PyroTimer is a performant tool to time copy number gains

Yano Wilke
27-06-2022

Layman summary

Throughout our life mutations, small changes to our DNA, accumulate in each cell and are passed on to the daughter cells. Most of these mutations will not have any impact on the functioning of the cell, these are called passenger mutations. However, sometimes a mutation can be a driver mutation. Driver mutations accelerate cell growth, division and cause cells to become cancer cells. A tumour starts from a single cell that acquired one or more driver mutations. As the tumour grows new mutations will continue to accumulate. These mutations will only be present in the descendent cells of cell where the mutation happened. After some time, a tumour will consist of several different subpopulation of cells with their own unique mutations. Understanding at what point in time driver mutations happen provides valuable information on how cancers arise. However, this is a difficult challenge since the sequencing data from a tissue sample taken at diagnosis shows all mutations currently present but does not reveal when they happened. One approach is to take multiple samples from the tumour at different points in time to keep track of new mutations. However, the limited number of samples also leads to a limited resolution in timing mutations. Interestingly, recent studies have presented a new method to time a specific type of mutations from only a single sample. This method works for a mutation where a sizable genetic region gets duplicated, including all the small mutations that are present in that region. The basic concept is as follows, when a small mutation occurs it is only present in once place in the genome. If a duplication of a genetic region takes place all mutations already present will also be duplicated and are now present at two locations in the genome. All mutations that occur after the duplication will only be present in one place. It is possible to count how many mutations occurred before and after the duplication. Mutations happen at a fairly stead rate over time and the number of mutations that accumulated can thus be translated to how much time has passed. This creates the opportunity to order driver mutations that initiated the cancer in time. To achieve this, we present a statistical model to time duplications called PyroTimer. We implement the model in a powerful, flexible and fast probabilistic programming framework called Pyro. We evaluate two different implementations of the model and also evaluate two different inference algorithms. To evaluate the accuracy of the model we simulate realistic data where we know the correct answer. We also compare the performance on real world data and compare it to an already existing implementation for timing duplications. We find that PyroTimer can accurately time duplications and runs quickly on large datasets. When used on real world data it seems that the estimation of the sample purity can be off in some cases which affects the ability to accurately time duplications.

Abstract

A tumour starts from a single precursor cell. During the evolution of the tumour, somatic SNVs and copy number alterations accumulate. At the point of diagnosis, the tumour often consists of multiple cell lineages with their own unique mutations these lineages are called subclones. A sample taken at diagnosis provides a snapshot in time of the current state of a tumour. We can reconstruct the clonal architecture of a tumour and express the size of the subclones as their cancer cell fraction (CCF). Besides spatial information we can also infer the temporal ordering of SNVs and copy number gains. The concept relies on identifying mutations that were present before the duplication and thus were also duplicated. Mutations after the duplication will not be duplicated. Using clock-like mutational processes we can time when a duplication occurred, from the very first cell division. Here we present PyroTimer, a Bayesian mixture model to time copy number gains. We implement it with a flexible and scalable probabilistic programming framework named Pyro. Using realistic simulated data, we validate the performance of PyroTimer using Hamiltonian Monte Carlo and Stochastic Variational Inference as inference algorithms. In the end we compare the results on real world data from PCAWG and provide a comparison between the current MutationTimeR implementation in R. We show that PyroTimer is a high performance and functional tool to time copy number gains.

Contents

Introduction

Somatic mutations accumulate in the human genome from the first cell division until death. These mutations originate from endogenous as well as exogenous processes. The vast majority of mutations will be selectively neutral passenger mutations. However, some will be driver mutations that increase proliferation and lead to tumorigenesis.

Since a tumour originates from a single precursor cell, the somatic mutations from the precursor cell will be present in all tumour cells. During the evolution of a tumour, single nucleotide variants (SNVs), copy number alterations (CNAs) and other types of mutations constantly create new lineages. Through somatic evolution some lineages will disappear while a lineage with a beneficial mutation will persist and expand (Figure 1). In the end a tumour will contain multiple different lineages called subclones that will carry mutations not shared with the other subclones. These subclonal mutations are only present in a proportion of the tumour cells which is expressed as the cancer cell fraction (CCF).

Sequencing data from a biopsy taken at diagnosis provides a snapshot in time of a tumour. Due to the limited sequencing depth with bulk sequencing approaches, mutations that are only present in a small fraction of the cells cannot be detected. The lower bound of detection depending on the sensitivity of the sequencing approach used. Therefore, only subclones with a higher CCF can be detected. Previous research outlines how the clonal architecture of a tumour can be reconstructed by using SNVs[1].

