular way, it is confirmed that the mutations simulated are SNVs, as other mutations (such as structural variants or CNVs) behave rather differently with respect to their CCF.

The resulting spectrum of CCFs is then used to determine the input of the tools. Although this is fairly particular for different tools, the guidelines for producing them are the same. It is made sure that the CCFs spectrum representing the SNVs is essentially the only information available for the tool. Next to this, the information is assumed to be copy number neutral. Let us consider the example where the tool requires the VAFs of the all SNVs: this can be calculated by simply dividing the current spectrum by two (diploid, copy number neutral), as done in Equation 15. In the case that information is requested by the tool, which not able to be calculated from the CCFs, this is left as a blank or a neutral value. This is repeated for all the samples fed into the tool, sometimes requiring meta-data of the samples - such as the sample chronological ordering - which ís provided as this is also known in real cases.

4.3 Tool analysis

The initial tools that were integrated into the framework are: *DPClust*, *PhyloWGS*, *SciClone*. These tools were chosen because of their popularity in literature and have been fully integrated into the framework. The full integration of the tools is important for automation purposes and makes the framework more user-friendly. Each of the tools was installed independently and called using terminal commands from a *Python* script during the analysis of the tool.

The goal of the framework is ultimately to compare the tools for different parameters, and give future users an overview of the responses to certain input parameters. It has to be noted that the side-by-side representation of the tools may invoke an inaccurate comparative judgement. Meaning that the tools should not be directly compared for certain settings, since resources such as the run time are not equally distributed over the tools, although this difference was kept to a minimum. Additionally, the results for *PhyloWGS* and *DPClust* are shown in the same plots, its must be kept in mind that *PhyloWGS* produces more information (subclonal phylogenies) than *DPClust*. Also, within this specific case *DPClust* had higher run times which means it used more resources and therefore had more time to make an accurate estimate (even though both tools ran for the same number of MCMC iterations). Keeping these kind of factors in mind, the tools can be compared.

The kind of actions taken to keep the resources constant include: the same cutoff frequency for the variants; the same number of MCMC iterations and burn-in samples; the same number of CPUs allocated; and certainly the same input data. If there was any deviation from these, it will be mentioned.

The analysis of the tools, was executed from a command-line-interface (CLI) connected to a *Python* script. The tools are ran consecutively for a single input, which is done for multiple processes in parallel, each handling a distinct simulation. A tool gets to do multiple estimates for the same simulation, the number of which can be specified. Each simulation-estimate pair outputs the following: estimated number of subclones, the estimated subclonal fractions, the true number of subclones, the true subclonal fractions and the run time of the estimation. These five data points are exported and later used to examine the tool's performance. No phylogenetic estimations are taken into account. An overview of the script is shown in algorithm 5 supported by Figure 10, where it can be seen how the data is fed into the tools, and how the output is stored and exported. For each simulation a page is added to the report giving an overview of the input data, on these pages a fishplot and the CCF spectrum histogram are presented together with an overview of the tools' estimations in a table. Finally all the data is summarized in a final page, containing a plot of the parameter space specified vs the tools' performances. The data is also exported using *Numpy*, which is later used for actual data analysis. The data visualization is rather straightforward and is comprised of the *Pandas* and *Seaborn* - *Python* libraries for data frame handling and visualization respectively.

The data that is extracted from a single run has 5 dimensions, as is illustrated in Figure 9. Of the 4 primary dimensions, 3 provide interesting results when summarizing over the others. These 3 dimensions are: the parameter values, the phylogenies and surely the tools (the fourth dimension, estimations per tools are regarded exchangeable and are only there for statistical validation). Effort was put into analyzing the relationships between these 3 data dimensions with regard to each of the data points. However, this encompasses too many combinations to be illustrated in this section, so most will be found in the Supplementary material.

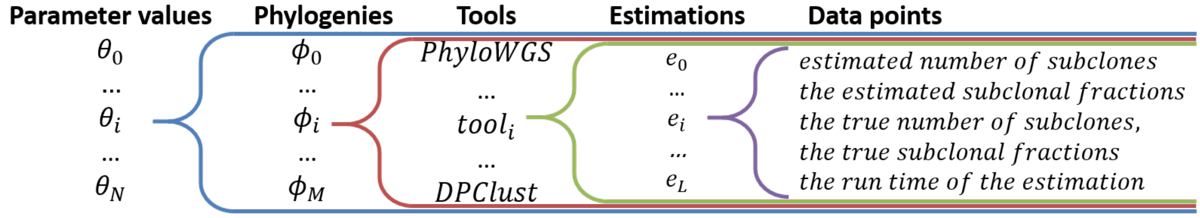


Figure 9: Schematic of the data structure and its dimensions. For each parameter value all phylogenies are simulated. The simulation samples of each of these is given to all of the selected tools. These tools make a specified number of estimations per simulation. The estimations consist of the: number of subclones, subclonal fractions and the run time. Next to these three values the true number of subclones and the true subclonal fractions relating to the simulation at hand are recorded.

Algorithm 5: Overview of framework

```
pseudo-code ui.py($CLI - arguments, settings.py$)
Input  : Within $CLI - arguments$:
           Set of tools(programs) to run $\mathcal{T}$,
           Single parameter with parameter space $\tilde{\theta}$,
           Set of simulation parameters $\theta$ (modifications to default)*,
           Sequencing noise $q$*,
           Number of phylogenies/loops $L$,
           Phylogeny ordering $\phi$*,
           Number of estimations(subloops) $S$            $*{=}optional$
         Within the $settings.py$:
           Set of default simulation parameters $\theta$and single parameter with range $\tilde{\theta}$ specified by user.
Output: Estimation data points of all tools $\mathcal{T}$ forset of simulation parameters $\theta$ and parameter space $\tilde{\theta}$and for
         different loops/phylogenies $L$, repeated $S$ times.
         Estimation data points consist of:
           Number of subclones estimated,
           Subclonal fractions estimated,
           The run time of the estimation.
         Each paired with 'true values':
           True number of subclones,
           True subclonal fractions.
$\theta$ = importSettings($CLI - arguments, settings.py$)
$\tilde{\theta}$ = constructParameterSpace($CLI - arguments(\tilde{\theta})$)
setPhylogeneticLoops($\phi, CLI - arguments(L)$)
data = []
for $\theta_i$ in $\tilde{\theta}$ do
    for $l$ in $0 : L$ - parallel execution do
        dir = setOutputDirectories($l$)
        reads,$datapoints_{true}$,fishplot,CCFs histogram = simulateData($l, \tilde{\theta}, \theta$)
        reads = addSequencingNoise(reads)
        data[$l$].append($datapoints_{true}$)
        for tool in $\mathcal{T}$ do
            estimations = []
            for $s$ in $0 : S$ do
                runTool(tool, $l$, reads, dir)
                $datapoints_{estimation}$ = extractDatapointFromEstimation(tool,dir)
                estimations.append($datapoints_{estimation}$)
            end
            data[$l$].append(estimations)
            addReportPage(fishplot,CCFs histogram, data[$l$])
        end
    end
end
$data_{processed}$ = summarizeParameterSpace(data)
addReportOverviewPage($data_{processed}$)
exportData(data)
clearAllDirectories()
```

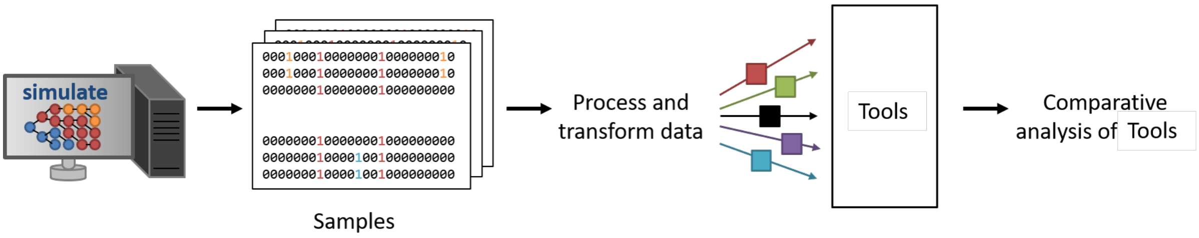
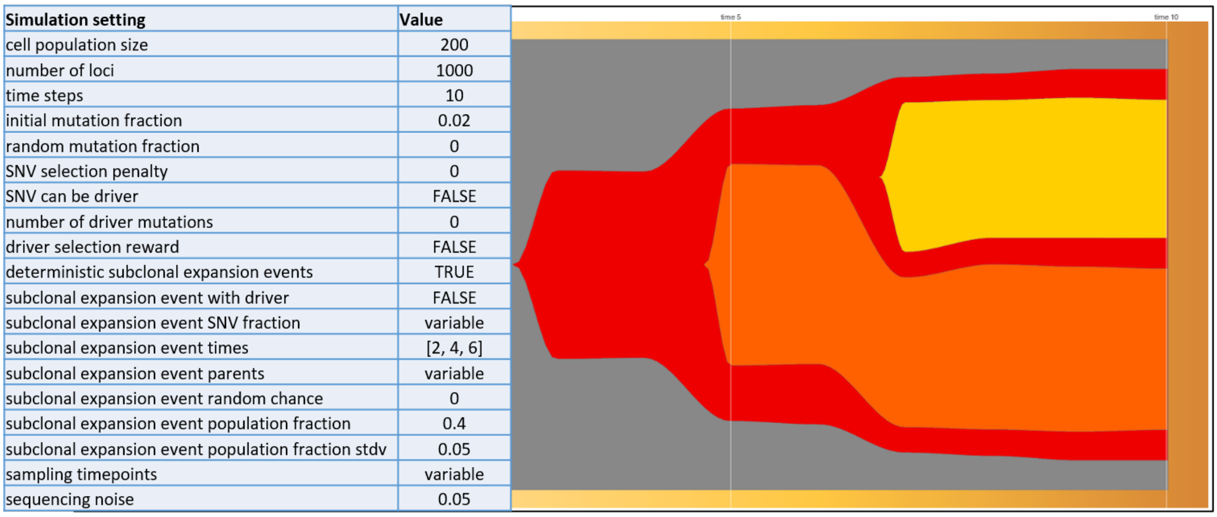


Figure 10: Schematic of the conceptual steps used in the constructed framework. Data is simulated, from which samples are taken. These samples are processed to CCFs and used to generate the tool input data. The tools then perform the estimations. Doing many of these estimations for different simulation parameters can then reveal possible characteristics: differences within tool for simulation parameter space, and the differences between tools.

4.4 Results

The performances of two of the tools will be shown in this section. As previously stated, the side-to-side representation of the tools should not be interpreted as a direct tool comparison. Next to this, it is good to keep in mind that the estimations given by the tools are based on very low iterations/MCMC sampling, in order to do more estimations and get a better statistical representation of the parameter space. The actual estimations of the tools are orders of magnitude better, but here only the relative performance between parameter settings is shown. Since the number of MCMC samples has a high impact on the accuracy of the estimations by the tools. An analysis of the MCMC sample number in relation to performance is shown in Supplementary subsection 7.2.

A look will will be taken at *DPClust* and *PhyloWGS*, since these tools responded well to various parameter inputs. The two simulation parameters that will be explored are "mutational load" 0.001 to 0.01 inclusive (0.0005 step size) and "the number of samples taken" ranging from 1 to 4 inclusive. The former is defined as the fraction of simulated genome (1000 loci) mutated in every subclonal expansion event. The auxiliary settings for the simulation are shown in Figure 11. Of these, worthy of mentioning are the number of subclones (*subclonal expansion event times*) and the mutation load (*subclonal expansion event SNV fraction*). The number of subclones was fixed at 3, chosen because of the simple integration of all possible phylogenetic compositions (only 6) and because this number should not be problematic for any tool. The mutation load, is the main simulation parameter for which the differential tool response is tested. Varying this parameter would allow us to see the number of variants that is required for a tool to make a good estimate. The metric "correctness" used below, refers to the fraction of correct estimations of the number of subclones present in the tumor architecture.



Simulation setting	Value
cell population size	200
number of loci	1000
time steps	10
initial mutation fraction	0.02
random mutation fraction	0
SNV selection penalty	0
SNV can be driver	FALSE
number of driver mutations	0
driver selection reward	FALSE
deterministic subclonal expansion events	TRUE
subclonal expansion event with driver	FALSE
subclonal expansion event SNV fraction	variable
subclonal expansion event times	[2, 4, 6]
subclonal expansion event parents	variable
subclonal expansion event random chance	0
subclonal expansion event population fraction	0.4
subclonal expansion event population fraction stdv	0.05
sampling timepoints	variable
sequencing noise	0.05

Figure 11: Overview of the settings of simulation runs corresponding the mutation load comparison. The left table shows all the important user-defined simulation parameters. Here the variable settings are: mutation load, simulation parameter space; phylogentic relation, all possible are executed; and number of sampling time points, either 1 or 2. The right partially shows one of the simulation fishplots (1 of 6 phylogenies)

In the first analysis as many as 9120 estimations were carried out by each of the two tools, of which a total of 3592 correctly estimated the present subclone number ($correctness \approx 20\%$). The data was collected in 4 separate runs, in order to validate that the run stochasticity would not have an effect on the actual result. The figures from the 4 separate runs had negligible differences, and would support the same conclusions. After this validation the data was merged and Figure 12a and Figure 12b were produced. The number of iterations/MCMC samples given to both tools was 25 with 12 burn-in samples (a 100-fold is recommended in normal estimations).

Looking at the performance of *PhyloWGS* in Figure 12a and Figure 12b, notice the stable behaviour and little change in performance for higher mutation loads. This plateau indicates that the difference in performance is smaller for this part of the parameter space. For the lower mutation loads however, it can be observed that this dramatically interferes with the effectiveness of the tool. Which is fair considering that 0.003 is approximately 9 variants. For *DPClust* on the other hand, a small number of variants per subclone allows for a better estimation of the subclonal composition. This parameter range does not yield conclusive differences for real world applications, but rather shows some of the intrinsic properties of the tools. The fact that *DPClust* seems to prefer 1-5 ($0.001 - 0.005$ mutation load) mutations per subclone over 6-10 mutations ($0.006 - 0.01$ mutation load) indicates that the tool does not necessarily benefit from extra data points. *PhyloWGS* responds differently to the slight increase in information, since it actually prefers the 6-10 mutation number over the 1-5 mutations.

