Dynamics of neuroblastoma initiation

Verena Körber

2022-09-03

Introduction

This script explains how mutation densities at ECA and MRCA together with the VAF distributions of somatic variants in individual tumors and the age at diagnosis were used to assess the dynamics of neuroblastoma initiation. This script is organized in three parts. First, we analyze the mutation densities at ECA and MRCA and introduce a model of neuroblastoma initiation that was fitted to this data. Second, we show how to analyze the SNV distributions of individual tumors using a population genetics model of tumor growth. Finally, we explain how the parameter estimates from both approaches together with the age at diagnosis yield insights into the actual time of tumor evolution.

To reproduce the analysis you should download the files on github and Mendeley and set the directories below accordingly. In addition, please download Supplementary Table 1 from the paper and store it on your local drive as well. We begin by loading the required libraries and defining directories necessary for the analysis.

```r
library("RColorBrewer")
library(ggplot2); theme_set(theme(panel.grid.major = element_blank(), panel.grid.minor =
↪  element_blank(), text=element_text(size=6, color="black"),
                                  panel.background = element_blank(), axis.line =
                                  ↪  element_line(colour = "black", size=0.75),
                                  axis.ticks = element_line(color = "black", size=0.75),
                                  axis.text = element_text(size=6, color="black")))
library(bedr)
library(openxlsx)
library(pammtools)
library(ComplexHeatmap)
library(ggsignif)
library(dplyr)
library(GenomicRanges)
library(ggbio)
library(deconstructSigs)
library(BSgenome.Hsapiens.UCSC.hg19)
library(bedr)
library(ggbeeswarm)
library(ggpubr)
library(survminer)
library(survival)
library(Hmisc)
library(scales)
library(mixtools)
library(reshape2)
library(circlize)
library(gridExtra)
library(MASS)
library(HDInterval)
library(cdata)
library(NBevolution)

## directory where RData files are stored
rdata.directory <- "./RData/"

## directory where the fitting results from the population genetics model of tumor
↪  initiation are stored as provided on Mendeley
fit.directory.initiation <- "../Model_fits_initiation/"

## directory where the fitting results from the population genetics model of tumor growth
↪  are stored as provided on Mendeley
fit.directory.growth <- "../Model_fits_tumor_growth/"

## directory, where the custom scripts are stored (corresponds to 'Analysis_and_plots' on
↪  github)
custom.script.directory <- "../Custom_scripts/"

## directory, where the data per tumor is stored
data.directory <- "../Example_data/"

## subdirectories storing SNVs
snv.directory <- "SNVs/"

## and copy number information
cnv.directory <- "ACEseq/"
```

Finally, we read in the supplementary table to get the sample information

```r
sample.information.discovery <- read.xlsx("../../Meta_data/Supplementary Table 1.xlsx",
↪  sheet="Discovery")
sample.information.validation <- read.xlsx("../../Meta_data/Supplementary Table 1.xlsx",
↪  sheet="Validation")
rownames(sample.information.discovery) <- sample.information.discovery$Tumor_ID
rownames(sample.information.validation) <- sample.information.validation$Tumor_ID
```

Prepare input data

We then select primary tumors/metastasis with an acquired telomere maintenance mechanism:

```r
tmm.tumors <-
↪  c(sample.information.discovery[sample.information.discovery$Telomere.maintenance.mechanism
↪  != "None" & sample.information.discovery$Sample.type %in% c("Primary",
↪  "Metastasis"),]$Tumor_ID,

↪  sample.information.validation[sample.information.validation$Telomere.maintenance.mechanism
↪  != "None" &  sample.information.validation$Sample.type %in% c("Primary",
↪  "Metastasis"),]$Tumor_ID)
```

Thereafter, we subset the mutation densities at ECA and MRCA in primary tumors / primary metastasis. Moreover, we don't use the ECA in tetraploid tumors since it is likely not the initiating event in these tumors. We then store the output in a list, which is subsequently used to fit the population genetics model to tumor initiation.