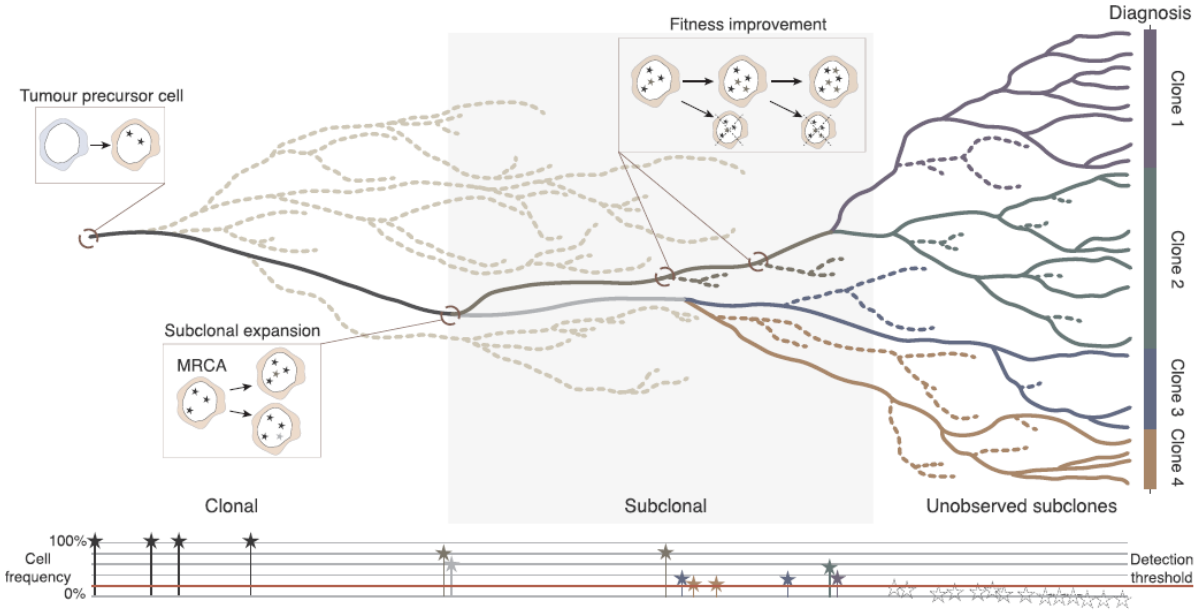

Figure 1. Evolution of a tumour. A driver mutation that increases proliferation forms a tumour precursor cell. Different cell lineages emerge and disappear during the somatic evolution of the tumour. At some point in time a lineage will persist and create a subclone. This subclonal expansion happens at the Most Recent Common Ancestor (MRCA) cell which is the last cell that all tumour cells originate from. Mutation that occur before the subclonal expansion are present in all tumour cells, also called clonal mutations. Subclones make up a fraction of the tumour cells expressed as the cancer cell fraction (CCF). Mutations with a low CCF cannot be detected by bulk sequencing. Figure take from[2]

Besides spatial insight into tumours there is a great research interest in the temporal ordering of mutations. Existing methods for temporal ordering of mutations relied on multiple samples taken from an individual tumour at different points in time[3]. Nevertheless, the temporal resolution and range is restricted by the limited number of samples. However recent studies present a new method to infer the

timing of duplications and provide temporal ordering for SNVs from a single sequencing sample. The concept of timing duplications relies on establishing the multiplicity of a mutation with a statistical model. The multiplicity of a mutation is the number of copies that are present in the genome.
The multiplicity of a de novo SNV is one but can be altered by copy number changes. When a duplication of a genetic region occurs, all SNVs present in that region before the mutation will also be duplicated and their multiplicity will increase. However, SNVs that arise after the duplication will have a multiplicity of one. A distinction can thus be made between SNVs that occurred before and after a duplication based on their multiplicity (Figure 2). The number of mutations for each multiplicity can be utilized as a proxy for time passed before and after a duplication[4]. Somatic mutations accumulate linearly with age in adult tissues, but elevated mutation rates are observed before birth[5]. Mutational signature analysis shows that single base substitution signature 1 is better suited for the use as a mutational clock since the rate stays consistent throughout life and is present in all tissues[5]. Single base substitution signature 1 is thought to be the result of spontaneous or enzymatic deamination of 5-methylcytosine to thymine. The time a duplication occurred can be expressed as molecular time which is a relative time based on the total number of clonal mutations. Molecular time starts at 0 with the first cell division of the zygote. It ends at 1 with the creation of the most recent common ancestor (MRCA) cell of the tumour since mutations after that point will no longer be clonal. Recent studies successfully applied this concept to create a temporal ordering of driver events in cancer[6–8]. The currently available implementation in R named MutationTimeR provides a statistical model for inferring mutation timing using an expectation–maximization algorithm[9]. However, it uses point estimates from external tools for certain input parameters which does not capture the uncertainty about these parameters. Here we present an updated implementation named PyroTimer which is built with the flexible and scalable probabilistic programming framework called Pyro[10]. It uses a Bayesian mixture model that captures the uncertainty of all parameters. With simulated data we evaluate the performance of two different Python libraries that use the Pyro framework and two different inference algorithms. Finally, we compare MutationTimeR and PyroTimer on PCAWG data.

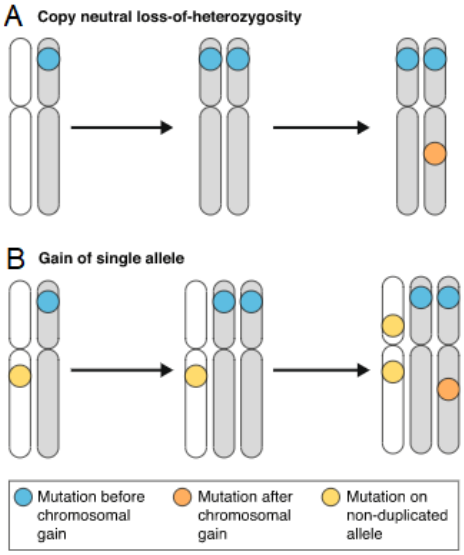

Figure 2: Identify mutations that occur before or after a duplication. A. A copy neutral loss of heterozygosity (CN-LOH) where one allele is duplicated and the other is lost. Mutations present before the duplication are also duplicated. Mutations that occur after the duplication accumulate on both alleles but only have one copy. B. A gain of a single allele. One allele is duplicated and the other allele remains. Mutations on the allele that duplicates will also duplicate. Mutations on the non-duplicated allele and mutations that occur after the duplication will only have a single copy. Figure take from [6]

The model

We present a Bayesian inference model to estimate the molecular time *T* of a copy number gain. Each sequenced sample might contain multiple gains that can be timed. Each gain can contain 0 or more clonal SNVs that occurred before or after the gain. *When* we assume the mutation rate is constant throughout time and with the number of clonal SNVs as *N* that occurred per copy before and after the gain, we can calculate *T* as follows; $T = N_{before} / (N_{before} + N_{after})$. To calculate *T* for specific a copy number gain configurations we use n_m which is the number of clonal SNVs with the multiplicity *m*. The simplest copy number gain to time is copy neutral loss of heterozygosity (CN-LOH) where one allele is duplicated and the other is lost (Figure 2a). For CN-LOH, N_{before} is the number of clonal SNVs with a multiplicity of 2 also written as n_2. N_{after} is the number of clonal SNVs with a multiplicity of 1 divided by number of alleles present after the CN-LOH also written as $n_1 / 2$. A CN-LOH event can be timed as follows; $T = n_2 / (n_2 + (n_1 / 2))$. Timing a single gain where there are 2 copies of the major allele and 1 copy of the minor allele is more complex to time (Figure 2b). N_{before} is still resented by n_2. To calculate N_{after} we must account for SNVs that were already present before the gain on the allele that was not duplicated and the number of alleles after duplication, it can be written as $(n_1 - n_2) / 3$. For a single gain *T* can be obtained as followed; $T = n_2 / (n_2 + (n_1 - n_2) / 3)$. When the multiplicity of the clonal SNVs is known, calculating *T* is straightforward however this cannot be directly observed from the sequencing data. It is required to infer the multiplicity and whether the SNV is clonal or subclonal. This can be achieved by evaluating the observed VAF of the SNV against the expected VAF. The expected VAF depends on the number of alleles carrying the mutation and can be calculated as follows; $E(VAF) = m*CCF*p / (N *(1-p) + C*p)$, in which *p* is the sample purity, N is the normal copy number, C is the tumour copy number. Copy number analysis data provides *N* and *C*. For each mutation three different states can be inferred by its VAF. For each state there is prior knowledge about the multiplicity and CCF.