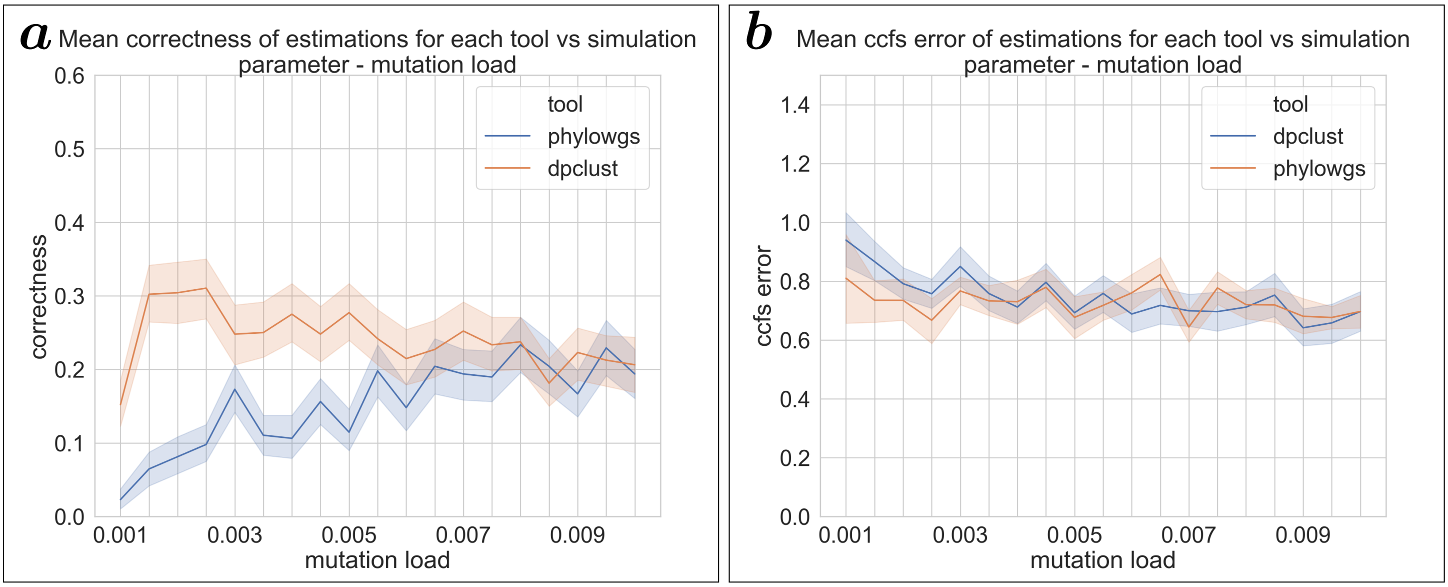


Figure 12: Data contains 18240 tool estimations divided over: 4 runs, 6 unique phylogenetic trees, 19 mutation load values, 2 tools (*DPClust* and *PhyloWGS*), 10 estimations and 2 sample numbers (1 and 2). **a**: Plot showing the fraction of correct estimations of the number of subclones vs the mutation load. **b**: Plot showing the average error in the estimated subclonal fractions (CCFs) vs the mutation load. The subclonal fractions (CCFs) errors are normalized by the number of samples taken.The colored interval marks one standard deviation.

Looking at a broader range of mutation load values, a more intuitive picture can be observed. Here the mutation load values studied range from 0.001 to 0.301 inclusive (step size of 0.06). These values in combination with the 1000 loci simulated genome size, therefore represent 1 to 301 mutations (with steps of 60 mutations) per subclone. This second mutation load analysis has 5760 estimations carried out by each of the two tools, with a total of 2992 correctly estimated the present subclone number ($correctness \approx 26\%$). Again data was collected in 4 runs, validating that the run stochasticity would not have an effect on the actual result. The data was merged into a single data frame and this resulted in Figure 13a and Figure 13b. The number of MCMC samples was unchanged.

Figure 13 shows that an increase in mutation load, and thus an increase in variants, enhances tool performance. This can be attributed to the increase in information available for the subclones, since they attain more passenger variants that can be sampled. A clear increase in performance can be seen in the first step of Figure 13a. Although *PhyloWGS* does not seem to improve with this specific amount of MCMC samples, *DPClust* does seem to profit from the increase in variants. This might be a result of *PhyloWGS*'s better scaling with the MCMC sample number value as seen in Supplementary subsection 7.2. Nonetheless, the error in the subclonal fractions (CCFs) for both tools remains very stable indicating no increase in accuracy, as seen in Figure 13b. Similarly, an analysis for ”genome size” is done in Supplementary subsection 7.3.

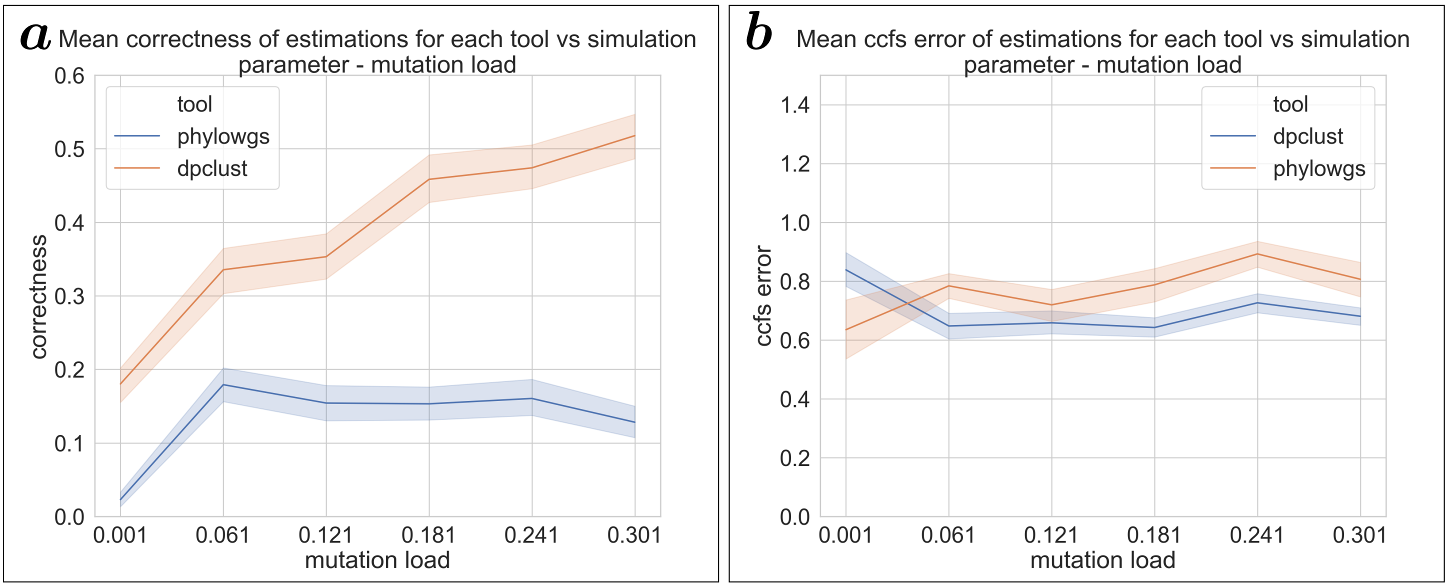


Figure 13: Data contains 11520 tool estimations divided over: 4 runs, 6 unique phylogenetic trees, 6 mutation load values, 2 tools (*DPClust* and *PhyloWGS*), each simulation comprised of 20 estimations and sampled at 1 or 2 time points. **a**: Plot showing the fraction of correct estimations of the number of subclones vs the mutation load. **b**: Plot showing the average error in the estimated subclonal fractions (CCFs) vs the mutation load. The subclonal fractions (CCFs) errors are normalized by the number of samples taken.The colored interval marks one standard deviation.

Secondly, the performance for different numbers of samples taken is examined in Figure 14a and Figure 14b. These figures were constructed using a partially identical data set consisting of 2 separate runs of 11520 estimations, of which a total of 1883 correctly estimated the present subclone number ($correctness \approx 16\%$). Within this data set the number of samples taken was the simulation parameter, covering 1,2,3 and 4 samples. Again the number of variants per subclone ranges from 1 to 301 mutations (60 step size), but are summarized, since this for now not the parameter of interest. Strikingly, the correctness generally doesn't increase when taking additional samples. This might indicate that the fitting of the estimated subclonal prevalences over the samples actually complicates the reconstruction overall. This was further investigated, hypothesizing the number of samples might require more MCMC samples (this is not the case as seen in Supplementary subsection 7.5). To confirm these results a similar analysis was done using the "genome size" parameter (see Supplementary subsection 7.4). Similar results are being achieved by another analysis done in the paper introducing *Mobster*[15]. Their finding is concerning the multi-spatial sampling, which might be influenced by the same effects.

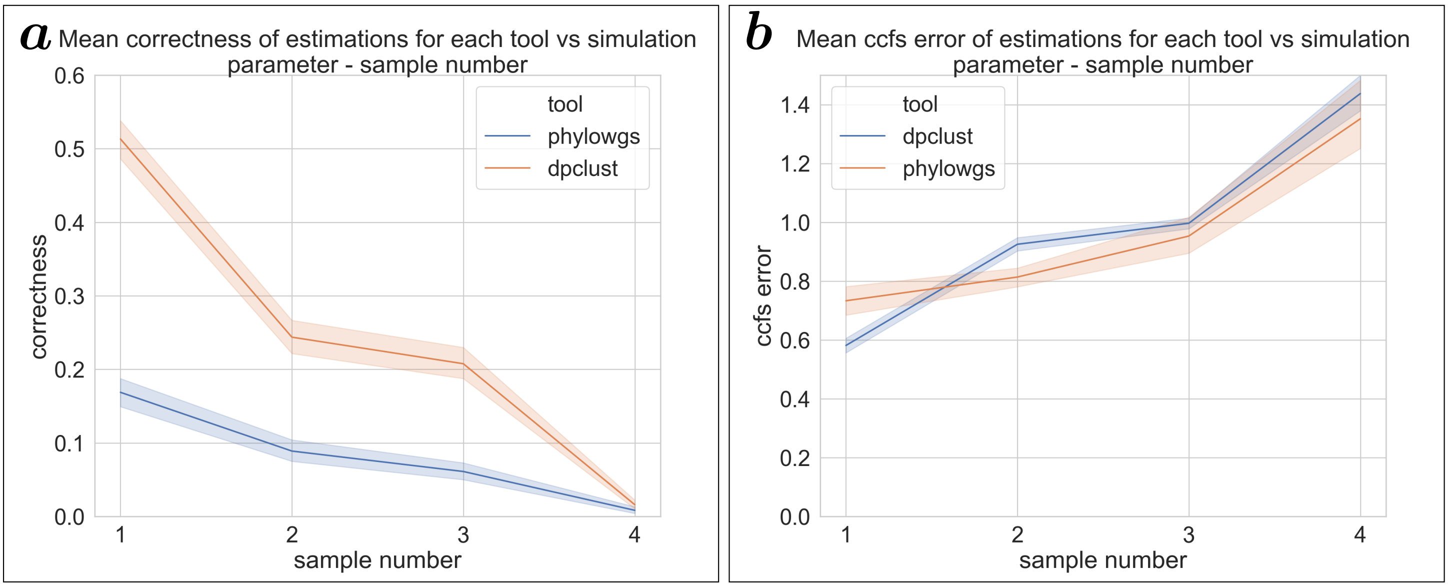


Figure 14: Data contains 11520 tool estimations divided over: 2 runs, 6 unique phylogenetic trees, 6 mutation load values (0.001 0.061 0.121 0.181 0.241 0.301), 2 tools (*DPClust* and *PhyloWGS*), 20 estimations and 4 sample numbers. **a**: Plot showing the fraction of correct estimations of the number of subclones vs the number of samples taken. **b**: Plot showing the average error in the estimated subclonal fractions (CCFs) vs the number of samples taken. The subclonal fractions (CCFs) errors are normalized by the number of samples taken.The colored interval marks one standard deviation.

To support the graphs in Figure 14 and show the differences between the execution times of *PhyloWGS* and *DPClust*, a plot of all the correct estimations of each tool is made. Figure 15a and Figure 15b illustrate the performance of a tool while also capturing the computation time needed to make the estimation. Again, here a clear correlation between the sample number increase and the subclonal fractions estimation error is observed. A notable contrast between the two tools is also seen in respect to their run time: clear separation of estimations for district sample numbers is observed for *DPClust*. *PhyloWGS* on the other hand actually has a different run time to sample number relation and shows no such definite separation.