To do this, we here load an intermediate result, the mutation densities at ECA and MRCA with 95% CI, stored as dataframes in the object 'MRCA_timing.RData'. To learn how to generate these estimates, please refer to 'Example_code_mutation_density.pdf'.

```
load(paste0(rdata.directory, "/MRCA_timing.RData"))
print(head(mutation.time.mrca))
```

```
##          Mean      Min       Max Sample
## NBE1 841.4296 738.9165  932.3013   NBE1
## NBE2 527.8564 455.9823  586.4698   NBE2
## NBE3 993.6887 876.2861 1117.4714   NBE3
## NBE4 235.6108 207.2902  267.3796   NBE4
## NBE5 292.8934 235.2057  349.7939   NBE5
## NBE6 826.2565 735.5078  939.6769   NBE6
```

```
print(tmm.tumors)
```

```
##  [1] "NBE2"   "NBE3"   "NBE4"   "NBE7"   "NBE9"   "NBE10"  "NBE11"  "NBE12"
##  [9] "NBE13"  "NBE14"  "NBE15"  "NBE16"  "NBE17"  "NBE18"  "NBE19"  "NBE20"
## [17] "NBE21"  "NBE23"  "NBE27"  "NBE30"  "NBE31"  "NBE32"  "NBE33"  "NBE34"
## [25] "NBE36"  "NBE37"  "NBE41"  "NBE43"  "NBE44"  "NBE46"  "NBE47"  "NBE48"
## [33] "NBE49"  "NBE51"  "NBE52"  "NBE53"  "NBE54"  "NBE55"  "NBE57"  "NBE59"
## [41] "NBE60"  "NBE61"  "NBE62"  "NBE63"  "NBE64"  "NBE65"  "NBE101" "NBE102"
## [49] "NBE103" "NBE104" "NBE106" "NBE107" "NBE108" "NBE109" "NBE110" "NBE111"
## [57] "NBE112" "NBE113" "NBE115" "NBE117" "NBE118" "NBE119" "NBE120" "NBE121"
## [65] "NBE122" "NBE123" "NBE124" "NBE125" "NBE127" "NBE128" "NBE129" "NBE131"
## [73] "NBE132" "NBE133" "NBE134" "NBE136" "NBE137" "NBE138" "NBE140" "NBE145"
## [81] "NBE149" "NBE151" "NBE152" "NBE155" "NBE158" "NBE160" "NBE163" "NBE164"
## [89] "NBE165" "NBE167" "NBE179" "NBE180" "NBE181" "NBE183" "NBE186"
```