1. Subclonal state; CCF < 1. Multiplicity is one assuming no subclonal copy number changes.
2. Early clonal state; CCF = 1. Multiplicity is two because of the duplication.
3. Late clonal state; CCF = 1. Multiplicity is one since it occurred after the duplication.

The model contains the following latent variables. For each sample a sample purity is estimated. Each sample might contain 0 or more subclones. The number of subclones is given by the tool DPClust[11]. For each clone the CCF and the proportion of mutations carried by that clone is estimated. A sample might contain one or more copy number gained segments to time. For each segment the molecular time *T* is estimated. For all parameters we start with a uniform prior. As observations we provide for each copy number gained segment the tumour copy number, normal copy number and for each SNV the alternate read count and coverage. The likelihood of all latent variables for each observation results in the final posterior distribution for each latent variable (Figure 3). The plate notation and steps of the mixture model can be seen in Figure 4.

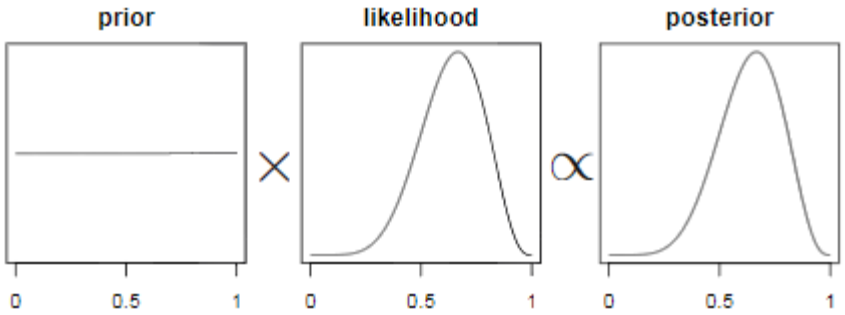

Figure 3: A Bayesian statistical model. When estimating a parameter, you start with prior distribution which you can use to incorporate prior knowledge. Based on an observation you calculate the likelihood for that observation for each possible value of the parameter. By multiplying the prior and likelihood distribution you obtain the posterior distribution. The posterior distribution can be used as the prior distribution for the next observation to update the posterior distribution. Figure take from [13]

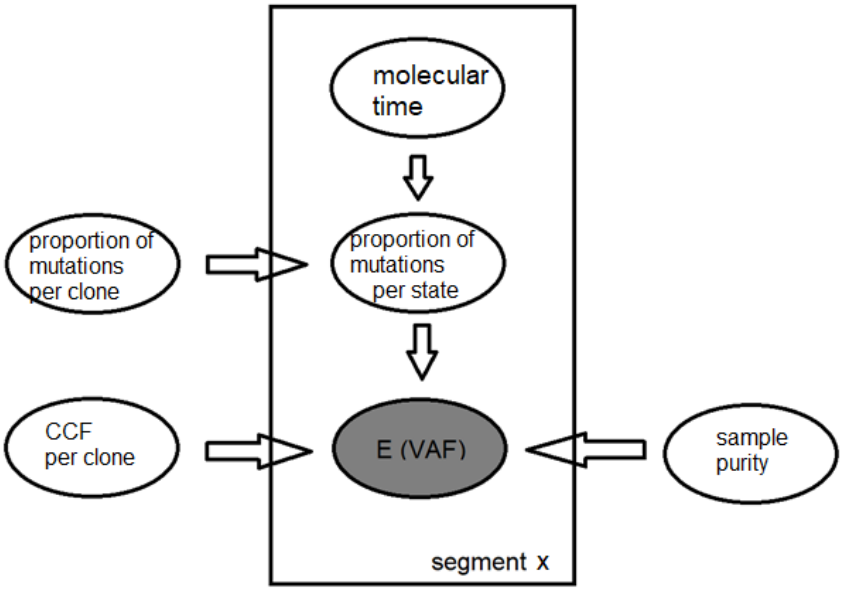

Figure 4: Plate notation of the PyroTimer Bayesian mixture model. Variables outside the plate are for all segments of that sample. Variables inside the plate are conditionally independent for each segment. First the proportion of mutations per state for the three above mentioned states is calculated. For the subclonal state it is the same as the proportion of mutations for the subclone. For the clonal clone the proportion of mutations is split into an early and a late clonal proportion based on the molecular time. Second, the expected VAF for each of the possible states is calculated using the CCF and purity parameter. Third, calculate the likelihood for the observations given the expected VAF and proportion of mutations for each state. Accounting for the proportion of mutations per state makes it a mixture model.

Methods

Model implementation

The model is implemented in two different python libraries.

The first implementation is in the probabilistic programming library Pyro which is uses PyTorch as a backend. It provides powerful high-level abstractions that make it fast to define a statistical model while remaining flexible. It supports computation on CPU as well as GPU for scalability with large datasets.

The second implementation is with the Numpyro library which is a continuation of Pyro and contains

most of the same functionality. It supports the well-known NumPy API while using JAX to perform just in time compilation for the NumPy functions. This results in a massive performance increase overall and up to 100x for the Markov chain Monte Carlo algorithms (MCMC). To utilize just in time compilation care must be taken to ensure all functions in the model are compatible. This means array sizes must be known before runtime and only limited control flow is possible. Numpyro is also still in activate development.