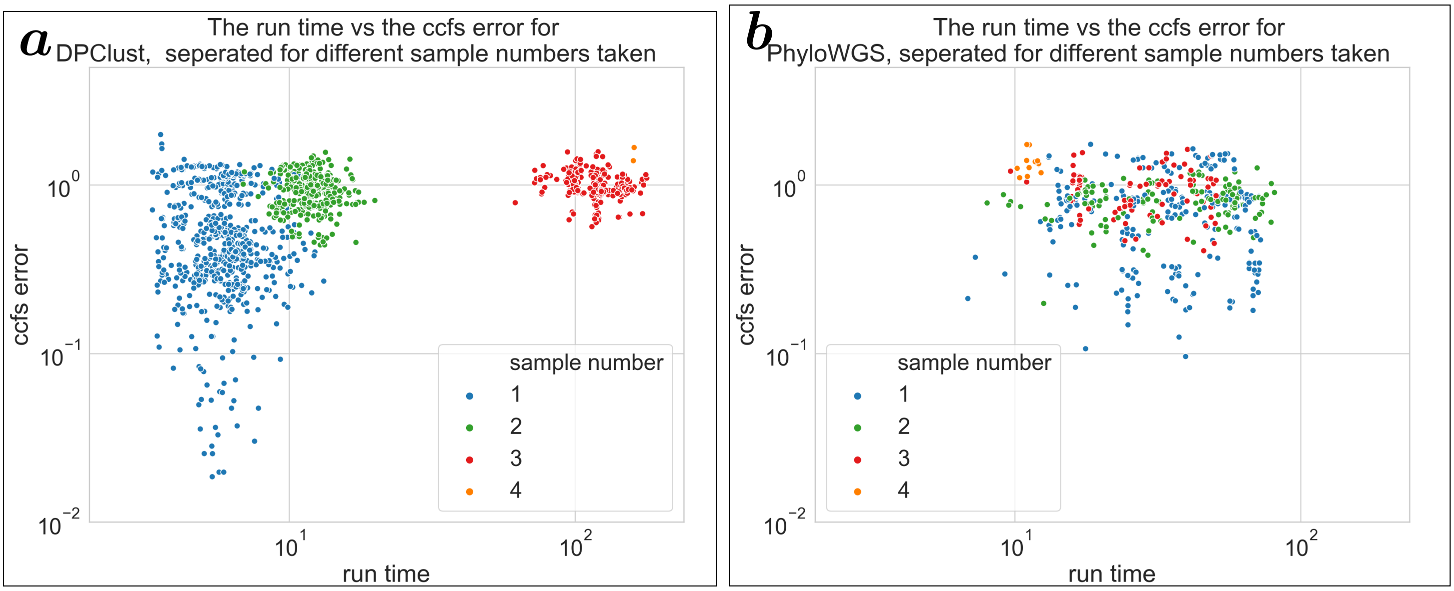


Figure 15: Data contains 11520 tool estimations divided over: 2 runs, 6 unique phylogenetic trees, 6 mutation load values (0.001 0.061 0.121 0.181 0.241 0.301), 2 tools (*DPClust* and *PhyloWGS*), 20 estimations and 4 sample numbers. Same data as in Figure 14. **a**: Scatterplot showing the correct subclone number estimations of *DPClust* in the *log* run time *log* subclonal fractions (CCFs) error space. The different number of samples taken are highlighted. This data is comprised of a total of 1412 corrects subclone number estimations. **b**: Scatterplot showing the correct subclone number estimations of *PhyloWGS* in the *log* run time *log* subclonal fractions (CCFs) error space. The different number of samples taken are highlighted. This data is comprised of a total of 471 corrects subclone number estimations.The colored interval marks one standard deviation.

5 Real data application

To show a different utility of the framework, the bulk tumor samples from 8 patients were analyzed. The data consists of the whole genome sequencing (WGS) data from the tumor, which was sampled at two distinct time points. All of these tumor samples where from metastatic sites of Bladder Cancer (BC) with the exception of patient 3, whose cancer was classified as an Upper Tract Urothelial Carcinoma (UTUC). All patients also had some form of treatment - either immunotherapy or chemotherapy - sometimes in combination with radiotherapy. The response to these treatments was categorized as one of the following: Partial Response (PR), Stable Disease (SD) or Progressive Disease (PD). These treatments might have had some effect on the subclonal composition in their tumors, which is the reason why the samples in the patient data are kept in this context.

The sample location of the 2 biopsies of each patient where taken from the same sites, which are either lymph node or soft tissue - with the exception of patient 7 whose biopsy sites where different, and whose first biopsy was from bone labelled as subcutaneous tissue. The tumor purity of the samples ranged from 0.21 to 0.93, which was corrected for in this report's analysis using the CCF calculation. The time between the two biopsies ranged from 1 month to 9 months. The number of variant loci measured in each of the sample was on average ±14700.

For each pair of samples, a subclonal reconstruction was made using both tools, *DPClust* and *PhyloWGS*. After removal of variants which were not found in both samples, the CCF was calculated for every variant.[37] From this value as in the simulated data the actual tool input was calculated.

The figures of patient 4 and 8 obtain some seemingly abrupt cutoffs for the CCF clustering. These are most likely the result of data processing rounding errors while calculating the CCFs of the variants, and are therefore only shown in Supplementary 7.6.

For *DPClust*, the input called for a nuanced picture of the variants. Copy number estimations and "the expected number of chromosomes containing a variant" were required. Although much of this information should be captured in the CCFs themselves, the tool did obligate accurate description of these values together with NGS data read counts. Unlike *PhyloWGS*, *DPClust* can work with the actual number of variants supplied. It does however not provide a phylogenetic estimation of the subclonal composition. The variants are simply assigned to a cluster using the DP, similar to *PhyloWGS*. The results from *DPClust* were plotted manually using a scatter-plot Figure 16. Additionally, the top 20 mutated cancer genes (percentage) for Bladder Cancer in the TCGA program[50] were labelled if present within any of the samples. These labels can also be found in the same figures. When taking a closer look at these top 20 mutated genes, it should be noted that (for most patients) the mutated genes are mainly present (if at all) in the parental subclone. This indicates that the metastatic site originated from cells containing these mutations and that the secondary subclones at this site did not originate from any of these mutations specifically. However, in patients 2,3,6 additional mutations can be seen in the secondary subclones. Patient 2, provokes the hypothesis that the EP300 variants have played a role in the expansion of the secondary subclone. Similarly in the case of patient 6, KMT2C may well have played a role in the genesis of the secondary subclone. For patient 3, already many of the selected gene mutations (TP53,FAT4) have occurred, upon which additional mutations to these genes are on to form the new subclone. Also the a PIK3CA mutation was introduced and clustered into the secondary subclonal variants. This broadens the spectrum of variants that could be responsible for the subclonal expansion of patient 3.

The most important input data required by *PhyloWGS* consists of the reference read count (non-variant reads) and the total read count, the latter being directly available from the sequencing data. The reference read count from the sequencing data was then corrected for the CNVs present. The value can be calculated from the CCF which were already determined for every variant in the patient data and using the total read count. This is shown in Equation 15.

$$\begin{aligned} &\text{Given total read count } r_{tot,i} \text{ and the CCF } f_i \\ &\text{CN-Adjusted reference read count } = f_i \times 0.5 \times r_{tot,i} \end{aligned} \tag{15}$$

Unfortunately, the *PhyloWGS* run times scale linearly with the number of variants used. This number of variants (average ±14700) would result in runs of ±41 hours per patient when extrapolating the run time graph within the *PhyloWGS* paper[10]. Because of this, and considering that many variants would come from a single subclone, it was chosen to make a random selection of 2000 variants to represent the tumor, which would drastically improve execution time. To additionally increase the speed of *PhyloWGS*'s estimation, the iterations where multi-threaded (as is an option inside the tool). To validate the results, another estimation was done with a different random selection of 2000 variants. This yielded similar findings. The number of MCMC samples and burn-in samples, which are used in the DP to make an estimate, were 250 and 125 respectively. The final estimation of *PhyloWGS* then shows the predicted number of subclonal populations and their CCF trajectories, with the variant-subclone assignments. Also each estimation from the *PhyloWGS* tool by default includes a most probable phylogenetic tree corresponding to the subclonal relations. The information from the highest tree - having the highest normalized log likelihood (nlgLH) - is manually plotted together with *PhyloWGS*'s phylogenetic estimation in Figure 17. As can be noticed from the phylogenetic graphs most lineages were estimated to have only linear dependence, which gives a better CCFs correspondence, although it is not shown this was also observed during the analyses. This may well mean that both secondary subclones estimated for patient 2,5 and 7 could instead have emerged from the same clonal population.

The coherence of both these tools, with respect to the estimation done on the CCFs of the subclonal populations, is striking, although different numbers of subclones are estimated in patients 2 and 5. Here we must consider the vagueness of the definition of a subclone.

The overall behaviour of most of the tumor's subclonal compositions are estimated to be rather stable, considering that these patients are going through cancer therapy. Nonetheless, some subclones show a growth or shrinkage of their prevalence in the tumor population, indicating the presence of the subclonal dynamics.

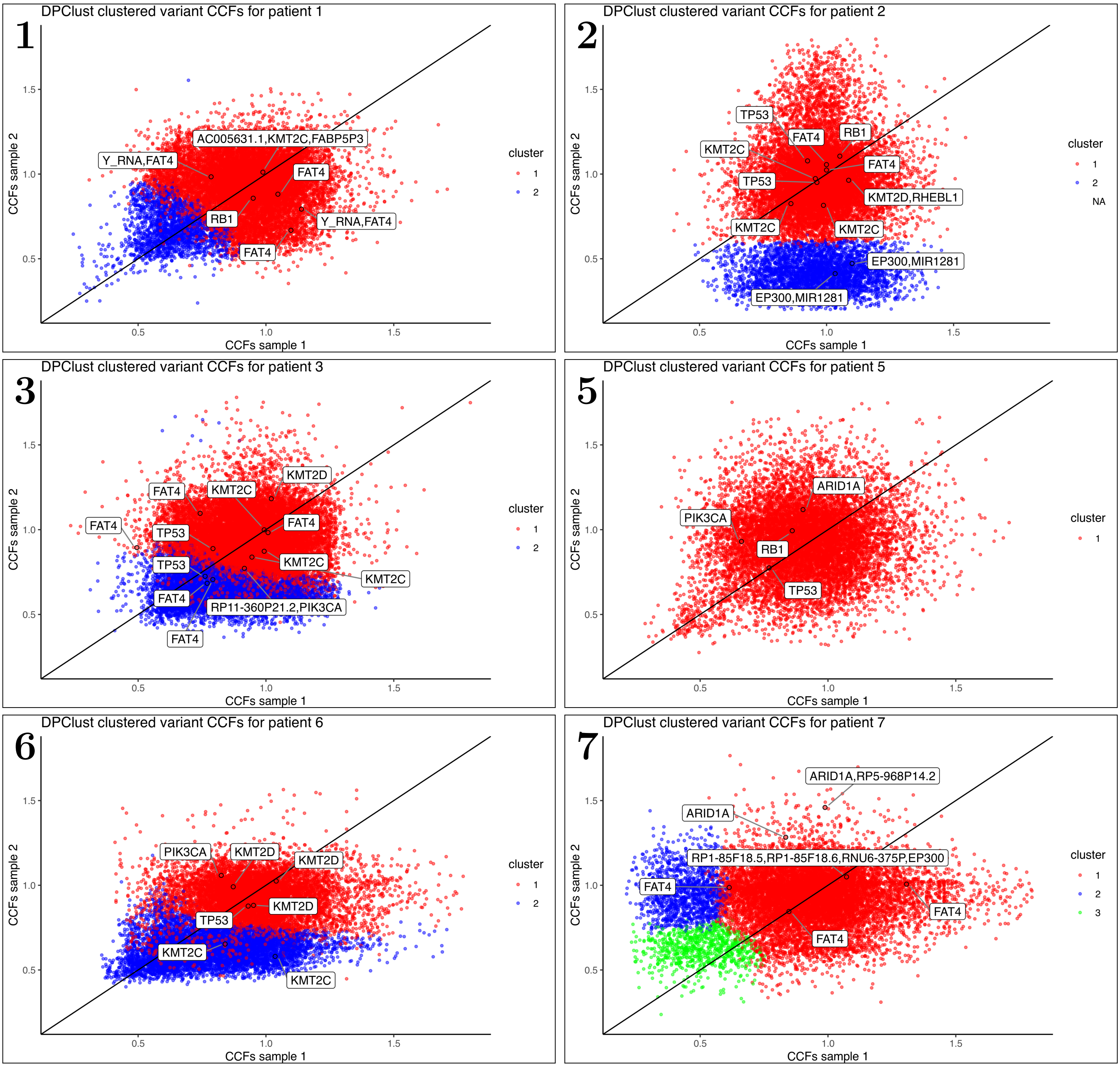


Figure 16: The DPClust clustering assignments of the patient data. Each panel corresponds to one of the 8 patients as indicated in their top-left corner (patients 4,8 are shown in Supplementary 7.6 Figure 31). Within every panel, an individual point represents a mutation VAF measurement, which was transformed to CCFs using the CN information. The horizontal axes indicate this CCF value for the sample in the first biopsy taken, and the vertical corresponds to the second biopsy. The cluster assignment - shown in different colors - was achieved with 375 MCMC samples (and 125 burn-in) of the DPClust tool.