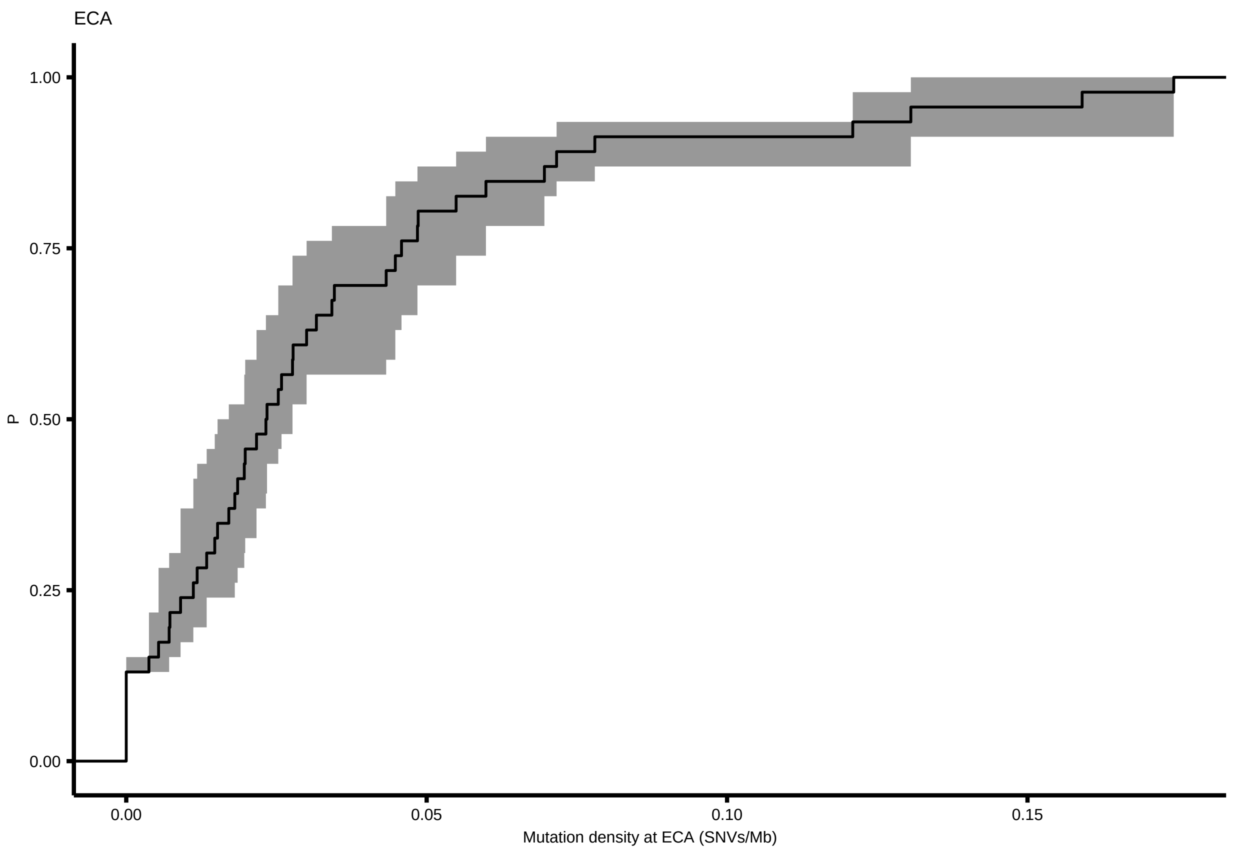
We now summarize the densities at ECA and MRCA to get a density distribution:

```
## collect the cumulative distribution of MRCA densities
P.MRCA = data.frame(Density=mutation.time.mrca[tmm.tumors,]$Mean)
rownames(P.MRCA) <- tmm.tumors
P.MRCA <- P.MRCA[order(P.MRCA$Density),,drop=F]
P.MRCA <- P.MRCA[!is.na(P.MRCA$Density),,drop=F]
P.MRCA$P <- seq(1, nrow(P.MRCA))/nrow(P.MRCA)
## lower and upper bounds
P.MRCA$P.upper = sapply(P.MRCA$Density, function(x){
  sum(mutation.time.mrca[tmm.tumors,]$Min <= x, na.rm = T)
})/nrow(P.MRCA)
P.MRCA$P.lower = sapply(P.MRCA$Density, function(x){
  sum(mutation.time.mrca[tmm.tumors,]$Max <= x, na.rm = T)
})/nrow(P.MRCA)

## collect the cumulative distribution of ECA densities
P.ECA = data.frame(Density=mutation.time.eca[tmm.tumors,]$Mean)
rownames(P.ECA) <- tmm.tumors
P.ECA <- P.ECA[order(P.ECA$Density),,drop=F]
P.ECA <- P.ECA[!is.na(P.ECA$Density),,drop=F]

## in tetraploid tumors, tetraploidy is likely not the initiating event; thus subset
tetraploid.tumors.discovery <- rownames(sample.information.discovery[
↪  sample.information.discovery$Rounded.ploidy == 4,])
tetraploid.tumors.validation <- rownames(sample.information.validation[
↪  sample.information.validation$Rounded.ploidy == 4,])
```