For the Pyro as well as the Numpyro implementation we applied two different inference algorithms on the model. The first being the Hamiltonian Monte Carlo (HMC) algorithm. Due to technical limitations in Numpyro the implementation is limited to timing a maximum of 20 copy number gained segments for a single sample.

The second inference algorithm is stochastic variational inference (SVI) which approximates the posterior distribution and therefore scales better with large datasets that have many parameters. Additionally, SVI requires a guide function that is used to approximate the posterior distribution. The guide function can also be utilized to incorporate additional assumptions about the data. We set all latent variables as Beta distributions in the guide since this is a very flexible distribution. Due to technical issues with the enumeration of discrete latent variables in Numpyro there is only an SVI implementation for Pyro and not Numpyro.

Data simulation

In order to evaluate the performance of the different implementations of PyroTimer we simulated observations. By applying the functions used to calculate the molecular time in reverse with chosen values for the latent variables we can create realistic data where the true value of the parameters is known. Slight randomization was added by drawing the observed reads from a distribution using the simulated expected VAFs. This also creates the possibility to tweak individual parameters and observe the impact on the model's performance. To quantify the performance of PyroTimer we calculate the Root Mean Squared Error (RMSE) using the posterior distribution of the latent variable and the known true parameter value used for the simulation (Figure 5).

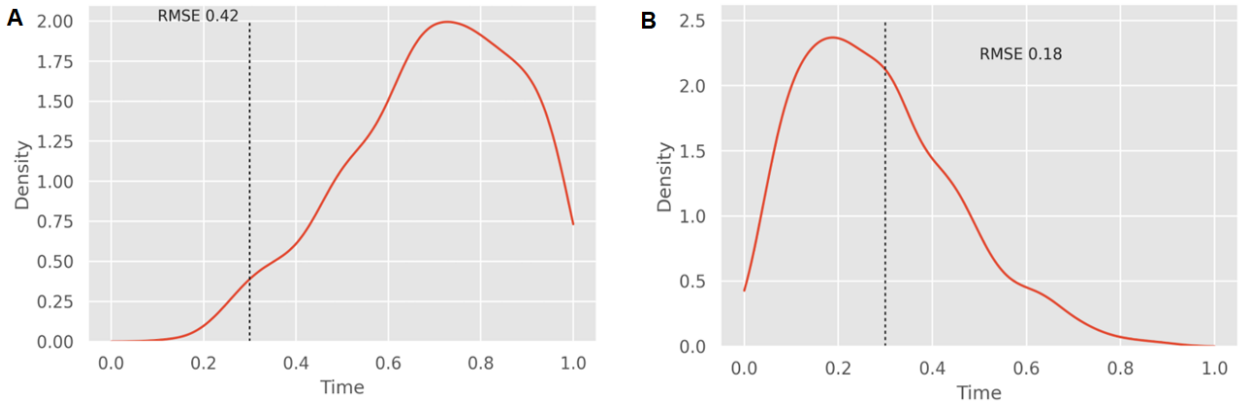

Figure 5: Example of Root Mean Squared Error (RMSE) to score a posterior distribution against a known true value. A. Example of a bad score, RMSE is higher. B. Example of a good score, RMSE is lower. Dotted line indicates true value of the parameter.

First, we optimized the hyper parameters for both inference algorithms by simulating a combination of different parameter values. For the HMC algorithm we can optimize the number of warmup rounds. For the SVI algorithm we can optimize the step size, number of iterations, samples taken from the guide

function. For each combination of parameters values we simulated 20 samples using the same set of 20 random seeds. The second step was to evaluate the performance of the different PyroTimer implementations when variables of the simulated data change. We evaluated the impact of coverage, purity, copy number gain configurations, time of duplication, number of mutations per gained segment, different subclone compositions. To simulate realistic values, we used the samples from PCAWG to establish a realistic range of values (Sup. Figure 12). For each variable value we simulated 50 samples.

PCAWG data

Even when trying to create realistic data a simulation will never be able to capture the variation present in real world data. We selected 193 samples from the PCAWG dataset to evaluate with the different implementations of PyroTimer. These samples were selected to conform with the 20-segment restriction from the Numpyro HMC implementation. We did not only perform comparisons for the different PyroTimer implementations but also compared against results from MutationTimeR.

Results

Hyper parameter optimalization

To find the optimal hyper parameters for each inference algorithm we simulated 20 samples for changing parameters values (Figure 6). The ideal hyper parameter provides accurate timing of duplications, indicated by a low RMSE score for the timing parameter and runs quickly.