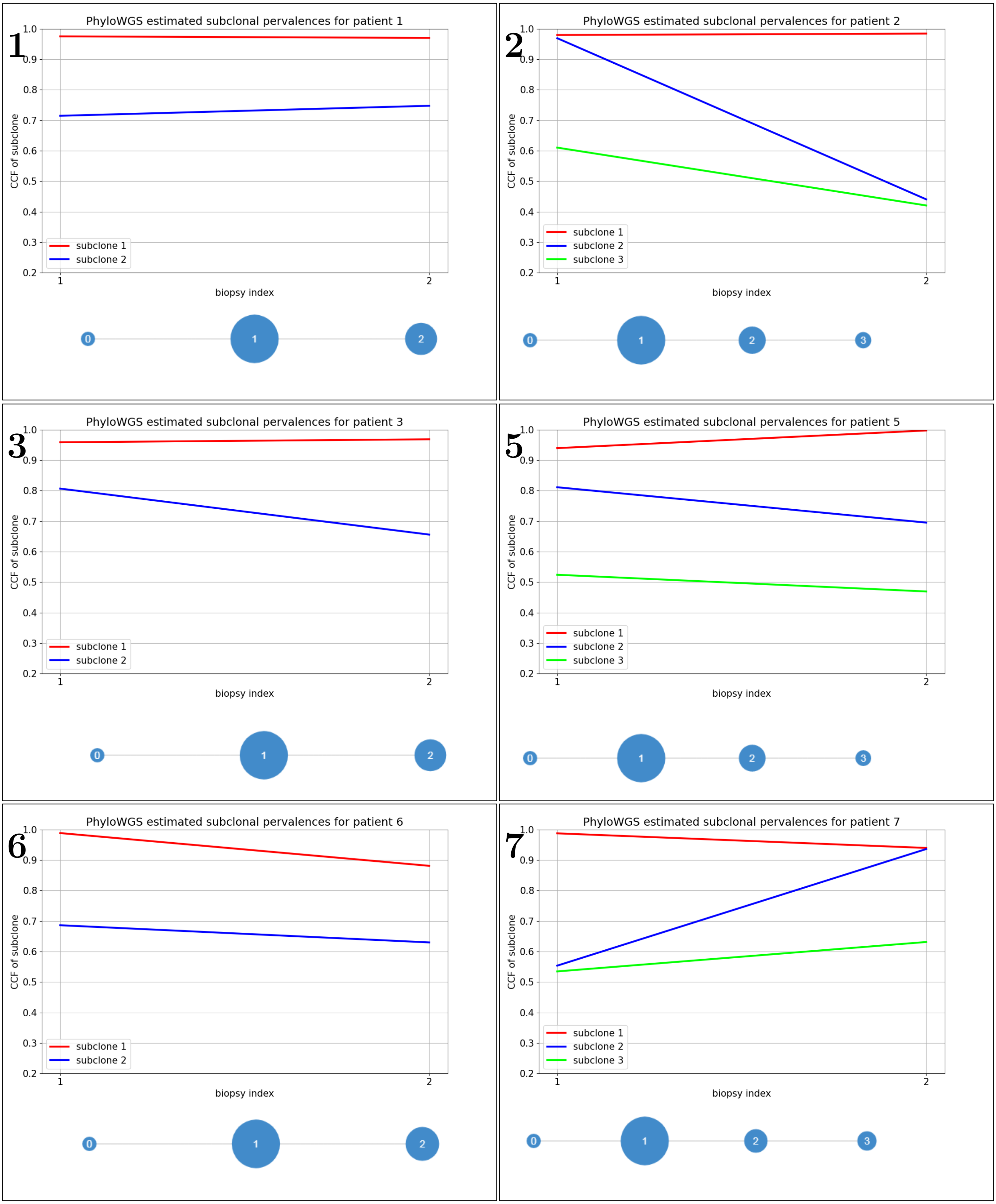


Figure 17: The PhyloWGS's subclonal fractions (CCFs) of its estimated subclonal composition with phylogenetic tree of the patient data. Each panel corresponds to one of the 8 patients as indicated in their top-left corner (patients 4,8 are shown in Supplementary 7.6 Figure 32). Within every panel, the subclonal fractions (CCFs) of all the subclones is shown for both biopsies. Below this plot a graph of the phylogenetic relationship between the subclones is shown (the "0" population is not actually present in the sample, but represents the founding cell tissue from which the subclones stem). Each subclone is shown in a different color, which corresponds to the DPClust estimation if possible. The shown results were achieved with 375 MCMC samples (and 125 burn-in) of the PhyloWGS tool (over 10 separate threads).

6 Discussion

Some of the subclonal reconstruction methods have briefly been explained in order to demonstrate the difficulties faced by the tools utilizing them. This report covered a possible solution for the problems involved in comparing many tools available in subclonal reconstruction. A framework was constructed for the comparative analysis with multiple directly integrated tools. The tools *PhyloWGS* and *DPClust* were then demonstratively inspected for their response to a "mutation load" and the "sample number" parameter space. This produced some interesting findings about each tool's characteristic response, which can be used to find out what the input requirements are for each tool. The difference in the behaviour of these tools can be kept in mind when using them for actual inference of subclonal architecture aspects. A main downside of the studies conducted, is the fact that they had to be done for low MCMC iteration numbers, which was needed for statistical validation. After all, it might make these results less relevant in the realistic case where high MCMC iteration numbers are used.

Although this study was primarily focused on development of a methodical approach, multiple concrete conclusions can be drawn from the analyses that were conducted. The most important finding is concerning the multi-time tumor sampling. The results indicate that using more than 2 tumor samples does not provide a better image of the subclonal composition. The reasons for this phenomenon stay unclear, though it could be hypothesized that greater number of tumor samples taken causes the tool to overfit the data. This has to be further investigated, specifically analyzing the tree-structured stick breaking procedure used to relate these tumor samples to each other. For now, it would be fair to say that the costs outweigh the benefits when collecting and processing more than 2 tumor samples. Still, if this magnitude of tumor samples needs to be analyzed, it should be done with the tool *PhyloWGS* and not *DPClust*, since the latter has a highly inefficient sample-number-to-run-time relation. The analysis conducted by *PhyloWGS* using the same amount of time could therefore use many more MCMC samples, increasing the accuracy. Another conclusion that can be observed is that *PhyloWGS*'s estimations do not necessarily improve when increasing the genetic material sequenced. However, the analysis conducted here does not include CNV information. This information is essential to the estimation of the subclonal composition, and this is where WGS is regarded superior to WES and targeted sequencing panels. Another takeaway is the fact that the two tools - *PhyloWGS* and *DPClust* - can be regarded to be robust and user-friendly, given they are still being used and updated after 5 years of their initial release. This can be confirmed, after having - successfully and unsuccessfully - integrated different tools into a framework.

Many possibilities in the application of this framework have not yet been put into use, due to time limitations. These include the analysis of other simulation parameters, the evaluation of other tools and the correlation of tool behaviour with the underlying methods in order to establish a method-performance investigation. To elaborate on the second - even though other tools were implemented, such as *SciClone* [12], *TargetClone* [32] and *SubClonalSelection* [33], running them inside the framework proved difficult for various reasons.

The integration of *TargetClone* simply did not seem to apply well to multi-time sampling character of the data provided by the simulation. This can be attributed to the central assumption used by the tool that a single subclonal population accounts for majority of any sample. This could be integrated into the framework's simulation by taking sample subsets. The feature of multi-spatial sampling however has not yet been incorporated yet, where the real world "micro-dissecting" would be represented.

SciClone seems to be working fully, although the estimations produced are miscellaneous, without coherence. It did therefore not seem right to include the produced results of this tool, possibly classifying it wrongly. Specifically the tool response seemed very dependent on the number of variants and maximum number of clusters after elaborate modification and testing. Not specific for this tool alone, it did illustrate the inflexibility of subclonal reconstruction tools in general.

Finally, the tool *SubClonalSelection* being programmed in the language *Julia*, was mainly abandoned due to practical reasons. The use of *Julia* environments in the CLI of the server or its cross-language calling from *Python* or *R* scripts proved tedious. Next to this the installation and loading of the pre-compiled packages associated took significant time. All these reasons contributed to the lack of interest in further integration. (Similar reasons limited the integration of the *TrAp* [31] tool)

The integration of the tools *Mobster*[15],*SCHISM*[13] and *LICHEe*[26] was not finished due to time restrictions. These could however be very interesting to combine/compare with the already in-place *PhyloWGS*,*DPClust* and *SciClone*, adding the element of phylogenetic comparison to the framework. Even though many of these tools try to establish similar things, it should be noted that their implementations are vastly different, this complicates their incorporation into the framework. More effort could be put into making the framework more flexible with regard to tool integration. This would greatly improve the effectiveness of the framework and would be the first priority.

Another addition to the framework could be the inclusion of CNV into the output data. This could easily be achieved by reinterpreting part of the variants simulated as CNV or by separate simulation of these kind of variants. This information should then be used to adjust the CCFs, but would require additional CNV output information for acquisition of tool input. When implemented, the effect of CNV on the tool performance could be measured, which is important because of the significant presence of these kind of variants in cancerous cells.

Apart from all this, many exploratory research is done on the therapy response of tumor with regard to clonality. Even though earlier it was thought that mainly tumor heterogeneity was responsible for relapse, because of the presence of resistant subclones. Now there are also signs of cell subpopulations showing adaptive behaviour to cancer therapies. These cells have drug-responsive mechanisms that are able to endure the treatment and 'persist'. These so-called 'persister cells' form a small subpopulation of a clonal cancer cell population and show some form of transcriptional transformation that allows them to survive. This therefore suggest that treatment is not only survived by 'well-adapted' pre-existing subclones, but is also 'persisted' by *de novo* generation of resistant subpopulations.[51] [52] This would call for a clonal analysis of tumor population on the RNA-level.

Other potential avenues of subclonal reconstruction are also being explored, such as the use of structural variant information into inference.[16] A growing trend is now the subclonal reconstruction methods focusing on the single-cell NGS data applications.[53] Tools applying this have also been developed.[20] [27] [34] Bulk samples require conversion use of VAF and other similar spectra to describe a cell population without being able to identify individual subclonal mutation signatures and inter-variant relations. Single-cell sequencing of the samples might negate this problem by direct sequencing of the individual cells. Doing this for a bigger set of cells the implications will certainly serve beneficial. The main limitation however, is its cost and consequently its difficulty of clinical application. Nonetheless, as these methods keep getting more common this could in the future be a realistic solution.

Technologies such as nanopore sequencing could also be an answer to the identification of subclonal variant signatures. This can be attributed to the immense increase in read length ($\approx 10kbp$) compared to conventional NGS methods ($\approx 300bp$). Issues of accuracy still play a role, although the concept of being able to describe individual reads to mutational signatures which can be linked to individual subclonal population seems promising.

The findings shown in this project give a hint of the complexity found within subclonal architecture reconstruction, or more generally in bioinformatics. They show that the tools used in clonality analysis have matured to a point that requires multidisciplinary expertise. Next to this, it can be seen how distinctly different algorithms are, even when based on the same principles. Knowing these differences and how to quantify them, gives additional insight in the instruments employed in cancer research. Further development of this framework and additional research into the comparison of these tools would allow for more competent tumor analysis.

Special thanks to everyone at the *CCBC* in the *ErasmusMC* for the guidance considering my lack of bioinformatic background and in general for the opportunity of doing my Bachelor end project.

References

[1] E. Friedman, K. Sakaguchi, A. E. Bale, A. Falchetti, E. Streeten, M. B. Zimering, L. S. Weinstein, W. O. McBride, Y. Nakamura, M.-L. Brandi, *et al.*, "Clonality of parathyroid tumors in familial multiple endocrine neoplasia type 1," *New England Journal of Medicine*, vol. 321, no. 4, pp. 213–218, 1989.

[2] J. Wainscoat and M. Fey, "Assessment of clonality in human tumors: a review," *Cancer Research*, vol. 50, no. 5, pp. 1355–1360, 1990.

[3] P. J. Fialkow, "Clonal origin of human tumors," *Annual review of medicine*, vol. 30, no. 1, pp. 135–143, 1979.

[4] D. Sidransky, P. Frost, A. Von Eschenbach, R. Oyasu, A. C. Preisinger, and B. Vogelstein, "Clonal origin of bladder cancer," *New England Journal of Medicine*, vol. 326, no. 11, pp. 737–740, 1992.

[5] A. L. Harris and D. E. Neal, "Bladder cancer — field versus clonal origin," *New England Journal of Medicine*, vol. 326, no. 11, p. 759–761, 1992.

[6] Ö. Acar, E. Özkurt, G. Demir, H. Saraç, C. Alkan, T. Esen, M. Somel, and N. A. Lack, "Determining the origin of synchronous multifocal bladder cancer by exome sequencing," *BMC cancer*, vol. 15, no. 1, p. 871, 2015.

[7] I. Dagogo-Jack and A. T. Shaw, "Tumour heterogeneity and resistance to cancer therapies," *Nature reviews Clinical oncology*, vol. 15, no. 2, p. 81, 2018.

[8] S. Nik-Zainal, P. Van Loo, D. C. Wedge, L. B. Alexandrov, C. D. Greenman, K. W. Lau, K. Raine, D. Jones, J. Marshall, M. Ramakrishna, *et al.*, "The life history of 21 breast cancers," *Cell*, vol. 149, no. 5, pp. 994–1007, 2012.

[9] R. Schwartz and A. A. Schäffer, "The evolution of tumour phylogenetics: principles and practice," *Nature Reviews Genetics*, vol. 18, no. 4, p. 213, 2017.

[10] A. G. Deshwar, S. Vembu, C. K. Yung, G. H. Jang, L. Stein, and Q. Morris, "Phylowgs: reconstructing subclonal composition and evolution from whole-genome sequencing of tumors," *Genome biology*, vol. 16, no. 1, p. 35, 2015.

[11] W. Jiao, S. Vembu, A. G. Deshwar, L. Stein, and Q. Morris, "Inferring clonal evolution of tumors from single nucleotide somatic mutations," *BMC bioinformatics*, vol. 15, no. 1, p. 35, 2014.

[12] C. A. Miller, B. S. White, N. D. Dees, M. Griffith, J. S. Welch, O. L. Griffith, R. Vij, M. H. Tomasson, T. A. Graubert, M. J. Walter, *et al.*, "Sciclone: inferring clonal architecture and tracking the spatial and temporal patterns of tumor evolution," *PLoS computational biology*, vol. 10, no. 8, 2014.

[13] N. Niknafs, V. Beleva-Guthrie, D. Q. Naiman, and R. Karchin, "Subclonal hierarchy inference from somatic mutations: automatic reconstruction of cancer evolutionary trees from multi-region next generation sequencing," *PLoS computational biology*, vol. 11, no. 10, 2015.

[14] A. Roth, J. Khattra, D. Yap, A. Wan, E. Laks, J. Biele, G. Ha, S. Aparicio, A. Bouchard-Côté, and S. P. Shah, "Pyclone: statistical inference of clonal population structure in cancer," *Nature methods*, vol. 11, no. 4, p. 396, 2014.

[15] G. Caravagna, T. Heide, M. Williams, L. Zapata, D. Nichol, K. Chkhaidze, W. H. Cross, G. D. Cresswell, B. Werner, A. Acar, *et al.*, "Model-based tumor subclonal reconstruction," *BioRxiv*, p. 586560, 2019.

[16] M. Cmero, K. Yuan, C. S. Ong, J. Schröder, N. M. Corcoran, T. Papenfuss, C. M. Hovens, F. Markowetz, and G. Macintyre, "Inferring structural variant cancer cell fraction," *Nature communications*, vol. 11, no. 1, pp. 1–15, 2020.

[17] M. A. Myers, G. Satas, and B. J. Raphael, "Calder: Inferring phylogenetic trees from longitudinal tumor samples," *Cell systems*, vol. 8, no. 6, pp. 514–522, 2019.

[18] M. El-Kebir, L. Oesper, H. Acheson-Field, and B. J. Raphael, "Reconstruction of clonal trees and tumor composition from multi-sample sequencing data," *Bioinformatics*, vol. 31, no. 12, pp. i62–i70, 2015.