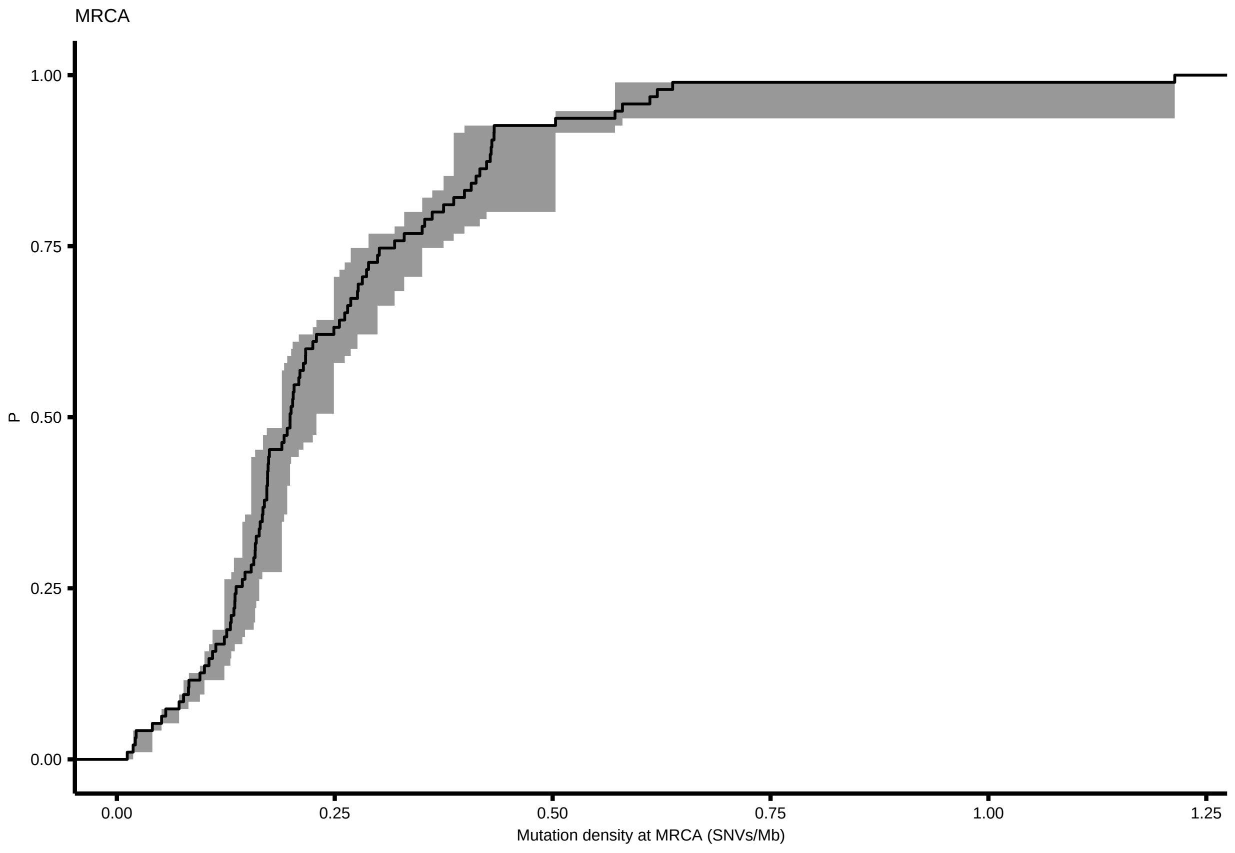
```
P.ECA <- P.ECA[setdiff(rownames(P.ECA), c(tetraploid.tumors.discovery,
↪  tetraploid.tumors.validation)),,drop=F]
P.ECA$P <- seq(1, nrow(P.ECA))/nrow(P.ECA)
## lower and upper bounds
P.ECA$P.upper = sapply(P.ECA$Density, function(x){
  sum(mutation.time.eca[rownames(P.ECA),]$Min <= x, na.rm = T)
})/nrow(P.ECA)
P.ECA$P.lower = sapply(P.ECA$Density, function(x){
  sum(mutation.time.eca[rownames(P.ECA),]$Max <= x, na.rm = T)
})/nrow(P.ECA)

save(P.MRCA, P.ECA,
     file=paste0(rdata.directory, "Input_data_NB_initiation.RData"))
```

Let's look into the distributions:

```
 ggplot(data = P.ECA, aes(x=Density/3.3/10^3, y=P, ymin = P.lower,ymax= P.upper)) +
          geom_stepribbon( alpha=0.5, col=NA)+
    stat_ecdf() + ggtitle("ECA") + scale_x_continuous("Mutation density at ECA
↪   (SNVs/Mb)")
```



```
 ggplot(data = P.MRCA, aes(x=Density/3.3/10^3, y=P, ymin = P.lower, ymax= P.upper)) +
        geom_stepribbon( alpha=0.5, col=NA)+
    stat_ecdf() + ggtitle("MRCA") + scale_x_continuous("Mutation density at MRCA
↪   (SNVs/Mb)")
```



Fit a population genetics model of tumor initiation to this data

We now used this data to learn dynamic parameters of neural crest development and tumor initiation in neuroblastoma. To this end, we fit a population genetics model to the mutation densities at ECA and MRCA. Briefly, the model assumes that there are two oncogenic events prior to tumor initiation, of which the first is associated with the ECA and the second with the MRCA. We considered two scenarios: in the first scenario, an initial phase of neuroblast expansion is followed by homeostatic turnover. In the second scenario, expansion is followed by contraction. The models have the following parameters:

- N: the maximal number of neuroblasts
- delta1: the loss rate during expansion (per division)
- mu: the neutral mutation rate (per division)
- muD1: the driver mutation rate associated with the first driver (per division)
- muD2: the driver mutation rate associated with the second driver (per division)
- r: the factor by which the first mutation reduces delta
- psurv: the survival probability of the second driver

Scenario 2 has the following additional parameter:

- delta2: the relative loss rate during contraction (=1 in scenario 1)

Using approximate Bayesian computation, we now simulated the onset of neuroblastoma initiation in both scenarios. To this end, we used the probabilities of acquiring 2 oncogenic drivers in either scenario to simulate the time point at which neuroblastoma growth commences. Please refer to the manuscript for a derivation and description of these probabilities.

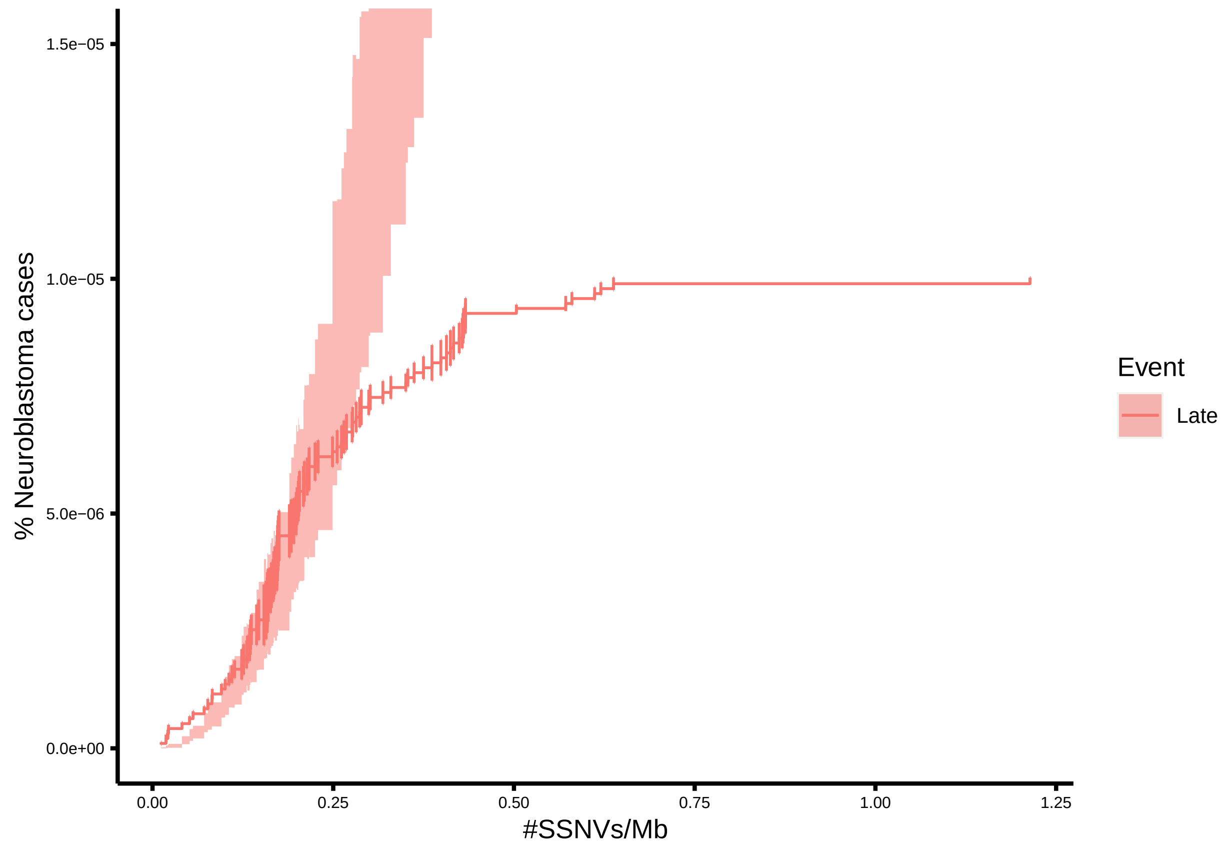
To reproduce the analysis you need to do the following steps: - for scenario 1: adjust the directories in the python script 'Expansion_homeostasis_continuous_evol.py' as well as in the associated R script 'Expansion_homeostasis_continuous_evol.R'. The script will source the function 'Model_NB_initiation.R', so make sure to have this script available as well and to set the path accordingly - for scenario 2: adjust the directories in the python script ‘Expansion_decay_continuous_evol.py’ as well as in the associated R script ‘Expansion_decay_continuous_evol.R’. The script will source the function ‘Model_NB_initiation.R’, so make sure to have this script available as well and to set the path accordingly - run both analyses ideally on a cluster