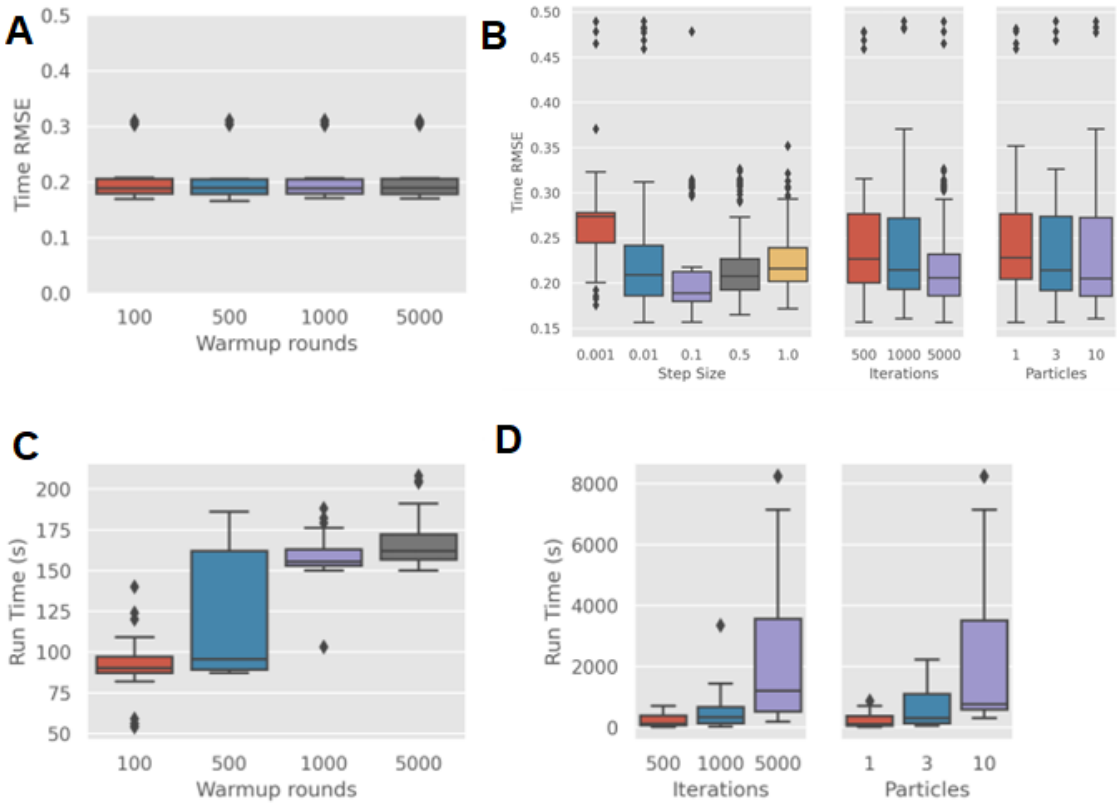

Figure 6: Optimizing hyper parameters for HMC and SVI algorithms. A. Timing RMSE score for Numpyro HMC warmup rounds. Increasing warmup rounds does not increase the score which indicates the model quickly becomes stable. B. Timing RMSE score

for SVI hyper parameters. A step size of 0.1 provides the best score for timing gains. Increasing the number of iterations and samples taken from the guide function slightly improves scoring. C. Runtime in seconds for different numbers of Numpyro HMC warmup rounds. Numpyro's just in time compilation keeps the runtime low and consistent. D. Runtime in seconds for different Pyro SVI hyper parameter values. Increasing iterations and samples taken from the guide function increases runtime rapidly.

Impact of data variables

To evaluate the impact of changing variables in the simulated data and how the different implementations of PyroTimer perform we simulated 50 samples for each data configuration (Figure 7). We compare the different implementations based on their timing RMSE score, lower is better. Overall, the performance of the methods for timing gains is very similar. Single gains are more difficult to time than CN-LOH or biallelic gains. The timing accuracy increases if there are more mutations per segment due to the increased number of datapoints that can be used to estimate the time of the gain.

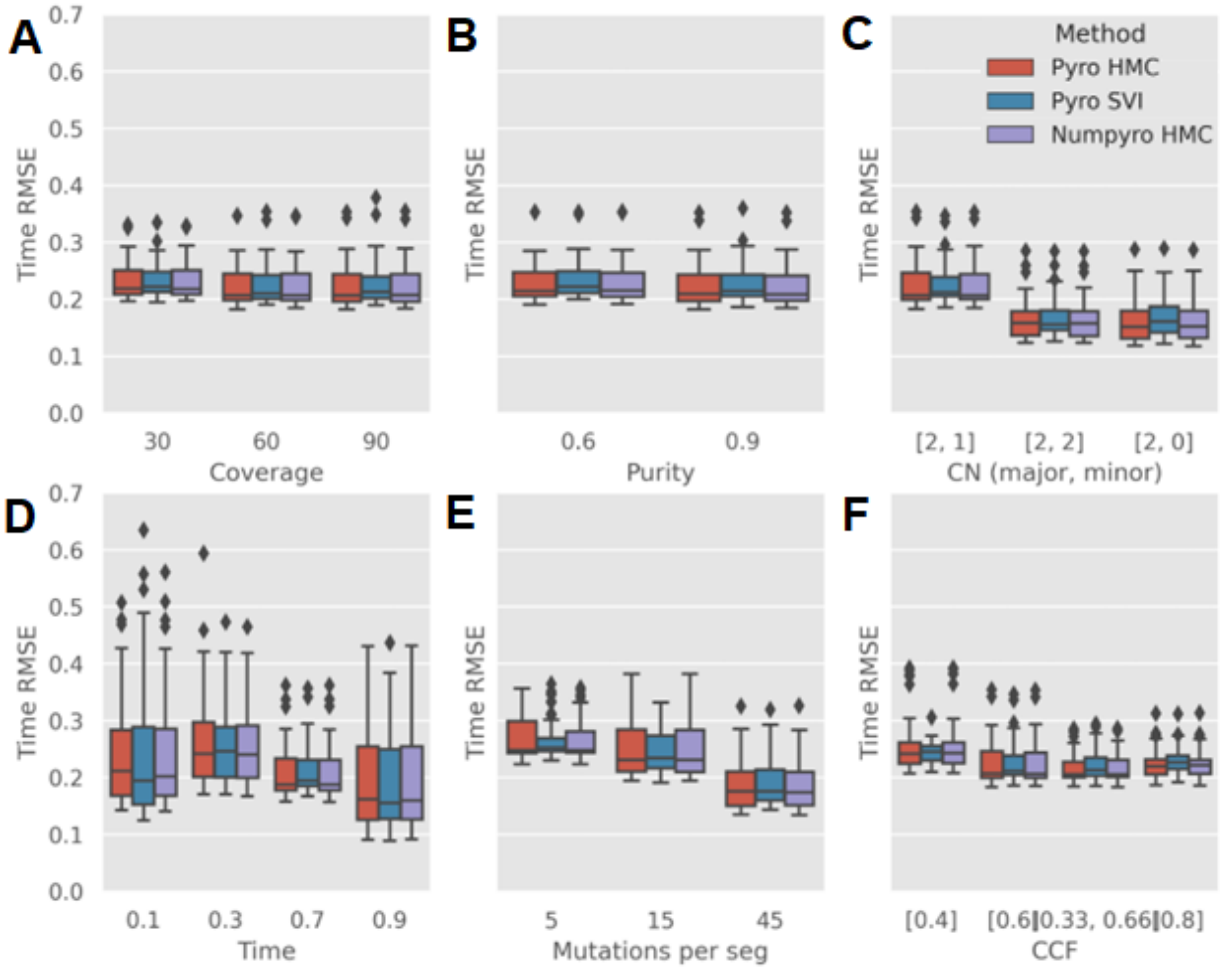

Figure 7: Comparing PyroTimer implementations and the impact of changing data variables by using simulated data. RMSE score for timing of the gained segment, lower is better. A. Coverage has little impact on the timing. B. Purity has little impact on the timing. C. Timing score for different copy number configurations. 2,1 is a single gain and is slightly harder to time than 2,2 which is a biallelic gain and 2,0 which is a CN-LOH. D. Timing score for different timings of the gain. Very early gains have larger spread in timing estimates. E. More mutations increase the timing score since there are more datapoints to use for the timing. F. Timing score for subclones with different CCF values.