[19] Y. Jiang, Y. Qiu, A. J. Minn, and N. R. Zhang, "Assessing intratumor heterogeneity and tracking longitudinal and spatial clonal evolutionary history by next-generation sequencing," *Proceedings of the National Academy of Sciences*, vol. 113, no. 37, pp. E5528–E5537, 2016.

[20] H. Zafar, N. Navin, K. Chen, and L. Nakhleh, "Siclonefit: Bayesian inference of population structure, genotype, and phylogeny of tumor clones from single-cell genome sequencing data," *Genome research*, vol. 29, no. 11, pp. 1847–1859, 2019.

[21] Y. Qiao, A. R. Quinlan, A. A. Jazaeri, R. G. Verhaak, D. A. Wheeler, and G. T. Marth, "Subcloneseeker: a computational framework for reconstructing tumor clone structure for cancer variant interpretation and prioritization," *Genome biology*, vol. 15, no. 8, p. 443, 2014.

[22] P. Deveau, L. Colmet Daage, D. Oldridge, V. Bernard, A. Bellini, M. Chicard, N. Clement, E. Lapouble, V. Combaret, A. Boland, *et al.*, “Quantumclone: clonal assessment of functional mutations in cancer based on a genotype-aware method for clonal reconstruction,” *Bioinformatics*, vol. 34, no. 11, pp. 1808–1816, 2018.

[23] H. Zare, J. Wang, A. Hu, K. Weber, J. Smith, D. Nickerson, C. Song, D. Witten, C. A. Blau, and W. S. Noble, “Inferring clonal composition from multiple sections of a breast cancer,” *PLoS Comput Biol*, vol. 10, no. 7, p. e1003703, 2014.

[24] M. El-Kebir, G. Satas, L. Oesper, and B. J. Raphael, “Inferring the mutational history of a tumor using multi-state perfect phylogeny mixtures,” *Cell systems*, vol. 3, no. 1, pp. 43–53, 2016.

[25] M. S. Rahman, A. E. Nicholson, and G. Haffari, “Hetfhmm: a novel approach to infer tumor heterogeneity using factorial hidden markov models,” *Journal of Computational Biology*, vol. 25, no. 2, pp. 182–193, 2018.

[26] V. Popic, R. Salari, I. Hajirasouliha, D. Kashef-Haghighi, R. B. West, and S. Batzoglou, “Fast and scalable inference of multi-sample cancer lineages,” *Genome biology*, vol. 16, no. 1, p. 91, 2015.

[27] G. Satas, S. Zaccaria, G. Mon, and B. J. Raphael, “Scarlet: Single-cell tumor phylogeny inference with copy-number constrained mutation losses,” *Cell Systems*, vol. 10, no. 4, pp. 323–332, 2020.

[28] L. K. Sundermann, “Lineage-based subclonal reconstruction of cancer samples,” 2019.

[29] L. Oesper, A. Mahmoody, and B. J. Raphael, “Theta: inferring intra-tumor heterogeneity from high-throughput dna sequencing data,” *Genome biology*, vol. 14, no. 7, p. R80, 2013.

[30] G. Ha, A. Roth, J. Khattra, J. Ho, D. Yap, L. M. Prentice, N. Melnyk, A. McPherson, A. Bashashati, E. Laks, *et al.*, “Titan: inference of copy number architectures in clonal cell populations from tumor whole-genome sequence data,” *Genome research*, vol. 24, no. 11, pp. 1881–1893, 2014.

[31] F. Strino, F. Parisi, M. Micsinai, and Y. Kluger, “Trap: a tree approach for fingerprinting subclonal tumor composition,” *Nucleic acids research*, vol. 41, no. 17, pp. e165–e165, 2013.

[32] M. M. Nieboer, L. C. Dorssers, R. Straver, L. H. Looijenga, and J. de Ridder, “Targetclone: A multi-sample approach for reconstructing subclonal evolution of tumors,” *PloS one*, vol. 13, no. 11, p. e0208002, 2018.

[33] M. J. Williams, B. Werner, T. Heide, C. Curtis, C. P. Barnes, A. Sottoriva, and T. A. Graham, “Quantification of subclonal selection in cancer from bulk sequencing data,” *Nature genetics*, vol. 50, no. 6, pp. 895–903, 2018, Github repository: `https://github.com/marcjwilliams1/SubClonalSelection.jl`.

[34] E. M. Ross and F. Markowetz, “Onconem: inferring tumor evolution from single-cell sequencing data,” *Genome biology*, vol. 17, no. 1, pp. 1–14, 2016.

[35] A. Salcedo, M. Tarabichi, S. M. G. Espiritu, A. G. Deshwar, M. David, N. M. Wilson, S. Dentro, J. A. Wintersinger, L. Y. Liu, M. Ko, *et al.*, “A community effort to create standards for evaluating tumor subclonal reconstruction,” *Nature Biotechnology*, vol. 38, no. 1, pp. 97–107, 2020.

[36] W. M. Ismail, E. Nzabarushimana, and H. Tang, “Algorithmic approaches to clonal reconstruction in heterogeneous cell populations,” *Quantitative Biology*, pp. 1–11, 2019.

[37] S. C. Dentro, D. C. Wedge, and P. Van Loo, “Principles of reconstructing the subclonal architecture of cancers,” *Cold Spring Harbor perspectives in medicine*, vol. 7, no. 8, p. a026625, 2017.

[38] J. M. Alves, T. Prieto, and D. Posada, “Multiregional tumor trees are not phylogenies,” *Trends in cancer*, vol. 3, no. 8, pp. 546–550, 2017.

[39] K. D. Siegmund, P. Marjoram, Y.-J. Woo, S. Tavaré, and D. Shibata, “Inferring clonal expansion and cancer stem cell dynamics from dna methylation patterns in colorectal cancers,” *Proceedings of the National Academy of Sciences*, vol. 106, no. 12, pp. 4828–4833, 2009.

[40] A. Shlien and D. Malkin, “Copy number variations and cancer,” *Genome medicine*, vol. 1, no. 6, p. 62, 2009.

[41] M. I. Jordan, “Dirichlet processes, chinese restaurant processes and all that.,” 2005.

[42] Y. W. Teh, “Dirichlet process.,” 2010.

[43] T. Hopper, “Notes on dirichlet processes,” 2019.

[44] E. B. Sudderth, *Graphical models for visual object recognition and tracking.* PhD thesis, Massachusetts Institute of Technology, 2006.

[45] A. D. Ewing, K. E. Houlahan, Y. Hu, K. Ellrott, C. Caloian, T. N. Yamaguchi, J. C. Bare, C. P’ng, D. Waggott, V. Y. Sabelnykova, *et al.*, “Combining tumor genome simulation with crowdsourcing to benchmark somatic single-nucleotide-variant detection,” *Nature methods*, vol. 12, no. 7, pp. 623–630, 2015.

[46] C. A. Miller, J. McMichael, H. X. Dang, C. A. Maher, L. Ding, T. J. Ley, E. R. Mardis, and R. K. Wilson, “Visualizing tumor evolution with the fishplot package for r,” *BMC genomics*, vol. 17, no. 1, p. 880, 2016.

[47] I. P. Tomlinson, M. Novelli, and W. Bodmer, “The mutation rate and cancer,” *Proceedings of the National Academy of Sciences*, vol. 93, no. 25, pp. 14800–14803, 1996.

[48] J. Kuipers, K. Jahn, B. J. Raphael, and N. Beerenwinkel, “Single-cell sequencing data reveal widespread recurrence and loss of mutational hits in the life histories of tumors,” *Genome research*, vol. 27, no. 11, pp. 1885–1894, 2017.

[49] R. Rabadan, G. Bhanot, S. Marsilio, N. Chiorazzi, L. Pasqualucci, and H. Khiabanian, “On statistical modeling of sequencing noise in high depth data to assess tumor evolution,” *Journal of statistical physics*, vol. 172, no. 1, pp. 143–155, 2018.

[50] K. Tomczak, P. Czerwińska, and M. Wiznerowicz, “The cancer genome atlas (tcga): an immeasurable source of knowledge,” *Contemporary oncology*, vol. 19, no. 1A, p. A68, 2015.

[51] M. Ramirez, S. Rajaram, R. J. Steininger, D. Osipchuk, M. A. Roth, L. S. Morinishi, L. Evans, W. Ji, C.-H. Hsu, K. Thurley, *et al.*, “Diverse drug-resistance mechanisms can emerge from drug-tolerant cancer persister cells,” *Nature communications*, vol. 7, no. 1, pp. 1–8, 2016.

[52] S. M. Shaffer, M. C. Dunagin, S. R. Torborg, E. A. Torre, B. Emert, C. Krepler, M. Beqiri, K. Sproesser, P. A. Brafford, M. Xiao, *et al.*, “Rare cell variability and drug-induced reprogramming as a mode of cancer drug resistance,” *Nature*, vol. 546, no. 7658, pp. 431–435, 2017.

[53] N. Navin, J. Kendall, J. Troge, P. Andrews, L. Rodgers, J. McIndoo, K. Cook, A. Stepansky, D. Levy, D. Esposito, *et al.*, “Tumour evolution inferred by single-cell sequencing,” *Nature*, vol. 472, no. 7341, p. 90, 2011.

7 Supplementary material

7.1 Simulation benchmarking

Here an overview of 2 simulation parameter benchmarks is shown. The chosen parameters are genome size and population size, since these have the biggest influence on the execution time of the simulation. The simulation run times shown include the time needed for writing the output data. For the highest genome size and population size values the output data covered $\approx$ 40MB in memory. Figure 18 and Figure 19 show that the simulation run time scales linearly with both parameters. A logarithmic parameter space was chosen for both parameters in the data set to illustrate a broader range of the run time magnitudes.

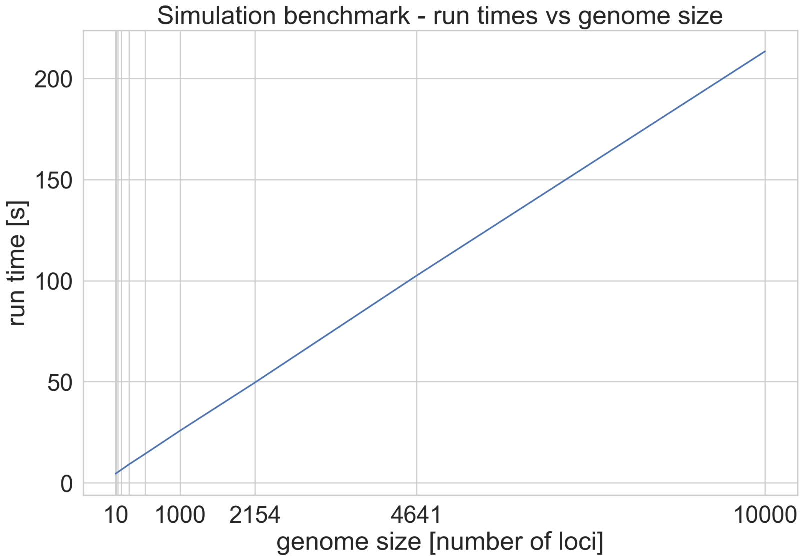


Figure 18: Data contains 50 simulations divided over: 10 genome size values and 5 population size values. Plot showing the run times for different genome sizes simulated. The genome sizes are representative of the number of variant loci analyzed during a clonality analysis.

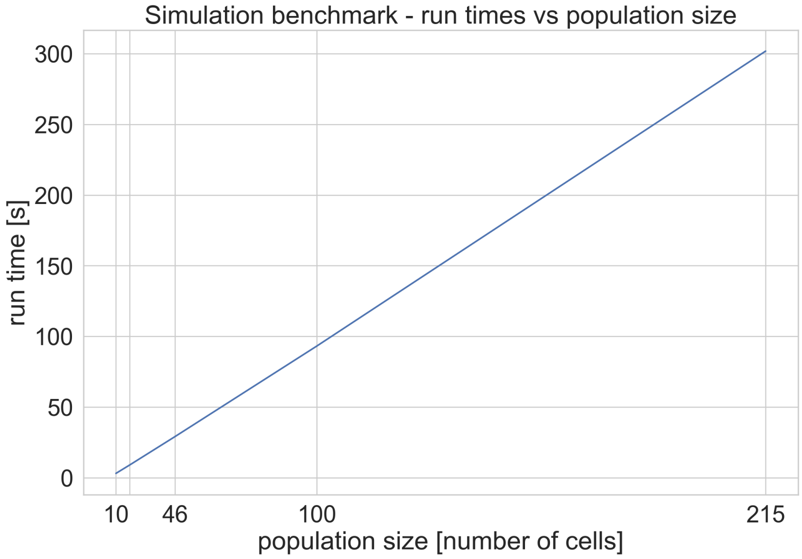


Figure 19: Data contains 50 simulations divided over: 10 genome size values and 5 population size values. This is the same data as in Figure 18, summarized over the other dimension. Plot showing the run times for different population sizes simulated. The population sizes are representative of the number of cells in the tumor sample analyzed during a clonality analysis. This group of cells can also be viewed as the representatives of a bigger tumor population.