We will now proceed by analyzing the fits to both scenarios.

```
fits <- read.csv(paste0(fit.directory.initiation,
↪  "Expansion_homeostasis_continuous_evol.csv"))

parameter.samples <- data.frame(N=fits$par_N, muD1=fits$par_muD1, muD2=fits$par_muD2,
↪  mu=fits$par_mu
                                , delta1=fits$par_delta1, psurv=fits$par_psurv,
↪  r=fits$par_r)

## Simulate incidence curves. This custom function returns the simulated cumulative
↪  probabilities of ECA and MRCA at the measured densities
sim <- simulateCI(parameter.samples = parameter.samples, measured.mutation.times =
↪  P.MRCA, measured.mutation.times.eca=P.ECA, mode="homeostasis")

min.probabilities <- sim$min.mrca
max.probabilities <- sim$max.mrca
min.probabilities.eca <- sim$min.eca
max.probabilities.eca <- sim$max.eca
if(length(which(max.probabilities==1))>1){
  min.probabilities[which(max.probabilities==1)[-1]] <- NA
  max.probabilities[which(max.probabilities==1)[-1]] <- NA
}

to.plot <- data.frame(x = P.MRCA$Density/3.3/10^3,
                      data = P.MRCA$P*10^-5,
                      sd = (P.MRCA$P.upper - P.MRCA$P.lower)/2/1.95*10^-5,
                      lower = min.probabilities,
                      upper = max.probabilities,
                      Event = rep("Late", nrow(P.MRCA)))
```

When analyzing the fit, we notice that scenario 1 cannot explain the data well.

```
ggplot(to.plot, aes(x=x, y = data, ymin = data-sd, ymax = data +sd, col=Event,
↪  fill=Event)) + geom_step() + geom_errorbar()+
        geom_stepribbon(aes(x=x, ymin = lower, ymax = upper), alpha=0.5, col=NA)  +
        scale_x_continuous(name="#SSNVs/Mb") + scale_y_continuous(name = "% Neuroblastoma
↪  cases") +
        theme(panel.grid.major = element_blank(), panel.grid.minor = element_blank(),
↪  text=element_text(size=10),
              panel.background = element_blank(), axis.line = element_line(colour =
              ↪  "black"))+
        coord_cartesian(ylim = c(0, 1.5*10^-5))
```