However, when examining the CCF parameter RMSE the SVI algorithm appears to outperform the HMC implementations when there are two subclones present but not where there is only one subclone (Figure 8A). One explanation for this observation might be a label switching problem in Bayesian mixture models that use MCMC [12]. While the estimation of the CCF parameter appears to be impacted there is no difference in the timing accuracy between the SVI and HMC implementation in the case of two subclones (Figure 7F).

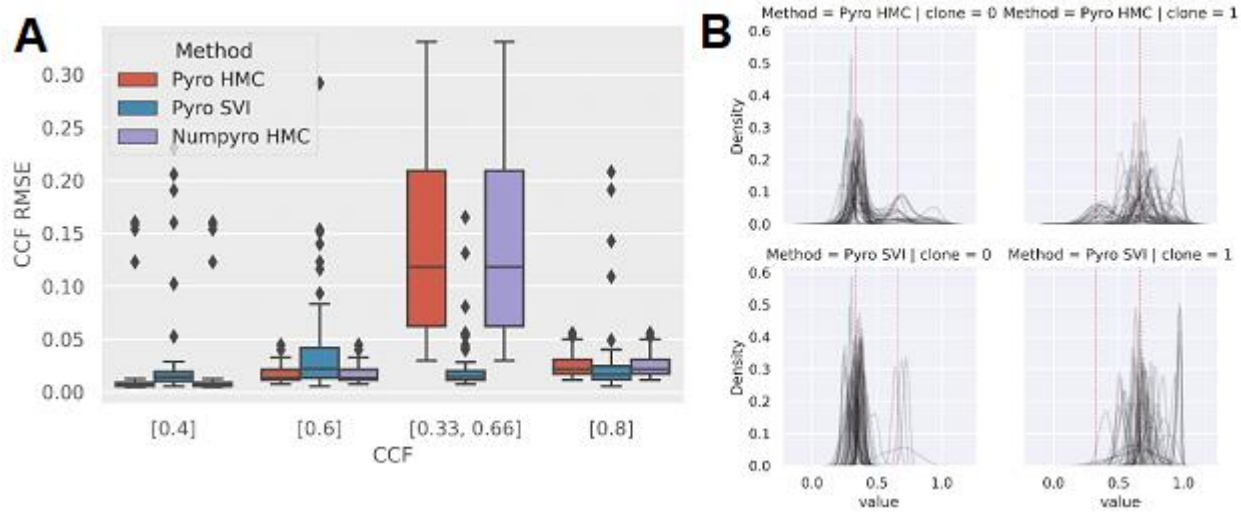

Figure 8: Estimation of the subclone CCF. A. CCF parameter score for different subclone configuration, lower is better. SVI outperforms both HMC methods when estimating the CCF of 2 subclones. B. Posterior distributions of the CCF parameter of the 50 samples. Redlines indicate the true CCF of the two clones. The Pyro HMC posterior distribution is divided between both correct values which might be the result of label switching that can occur for MCMC methods in Bayesian mixture models.

While there might be little difference in the accuracy of timing the gain between the implementations there is a significant difference between the runtime and how well they scale with larger datasets (Figure 9). The speed increase from the just in time compilation make Numpyro HMC the fastest implementation and it scales extremely well since the majority of the runtime is the initial compilation. The SVI algorithm does indeed scale better with larger datasets as expected.

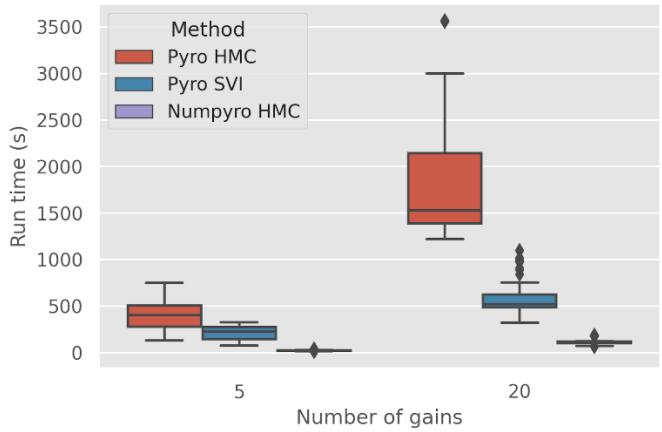

Figure 9: Runtime of Pyro implementations when increasing number of gains timed per sample. The Numpyro HMC implementation performs the fastest and scales best when the dataset grows. The Pyro HMC implementation is the slowest and scales poorly. The Pyro SVI implementation outperforms the Pyro HMC implementation.

PCAWG samples

We selected 193 samples from the PCAWG dataset to compare the different PyroTimer implementations when applied to real world data. Due to technical limitations of the Numpyro HMC implementation only samples with 20 or less gained segments were selected. PyroTimer is only able to time simple gains, CN-LOH and biallelic gains. Segments with multiple duplications could not be timed. This resulted in 1898 segments that were timed by PyroTimer. Since the accuracy of the Pyro HMC and the Numpyro HMC implementation were nearly identical, we continued with the Numpyro HMC because of the fast runtimes. The timed segments show a fairly strong positive correlation between the Pyro SVI and Numpyro HMC implementation (Figure 10). However, for the segments where there is major disagreement about the timing there is also a strong disagreement about the estimated purity of the samples. If both implementations assume a different purity this would affect the expected VAF frequencies and cause disagreement about timing of gained segments. Unfortunately, unlike the simulated data it is not possible to say which method has the correct timing.