7.2 MCMC samples analysis

An analysis was conducted to illustrate the performance of the studied qualities - correctness and subclonal fraction (CCF) error - in relation to the number of MCMC samples/iterations provided to *PhyloWGS* and *DPClust*. Since the data points produced were computationally demanding, only a single mutation load parameter value is tested (0.1). Unique to this run, the simulations were not redone for every parameter value (in this case the MCMC sample number). This was done to ensure that the exact same input was given, which makes the evaluation of the parameter space easier. This can not be done for the other simulation parameters tested, since these influence the population dynamics, thus requiring a redo of the whole simulation for every parameter value. The results shown in Figure 20 and Figure 21 are a summary of 3000 estimations ($correctness \approx 34\%$) divided over 6 MCMC sample numbers values. The data was collected using the default settings (shown in Figure 11) over 30 parallel running phylogenies. The variable values:

- mutation load - *subclonal expansion event SNV fraction*: 0.1
- phylogenies - *subclonal expansion event parents*: 6 unique phylogenies, with multiplicity of 5
- tumor samples taken - *sampling timepoints*: 2 time points (the vertical lines Figure 11)

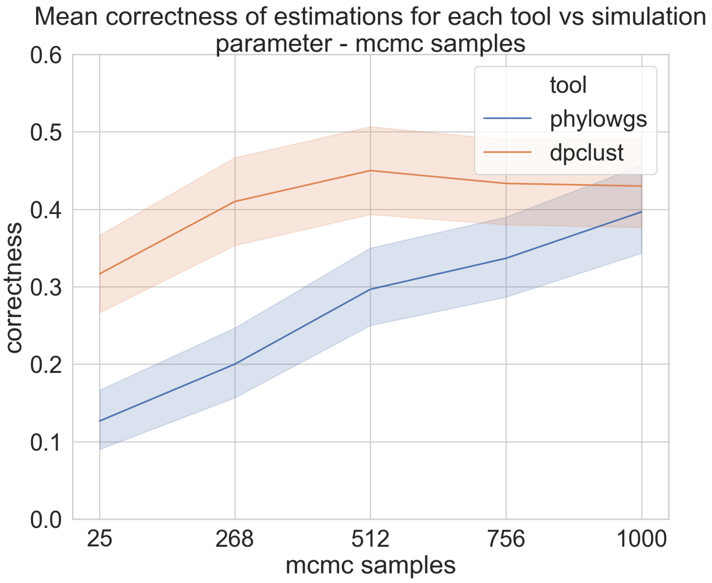


Figure 20: Plot showing the fraction of correct estimations of the number of subclones for the tools *DPClust* and *PhyloWGS* vs the number of MCMC samples/iterations. Data contains 3000 tool estimations divided over: the 2 tools, 6 unique phylogenetic trees each with a 5 times multiplicity, 2 tumor samples numbers (1 and 2), and 10 estimations. The run was conducted with a 0.1 mutation load value. The colored interval marks one standard deviation.

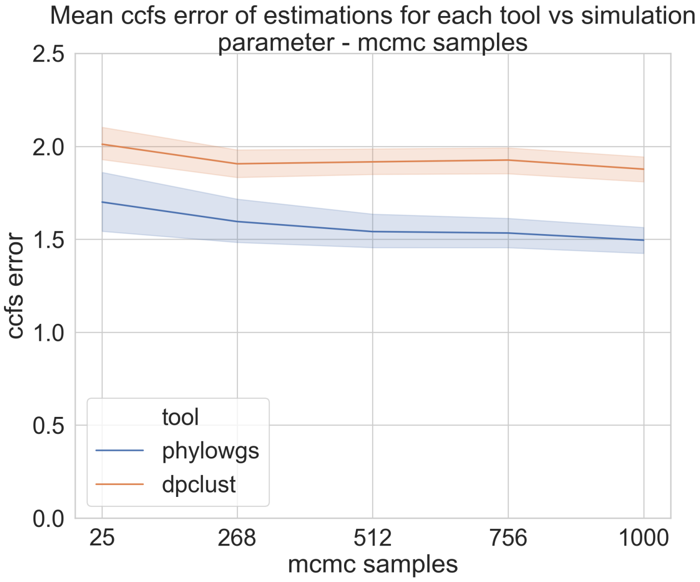


Figure 21: Plot showing the average error in the estimated subclonal fractions (CCFs) for the tools *DPClust* and *PhyloWGS* vs the number of MCMC samples/iterations. Data contains 3000 tool estimations divided over: the 2 tools, 6 unique phylogenetic trees each with a 5 times multiplicity, 2 tumor samples numbers (1 and 2), and 10 estimations. The run was conducted with a 0.1 mutation load value. The colored interval marks one standard deviation.

The results in Figure 20 showcase the intuitive increase in precision of the tool, when provided with more MCMC samples. This relation can be clearly observed with respect to the correctness. However, there is no clear relation between the error in estimated subclonal fraction (CCF) and the MCMC sample number, as seen in Figure 21. This points out that the accuracy does not increase significantly. To support this finding, the scatterplot of the same data is given in Figure 22 and Figure 23. Here the relation to the run time is also shown, which for both tools scales linearly, where *PhyloWGS* has a steeper slope than *DPClust*. The number of points for each MCMC sample number in Figure 23 and Figure 22 again indicates the correctness measure, that was shown in Figure 20.

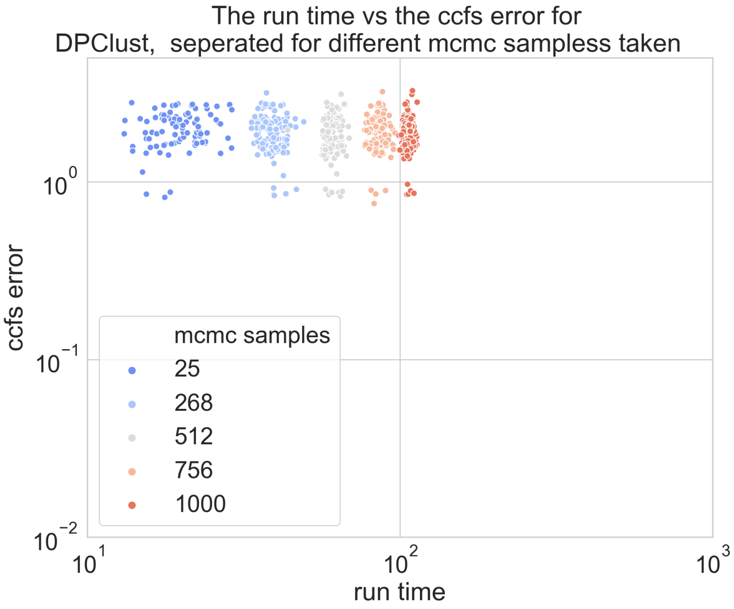


Figure 22: Scatterplot showing the correct subclone number estimations of *DPClust* in the *log* run time *log* subclonal fractions (CCFs) error space. The different numbers of MCMC samples are highlighted. This data is comprised of a total of 612 corrects subclone number estimations.

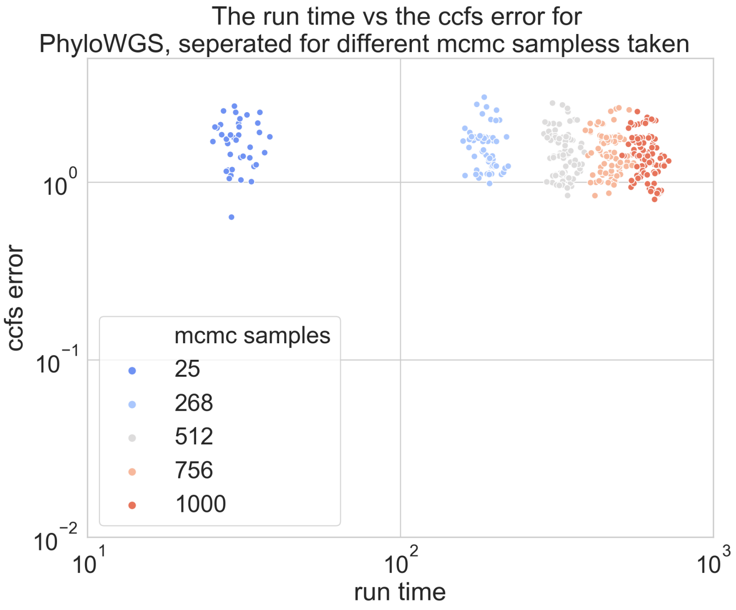


Figure 23: Scatterplot showing the correct subclone number estimations of *PhyloWGS* in the *log* run time *log* subclonal fractions (CCFs) error space. The different numbers of MCMC samples are highlighted. This data is comprised of a total of 407 corrects subclone number estimations.

7.3 Genome size analysis

For the two tools - *DPClust* and *PhyloWGS* - the genome size parameter was also explored. Here the genome size varies from 500 to 5000 inclusive (500 step size). The parameter is in actuality more related to mutation load and should be considered as such. The mutation fraction for every expansion event is fixed at 0.05. With this mutation fraction the mutation load of the respective genome sizes ranges from 25 to 250 mutations per expansion event (25 step size).

As many as 9600 estimations were carried out by each of the two tools, of which a total of 6018 correctly estimated the present subclone number ($\approx 30\%$). The data was collected in 2 separate runs, in order to validate that the run stochasticity would not have an effect on the actual result. The figures from the 2 separate runs had negligible differences, and would support the same conclusions. After this validation the data was merged and Figure 24 and Figure 25 were produced. The number of iterations given to both tools was 25 with 12 burn-in samples.

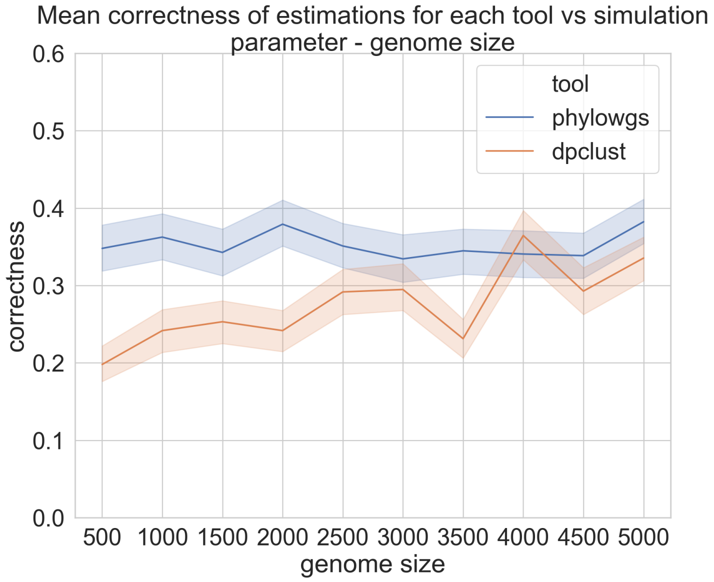


Figure 24: Plot showing the fraction of correct estimations of the number of subclones for the tools *DPClust* and *PhyloWGS* vs the genome size (mutation load). Data contains 6 different phylogenetic trees, for each 1-4 sample number, each comprised of 40 estimations. The colored interval marks one standard deviation.

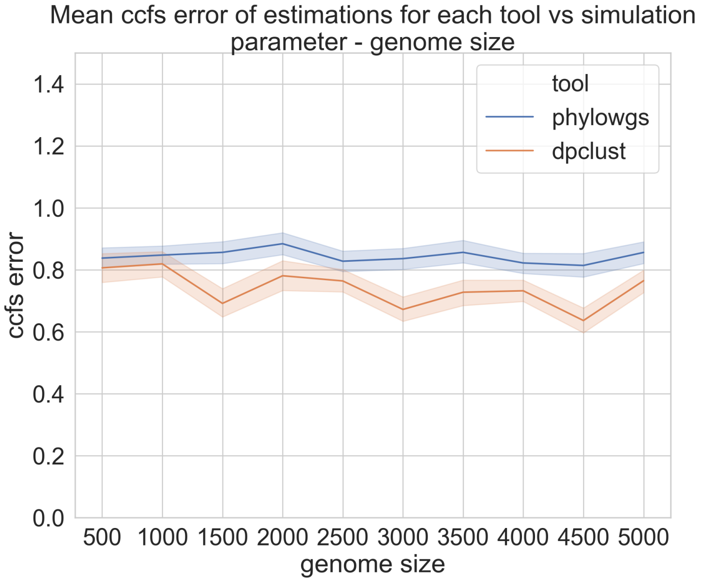


Figure 25: Plot showing the average error in the estimated subclonal fractions (CCFs) for the tools *DPClust* and *PhyloWGS* vs the genome size (mutation load). The subclonal fractions (CCFs) errors are normalized by the number of samples taken. Data contains 6 different phylogenetic trees, for each 1-4 sample number, each comprised of 40 estimations. The colored interval marks one standard deviation.

Looking at the performance of *PhyloWGS* in Figure 24 and Figure 25, notice the stable behaviour and little change in performance for different mutation loads. For *DPClust* on the other hand, notice a correlation between an increased mutation load and a higher accuracy as well as precision. This is intuitive, yet informative noticing that *DPClust* will make inarguable better estimations of the subclonal population when given more mutations. For the case of *PhyloWGS* this parameter range does not yield conclusive differences, but illustrates that the tool performs almost exchangeable within it.

7.4 Sample number - genome size analysis

A further look is taken at the two tools *DPClust* and *PhyloWGS*. The performance for different numbers of samples taken is examined in Figure 26 and Figure 27. These figures were constructed using the same data set as in subsection 7.3, consisting of 2 separate runs of 9600 estimations, of which a total of 6018 correctly estimated the present subclone number ($correctness \approx 30\%$). Within this data set the number of samples taken was the simulation parameter, covering 1,2,3 and 4 samples. The number of variants here ranges from 25 to 250 mutations per expansion event (25 step size), but are summarized, since this now is not the parameter of interest (see supplementary subsection 7.3 for the mutation load tool responses). The data set was validated by the second separate run, resulting in very similar output. Strikingly the correctness (correct estimation of number of subclones) is generally doesn't increase giving when taking additional samples. This might indicate that the fitting of the estimated subclonal prevalences over the samples actually complicates the reconstruction overall. This has to be further investigated.

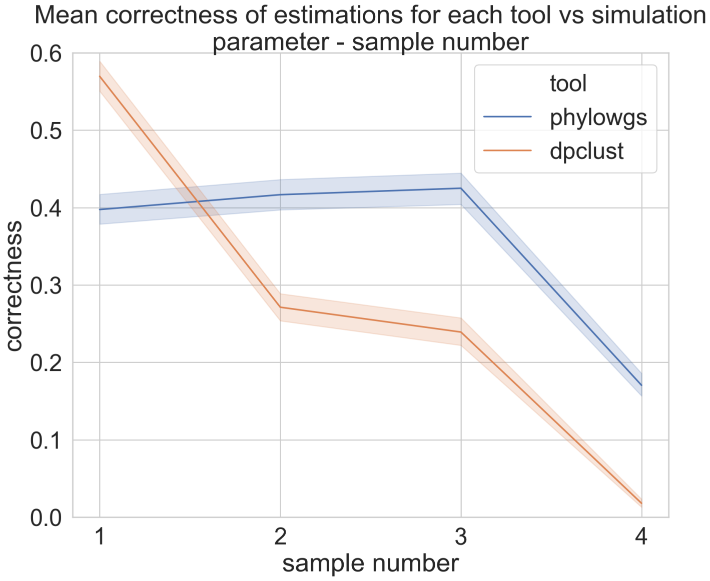


Figure 26: Plot showing the fraction of correct estimations of the number of subclones for the tools *DPClust* and *PhyloWGS* vs the number of samples taken. Data contains 6 different phylogenetic trees, for each 500-5000 genome sizes, each comprised of 40 estimations. The colored interval marks one standard deviation.

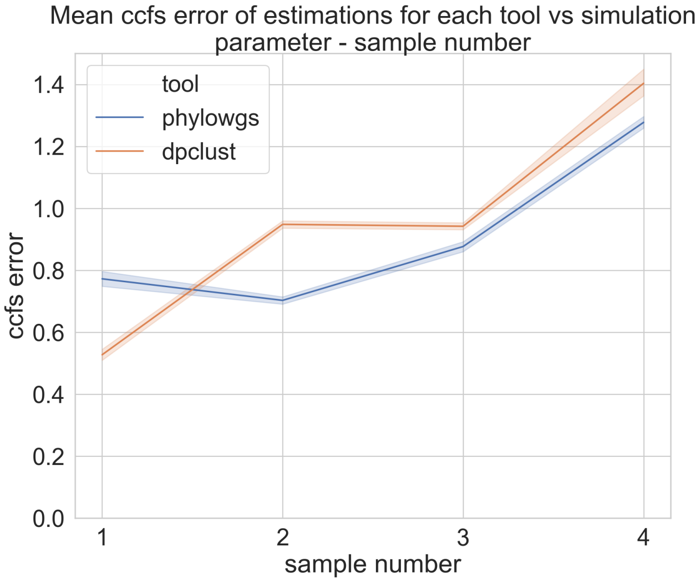


Figure 27: Plot showing the average error in the estimated subclonal fractions (CCFs) for the tools *DPClust* and *PhyloWGS* vs the number of samples taken. The errors are normalized by the number of samples taken. Data contains 6 different phylogenetic trees, for each 500-5000 genome sizes, each comprised of 40 estimations. The colored interval marks one standard deviation.

To support the graphs in Figure 26 and Figure 27 and show the differences between the execution times of *PhyloWGS* and *DPClust* a plot of all the correct estimations of each tool is made. Figure 29 and Figure 28 illustrate the performance of a tool while also capturing the computation time needed to make the estimation. Here we can again clearly correlate the sample number increase with the subclonal fractions estimation error increase. A notable contrast between the two tools is also seen in respect to their run time: clear separation of estimations for district sample numbers is observed for *DPClust*. *PhyloWGS* on the other hand actually has a different run time to sample number relation and shows no such definite separation.

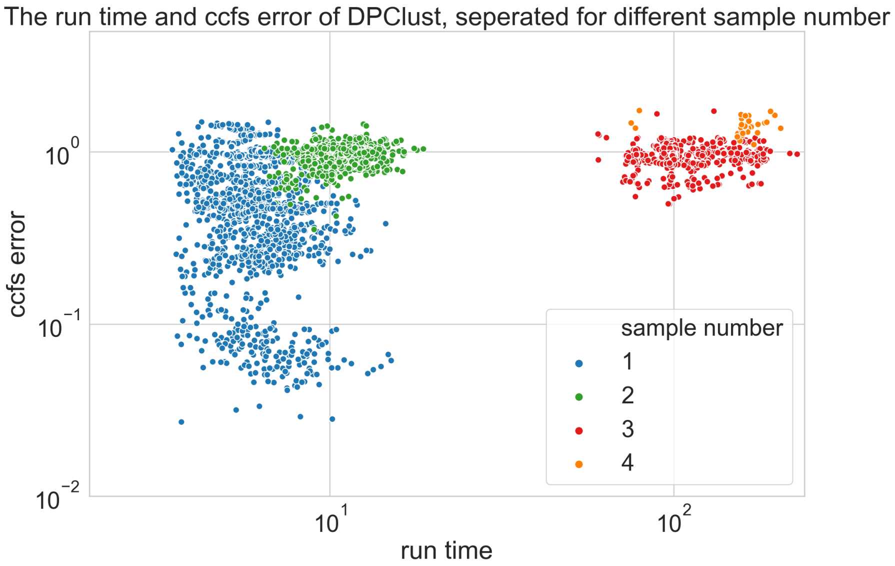


Figure 28: Scatterplot showing the correct subclone number estimations of *DPClust* in the *log* run time *log* subclonal fractions (CCFs) error space. The different number of samples taken are highlighted. This data is comprised of a total of 2635 corrects subclone number estimations.

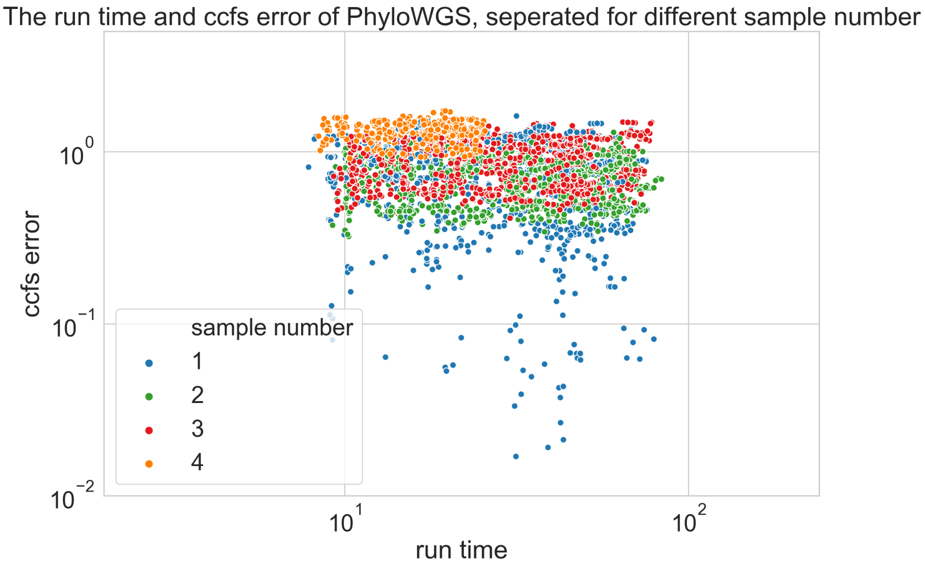


Figure 29: Scatterplot showing the correct subclone number estimations of *PhyloWGS* in the *log* run time *log* subclonal fractions (CCFs) error space. The different number of samples taken are highlighted. This data is comprised of a total of 3383 corrects subclone number estimations.

7.5 Sample number - MCMC samples analysis

This section shows whether the estimations improve, in the case where we scale the number of MCMC samples together with the number of tumor samples taken. The range of tumor samples taken is 1,2,3 and 4, which where given 25,50,75 and 100 MCMC samples, respectively. The results of this run are shown in Figure 30.

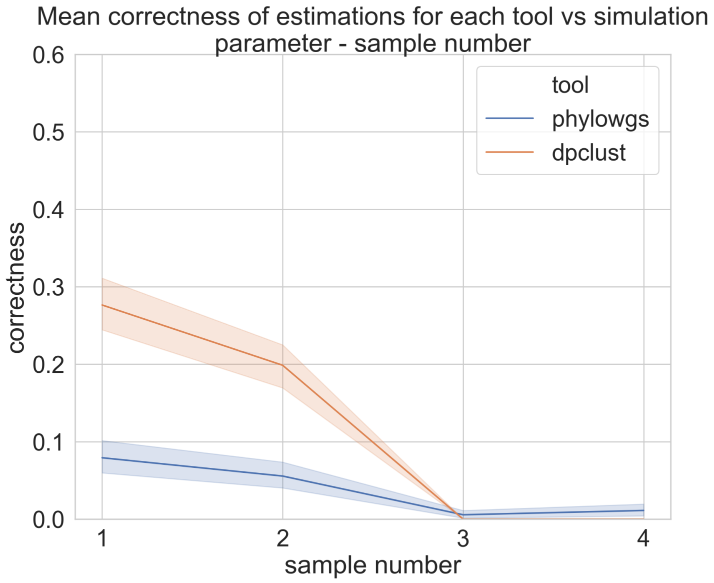


Figure 30: Data contains 5760 tool estimations divided over: 6 unique phylogenetic trees, 4 mutation load values (0.001 0.101 0.201 0.301), 2 tools (*DPClust* and *PhyloWGS*), 30 estimations and 4 sample numbers. Plot showing the fraction of correct estimations of the number of subclones vs the number of samples taken. The colored interval marks one standard deviation.

These results indicate that the tools do not necessarily perform better when the number of MCMC samples is corrected for the number of tumor samples taken. To elaborate on the noticeable difference between Figure 14a and Figure 30, it has to be kept in mind that the former had more mid-ranged mutation load values compared to the latter. This difference in mutation load representatives therefore shifts the correctness down, since mid-ranged mutation load values actually increase the average score of a sample number value.

7.6 Patient 4 and 8 figures

Here the results for patient 4 and 8 are shown.

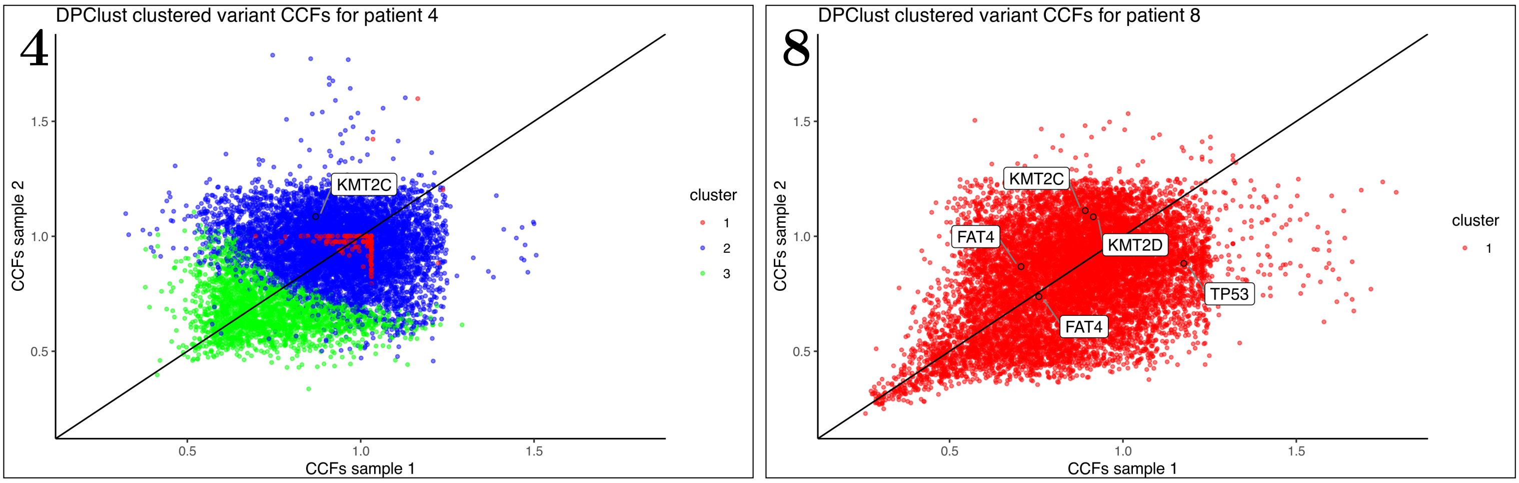


Figure 31: The DPClust clustering assignments of the patient data. The two panels corresponds to patients 4 and one of the 8 patients as indicated in their top-left corner. Within every panel, an individual point represents a mutation VAF measurement, which was transformed to CCFs using the CN information. The horizontal axes indicate this CCF value for the sample in the first biopsy taken, and the vertical corresponds to the second biopsy. The cluster assignment - shown in different colors - was achieved with 375 MCMC samples (and 125 burn-in) of the DPClust tool.

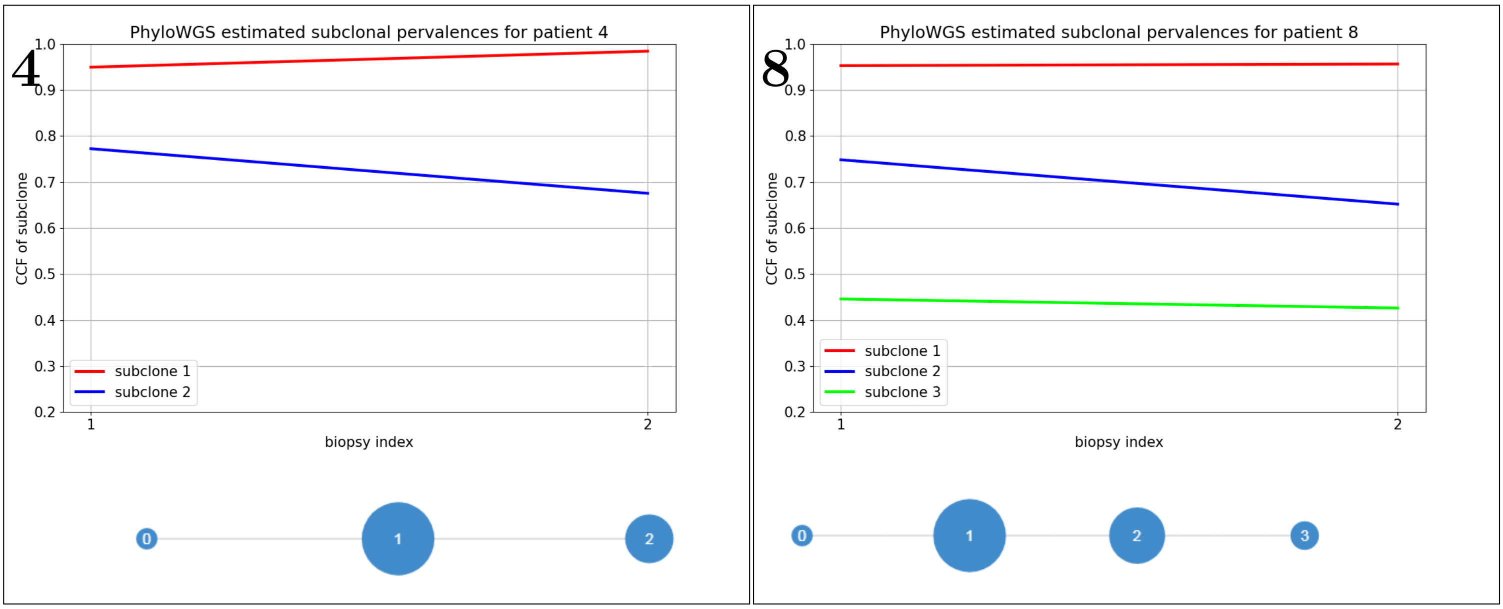


Figure 32: The PhyloWGS's subclonal fractions (CCFs) of its estimated subclonal composition with phylogenetic tree of the patient data. The two panels corresponds to patients 4 and one of the 8 patients as indicated in their top-left corner. Within every panel, the subclonal fractions (CCFs) of all the subclones is shown for both biopsies. Below this plot a graph of the phylogenetic relationship between the subclones is shown (the ”0” population is not actually present in the sample, but represents the founding cell tissue from which the subclones stem). Each subclone is shown in a different color, which corresponds to the DPClust estimation if possible. The shown results were achieved with 375 MCMC samples (and 125 burn-in) of the PhyloWGS tool (over 10 separate threads).