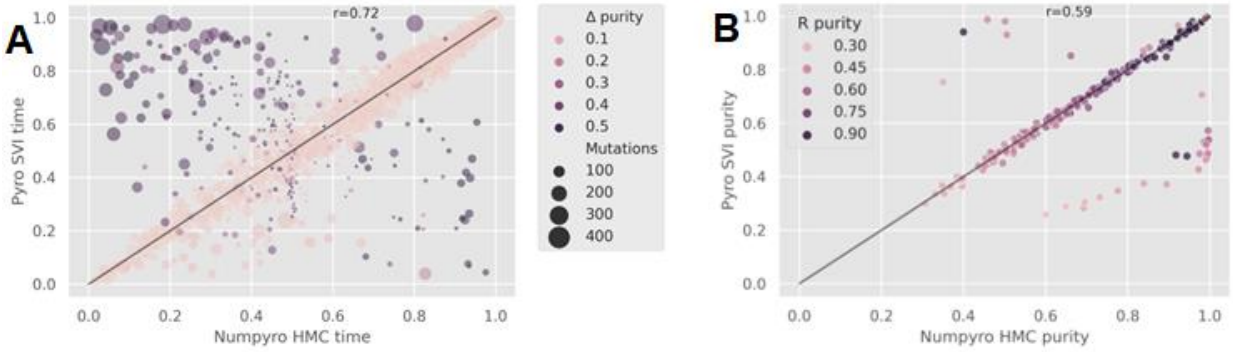

Figure 10: Comparison between Pyro SVI and Numpyro HMC on PCWAG samples. A. Pearson correlation coefficient is 0.72. Segments where the timing does not correspond is also where the purity estimate does not correspond. B. Correlation of the sample purity estimate for the Pyro SVI and Numpyro HMC. Colour indicates the consensus purity that the R implementation used. Both implementations are sometimes in disagreement with the consensus purity.

Besides comparing the different implementations of PyroTimer we also compared the estimated timings against the R implementation MutationTimeR. Again, there is a moderate positive correlation between the gained segments timing (Figure 11). The segments for which there is a major disagreement about the timing between MutationTimeR and PyroTimer are again from samples where there is also disagreement about sample purity. MutationTimeR does not estimate sample purity but uses a consensus purity generated by external tools. Other segments for which there is disagreement about the timing but not about the purity all have a low number of mutations, less than 5 mutations. The disagreement between the timing for MutationTimeR and PyroTimer seems to be caused by either the purity estimate or a too small amount of datapoints.

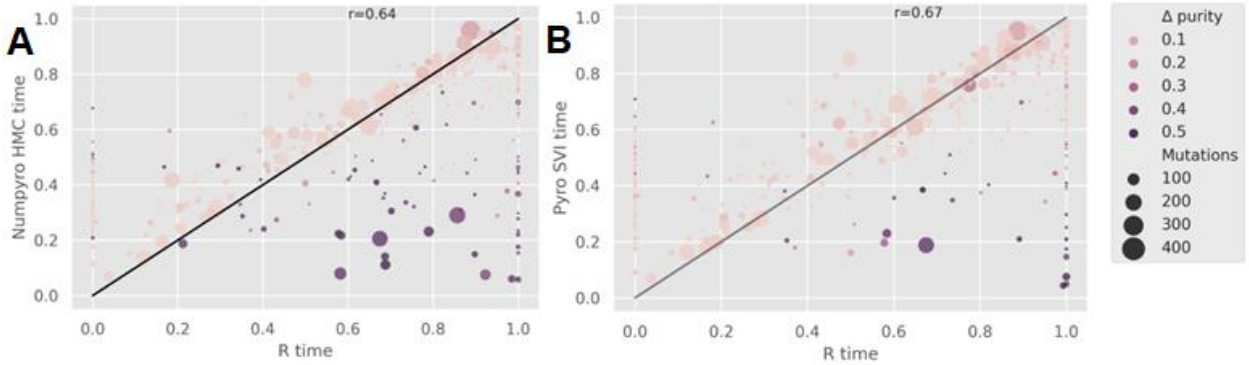

Figure 11: Comparison between PyroTimer and MutationTimeR. A. Pearson correlation coefficient is 0.64. Where the segments disagree about the timing there is either a disagreement about the purity or only a small number of mutations on the segment. B. Pearson correlation coefficient is 0.67. Where the segments disagree about the timing there is either a disagreement about the purity or only a small number of mutations on the segment.

Discussion

Our study presents PyroTimer, a Bayesian mixture model to time copy number gains implemented in a flexible probabilistic programming library that scales well with large datasets. We observed the performance gains which can be obtained by utilizing just in time compilation. The SVI algorithm performed as well as the HMC algorithm while also being quicker. Using simulated data based on real world data we validate the accuracy of PyroTimer. Lastly, we compared PyroTimer against the current implementation MutationTimeR.

While PyroTimer and MutationTimeR show a moderatly strong positive correlation between timing estimates, the purity estimate plays an important role in the disagreement between the methods. MutationTimeR uses a consensus purity that is estimated by external tools while PyroTimer estimates the purity along with the other latent variables. A possible compromise could be to use the consensus purity as a prior for the purity parameter. This would start PyroTimer of in the right direction but still maintain the benefit of not having a fixed value.

It is clear that the Numpyro library can give a major performance boost compared to Pyro by using the just in time compilation. It also contains several quality-of-life features that create a smoother experience compared to Pyro. However, the fact that it is still under active development resulted in significant amounts of time spend trying to resolve technical limitations that Pyro does not have.

The Numpyro HMC implementation is currently limited to a maximum of 20 copy number gains that can be timed per sample. PyroTimer is also still lacking some features compared to MutationTimeR like ordering SNVs in time and timing two single gains that happened at different timepoints. However, since

Pyro allows for much easier modelling than R it could replace the R version in the future and become the new standard for mutation timing as an easier to maintain and expand tool with great performance.

Acknowledgements

I would like to thank the German Cancer research centre for hosting me. I am grateful that Moritz Gerstung took me up in his group. I would like to thank Stefan Dentro for being my supervisor. I would also like to thank Artem for his technical assistance.