7.7 Manual

This is the manual for the framework that was developed for the assessment of subclonal architecture reconstruction tools. The following will explain how to use an apply elements of the framework to do analysis. The actual code can be found in the *Bitbucket* of *CCBC* and on *GitHub*. Link to the repository: https://bitbucket.org/ccbc/subclonal-reconstruction/src/master/. For more information e-mail: burghoornniels@gmail.com.

7.7.1 Interface

The framework is operated from the Unix command line interface (CLI) on a high preformance cluster (HPC).

The main scripts that have to be considered by the user are the **run.sh**, **ui.py** and the **stats_files/statsrun_script.sh**. The former is used to do simulations of tumor populations separately, while the latter is used to look at tool output in combination with simulated data. Both these scripts can to be used in combination with the settings file **settings.py**. This file contains all the important setting simulation parameters.

Note that the usage of the *Bash* files might require a modification of the permissions of the files. To do this use the command `chmod u+x [bash file]`.

7.7.2 run.sh

The **run.sh** bash file has to be used in combination with the **settings.py** file in order to simulate the desired tumor population dynamics. The output of running this script is found in the **output** folder inside the main folder. The output of this script contains elements of:

- **log.txt**: a log of the simulated population.
- **prevalences.txt**: an overview of the subclonal fractions that have been tracked during the simulation.
- **fishPlot.png**: a fishplot showing the overall subclonal architecture of the tumor.
- **reads_data**[_x]: the mutational signatures of the tumor population samples in binary format.
- **density**[_x]: the CCF spectra.
- **settings.txt**: a copy of the **settings.py** file in txt format at the time of the run.
- **simulation_report.pdf**: an overview of the fishplot and the density plots in a pdf report.

The **log.txt** file can be made as detailed as desired by tweaking the *general* dictionary object in the **settings.py**. The **DEBUG** variable should be used to turn on both *DEBUG_READS* and *DEBUG_PHYLO*. The *DEBUG_READS* makes sure that the *log.txt* shows a visual representation of the mutations found each of the cells in the simulated tumor, for each time step simulated. The *DEBUG_PHYLO* makes sure that the *log.txt* contains a visual representation of the phylogenetic relationships between each of the cells in the simulated tumor population throughout time.

The number of **reads_data**[_x] and **density**[_x] is dependent on the number of tumor samples taken that was specified by the user in the **settings.py** file. Here the "[_x]" should be replaced with the actual number of the sample, starting from zero. The last sample is never indexed and does not have a suffix "_".

The commands used to operate the **run.sh**, should be run from the main folder and are:

- `run.sh init`: removes the txt files in the output folder and creates a copy of the **settings.py** in the output folder.
- `run.sh`: does `run.sh init` and runs the simulation.
- `run.sh fish`: creates a fishplot from the **prevalences.txt**
- `run.sh density`: creates the CCF density spectra plots from (all) the **reads_data[_x].txt** files
- `run.sh simtest`: exports the simulation results (whole output folder) to a folder next to the main folder called **simtests** (this has to be manually made beforehand) and names it **simtest_[x]**, where x is the number of files found in the **simetest** folder.
- `run.sh pwgs`: this clears the txt files in the output, runs the simulation and executes PhyloWGS to estimate.
- `run.sh report`: makes the **simulation_report.pdf** file from the from the **prevalences.txt** and **reads_data**[_x]**.txt** files.

7.7.3 ui.py

The **ui.py** file is executed using *Python 3* and can be used in combination with the **settings.py** file for easy determination of simulation parameters. The most essential simulation parameters can also be manipulated in the CLI. Next to this the CLI of **ui.py** needs to be used in order to assign a variable or parameter space that is desired to be studied. There are numerous options that can be used in combination with the of **ui.py** command. The simulation parameters that can be indicated in the CLI are ready to be studied for a whole parameter range. The parameters that are not contained in the CLI options have to be manually defined in the **setting.py** file. Similar to running **run.sh**, within this file the dictionary object *rrg* can be manipulated to simulate the desired tumor dynamics. However, in the case of doing a study with **ui.py** priority is given to the dictionary object *default_rrg*, since these values can be variables during the analysis. Manually editing the *rrg* values that have an extra indentation therefore does not change the **ui.py** study and should instead be altered in the *default_rrg* dictionary object.

The CLI options contain:

- `python3 ui.py -p [tool name or tool indicators]`: is used to assign the tools - to be studied - to the analysis. CANNOT BE LEFT UNSPECIFIED. If multiple tools are chosen they have to be separated by comma's. Both indicators and the full capitalized name of the tools can be used to refer to the tools.

  Example:
  `python3 ui.py -p pwgs,dpc` and `python3 ui.py -p PhyloWGS,DPClust` can both be used to indicate the use of the *PhyloWGS* and *DPClust* tools.

- `python3 ui.py -l [number of simulations (loops)]`: is used to assign the number of simulations/phylogenies to be studied in the analysis. THIS IS RAN IN PARALLEL THUS REQUIRES THE SAME NUMBER OF CPU'S. A single value can be provided, although without specification is 1 by default. The number specified can be used in combination with the `-r` option to indicte the number of phylogenies desired. If 3 subclones are simulated and more than 6 phylogenies are indicated the phylogenies are repeated.

  Example:
  `python3 ui.py -l 6` or `python3 ui.py -l 30` can both be used to indicate a different number of simulations.

- `python3 ui.py -s [number of estimations of tool per simulation (subloops)]`: is used to assign the number of estimations for each simulation in the analysis. THIS INCREASES THE ANALYSIS ACCURACY AND RUN TIME. A single value can be provided, although without specification is 1 by default.

  Example:
  `python3 ui.py -s 10` or `python3 ui.py -s 40` can both be used to indicate a different number of estimations of each tool per simulation.

- `python3 ui.py -r [1 - for 3 subclones simulated, 1,2 or 3 - for 4 subclones simulated]`: is used to assign the phylogenetic relations to each simulation in the analysis. WITHOUT THIS THE SIMULATION HAVE RANDOM PHYLOGENETIC RELATION. A single value can be provided. For 3 subclones 1 is the only acceptable set of 6 phylogenies. For 4 subclones the set of 24 phylogenies is divided into 3 groups of 8: the "least linear" group 1, group 2 and the "most linear" group 3. The acceptable values are therefore only 1,2 or 3 for 4 subclones simulated. This settings has to be used in combination with the `-l` command to supply the required number of simulations paired to the defined number of phylogenies.

  Example:
  `python3 ui.py -l 6 -r 1` for the 6 phylogenies for 3 subclones or `python3 ui.py -l 8 -r 3` can both be used to indicate fixed phylogeny for each simulation index.

- `python3 ui.py -i 1`: is used to fix the simulation seeds for each simulation in the analysis. THIS ONLY WORKS WITH THE SEQUENCING NOISE OR MCMC SAMPLE PARAMETERS. A 1 has to be provided to turn the feature on.

  Example:
  `python3 ui.py -x -i 10` or `python3 ui.py -s 40` can both be used to indicate a different number of estimations of each tool per simulation.

- `python3 ui.py -g [genome size(s)]`: is used to manipulate the simulation parameter ”genome size” (*number of loci*). Use colons (’:’) to indicate a range of values in the format [start value:steps:end value].

  Example:
  `python3 ui.py -g 1500` for a single parameter and `python3 ui.py -g 1000:3:1500` to do an analysis for each of the parameter values $1000, 1250$ and 1500.

- `python3 ui.py -m [mutation load(s)]`: is used to manipulate the simulation parameter ”mutation load” (*subclonal expansion event SNV fraction*). Use colons (’:’) to indicate a range of values in the format [start value:steps:end value].

  Example:
  `python3 ui.py -m 0.25` for a single parameter and `python3 ui.py -m 0.01:5:0.09` to do an analysis for each of the parameter values $0.01, 0.03, 0.05, 0.07$ and 0.09.

- `python3 ui.py -n [tumor sample number value(s)]`: is used to manipulate the simulation parameter ”tumor sample number” (*sampling timepoints*). The samples will be regularly spaced automatically. Use colons (’:’) to indicate a range of values in the format [start value:steps:end value].

  Example:
  `python3 ui.py -n 2` for a single parameter and `python3 ui.py -n 1:2:2` to do an analysis for each of the parameter values 1 and 2.

- `python3 ui.py -c [subclone population fraction size value(s)]`: is used to manipulate the simulation parameter ”subclone size” (*subclonal expansion event population fraction*). Use colons (’:’) to indicate a range of values in the format [start value:steps:end value].

  Example:
  `python3 ui.py -c 0.4` for a single parameter and `python3 ui.py -c 0.2:3:0.4` to do an analysis for each of the parameter values $0.2, 0.3$ and 0.4.

- `python3 ui.py -w [subclone population fraction standard deviation value(s)]`: is used to manipulate the simulation parameter ”subclone standard deviation” (*subclonal expansion event population fraction stdv*). Use colons (’:’) to indicate a range of values in the format [start value:steps:end value].

  Example:
  `python3 ui.py -w 0.05` for a single parameter and `python3 ui.py -g 0.03:2:0.05` to do an analysis for each of the parameter values 0.03 and 0.05.

- `python3 ui.py -q [sequencing noise value(s) (gaussian blur standard deviation)]`: is used to manipulate the simulation parameter ”sequencing noise” (*sequencing noise*). The samples will be regularly spaced automatically. Use colons (’:’) to indicate a range of values in the format [start value:steps:end value].

  Example:
  `python3 ui.py -q 0.05` for a single parameter and `python3 ui.py -n 0.02:4:0.08` to do an analysis for each of the parameter values $0.02, 0.04, 0.06$ and 0.08.

- `python3 ui.py -x [MCMC samples/iterations number(s)]`: is used to manipulate the parameter "MCMC samples". Use colons (':') to indicate a range of values in the format [start value:steps:end value].

  Example:
  `python3 ui.py -x 25` for a single parameter value and `python3 ui.py -x 25:4:1000` to do an analysis for each of the parameter values 25, 350, 675 and 1000.

Any combination of these can be used to assemble a composition of parameters designed to simulate the required tumor population. The only two parameters that can be used as variables and are separate from the simulation are "sequencing noise" and "MCMC samples". These can therefore be used in combination with fixed tumor dynamics (fixed seed), by indicating the "identical" `-i` option.

The output of the analysis conducted with **ui.py** consists of the following elements:

- **visual_report.pdf** containing all simulation report pages and an overview of the tool-vs-correctness page at the end.
- **settings.txt**: a copy of the **settings.py** file in txt format at the time of the run.
- **log.txt**: a log of the analysis conducted.
- **pipe_estimations.npy**: the *Numpy* data file containing a data frame of the estimation results of the analysis. The data structure of this file is explained in the report Figure 9 (Data structure), but this file only contains the "estimated" data points.
- **true_values.npy**: the *Numpy* data file containing a data frame of the simulation results of the analysis, serving as the ground truth. The data structure of this file is explained in the report Figure 9 (Data structure), but this file only contains the "true values".

Using the *Pandas* and *Seaborn* packages in *Python 3* the *Numpy* data files can be summarized and plotted for various dimensions of this data.

7.7.4 stats_files/statsrun_script.sh

For even further automation, the **stats_files/statsrun_script.sh** was developed, in here multiple **ui.py** studies can be executed consecutively. The format for this execution is:

```
stats_run \
"python3 ${rrg_path}ui.py [options]" \
[analysis name]
```

This has to be added below the *stats_run* function.

Example:

```
stats_run \
"python3 ${rrg_path}ui.py -p pwgs,dpc -r 1 -s 30 -l 6 -m 0.001:6:0.301 -n 1" \
SAMPLE_MUTLOAD_s1
stats_run \
"python3 ${rrg_path}ui.py -p pwgs,dpc -r 1 -s 30 -l 6 -m 0.001:6:0.301 -n 2" \
SAMPLE_MUTLOAD_s2
stats_run \
"python3 ${rrg_path}ui.py -p pwgs,dpc -r 1 -s 30 -l 6 -m 0.001:6:0.301 -n 3" \
SAMPLE_MUTLOAD_s3
stats_run \
"python3 ${rrg_path}ui.py -p pwgs,dpc -r 1 -s 30 -l 6 -m 0.001:6:0.301 -n 4" \
SAMPLE_MUTLOAD_s4
```

The analysis done with this script are automatically transported to a folder next to the main folder called **stats_data**, which has to be defined beforehand. These resulting data frames in the analysis folders found here can then be merged to add an extra dimension to your data that can be summarized or studied.