Supplementary figures

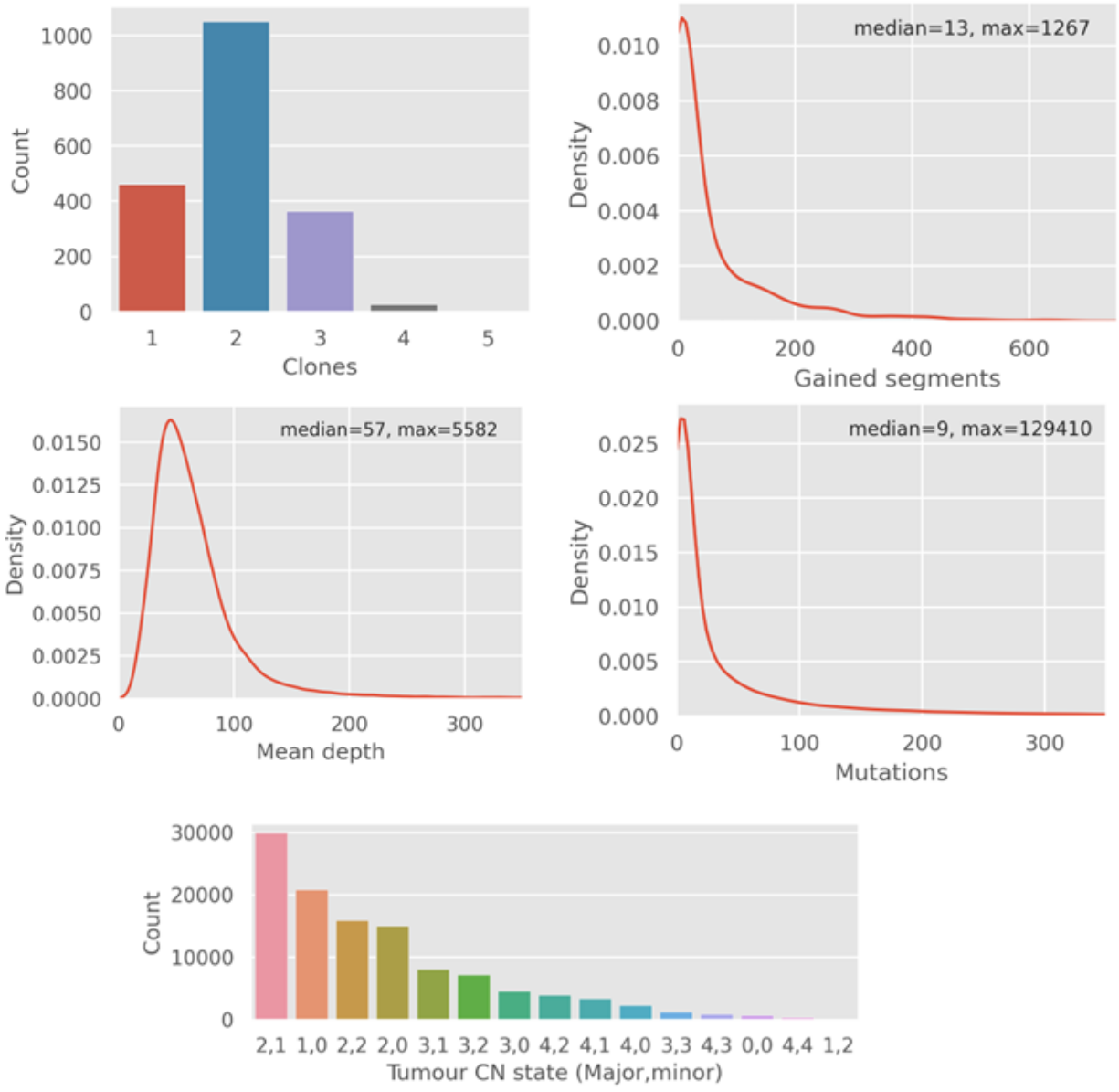

Figure 12: Distribution of data variables for PCAWG samples.

References

1. Dentro, S. C., Wedge, D. C. & van Loo, P. Principles of Reconstructing the Subclonal Architecture of Cancers. *Cold Spring Harb Perspect Med* **7**, (2017).

2. González, S., Volkova, N., Beer, P. & Gerstung, M. Immuno-oncology from the perspective of somatic evolution. *Seminars in Cancer Biology* **52**, 75–85 (2018).

3. Werner, B. *et al.* Measuring single cell divisions in human tissues from multi-region sequencing data. *Nature Communications* **11**, 1035 (2020).

4. Greenman, C. D. *et al.* Estimation of rearrangement phylogeny for cancer genomes. *Genome Research* **22**, 346–361 (2012).

5. Manders, F., van Boxtel, R. & Middelkamp, S. The Dynamics of Somatic Mutagenesis During Life in Humans. *Frontiers in Aging* **2**, (2021).

6. Purdom, E. *et al.* Methods and challenges in timing chromosomal abnormalities within cancer samples. *Bioinformatics* **29**, 3113–3120 (2013).

7. Jolly, C. & van Loo, P. Timing somatic events in the evolution of cancer. *Genome Biology* **19**, 1–9 (2018).

8. Gerstung, M. *et al.* The evolutionary history of 2,658 cancers. *Nature* **578**, 122–128 (2020).

9. Dentro, S. C. *et al.* Characterizing genetic intra-tumor heterogeneity across 2,658 human cancer genomes. *Cell* **184**, 2239-2254.e39 (2021).

10. Bingham, E. *et al.* Pyro: Deep Universal Probabilistic Programming. (2018).

11. Nik-Zainal, S. *et al.* DPClust. *Cell* **149**, 994–1007 (2012).

12. Grün, B. & Leisch, F. Dealing with label switching in mixture models under genuine multimodality. *Journal of Multivariate Analysis* **100**, 851–861 (2009).

13. McElreath, R. *Statistical Rethinking; A Bayesian Course with Examples in R and Stan; Second Edition*. https://www.crcpress.com/Chapman- doi:10.4324/9780429029608.