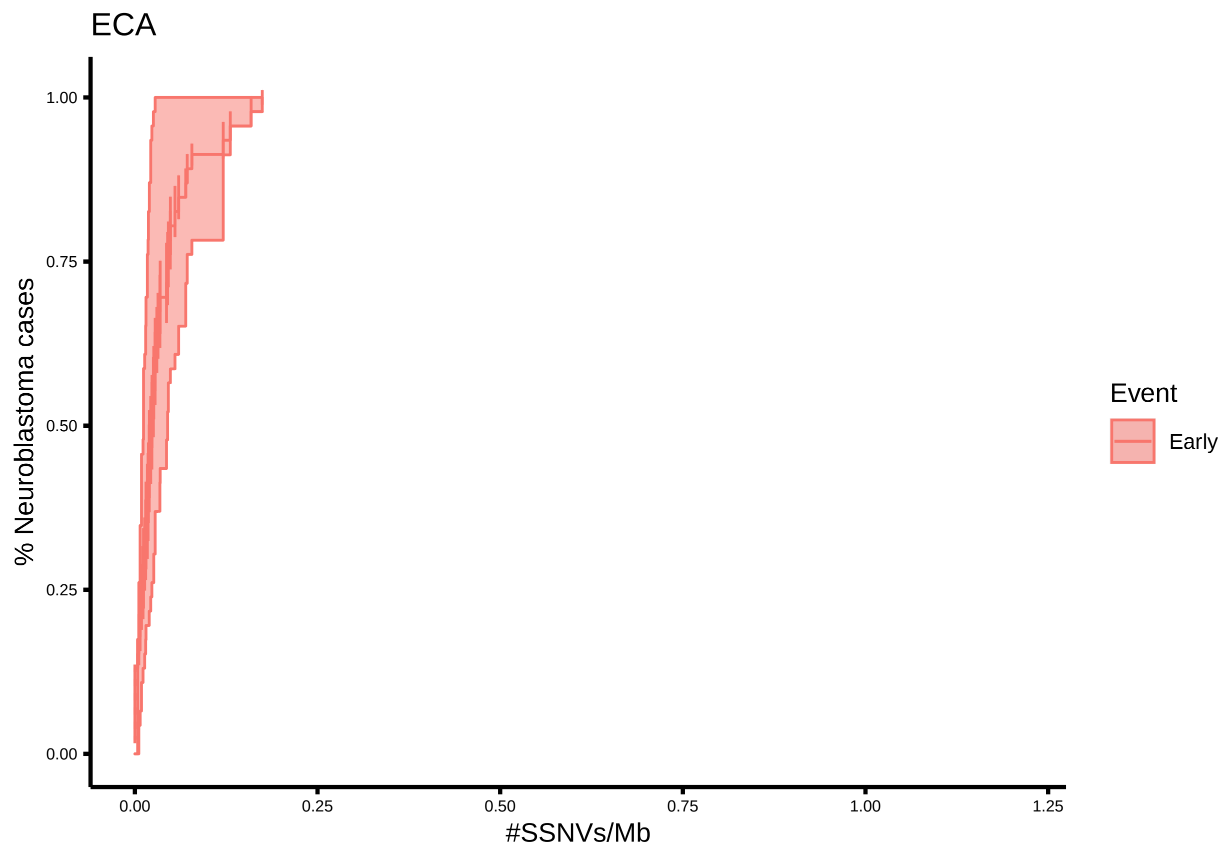


Next, we analyze scenario 2:

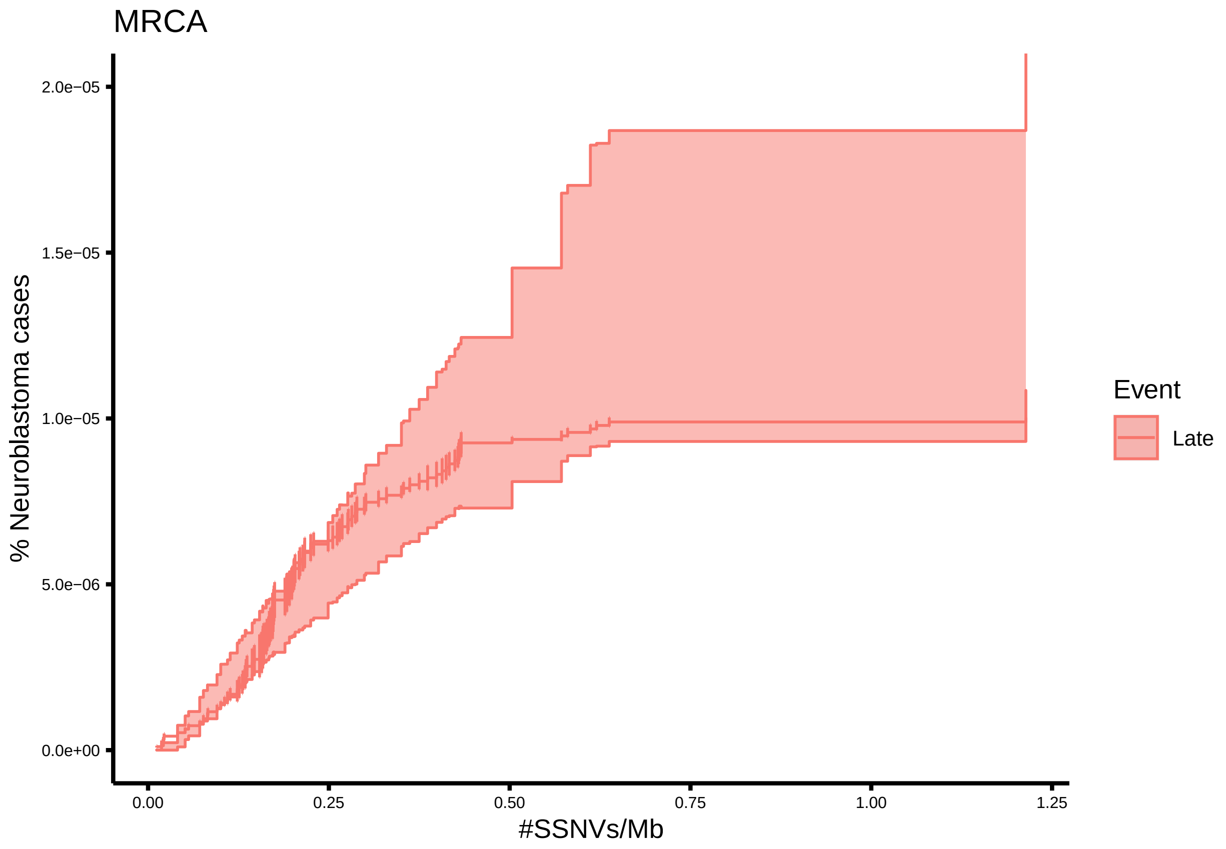
```
fits <- read.csv(paste0(fit.directory.initiation, "Expansion_decay_continuous_evol.csv"))

parameter.samples <- data.frame(N=fits$par_N, delta1=fits$par_delta1, muD1=fits$par_muD1,
↪
                                muD2=fits$par_muD2, mu=fits$par_mu,
                                delta2=fits$par_delta2, psurv=fits$par_psurv,
                                ↪  r=fits$par_r)


## Simulate incidence curves. This custom function returns the simulated cumulative
↪  probabilities of ECA and MRCA at the measured densities
sim <- simulateCI(parameter.samples = parameter.samples, measured.mutation.times =
↪  P.MRCA, measured.mutation.times.eca=P.ECA, mode="decay")
min.probabilities <- sim$min.mrca
max.probabilities <- sim$max.mrca
min.probabilities.eca <- sim$min.eca
max.probabilities.eca <- sim$max.eca
if(length(which(max.probabilities==1))>1){
  min.probabilities[which(max.probabilities==1)[-1]] <- NA
  max.probabilities[which(max.probabilities==1)[-1]] <- NA
}

to.plot <- data.frame(x = P.MRCA$Density/3.3/10^3, ## conver to SNVs/Mb
                      data = P.MRCA$P*10^-5,
                      sd = (P.MRCA$P.upper - P.MRCA$P.lower)/2/1.95*10^-5,
                      lower = min.probabilities,
                      upper = max.probabilities,
                      Event = rep("Late", nrow(P.MRCA)))

to.plot.eca <- data.frame(x = P.ECA$Density/3.3/10^3,
                          data = P.ECA$P,
                          sd = (P.ECA$P.upper - P.ECA$P.lower)/2/1.95,
                          lower = min.probabilities.eca,
                          upper = max.probabilities.eca,
                          Event = rep("Early", nrow(P.ECA)))
```

With scenario 2 we obtain good fits to the data:

```
## ECA
ggplot(to.plot.eca, aes(x=x, y = data, ymin = data-sd, ymax = data +sd, col=Event,
↪   fill=Event)) + geom_step() + geom_errorbar()+
        geom_stepribbon(aes(x=x, ymin = lower, ymax = upper), alpha=0.5)  +
        scale_x_continuous(name="#SSNVs/Mb", limits=c(0, max(to.plot$x))) +
        scale_y_continuous(name = "% Neuroblastoma cases") +
        theme(panel.grid.major = element_blank(), panel.grid.minor = element_blank(),
↪   text=element_text(size=10),
              panel.background = element_blank(), axis.line = element_line(colour =
              ↪   "black")) + ggtitle("ECA")
```



```
## MRCA

ggplot(to.plot, aes(x=x, y = data, ymin = data-sd, ymax = data +sd, col=Event,
↪   fill=Event)) + geom_step() + geom_errorbar()+
      geom_stepribbon(aes(x=x, ymin = lower, ymax = upper), alpha=0.5)  +
      coord_cartesian(ylim=c(0, 2*10^-5)) +
      scale_x_continuous(name="#SSNVs/Mb") +
      scale_y_continuous(name = "% Neuroblastoma cases") +
      theme(panel.grid.major = element_blank(), panel.grid.minor = element_blank(),
↪  text=element_text(size=10),
            panel.background = element_blank(), axis.line = element_line(colour =
            ↪  "black")) + ggtitle("MRCA")
```

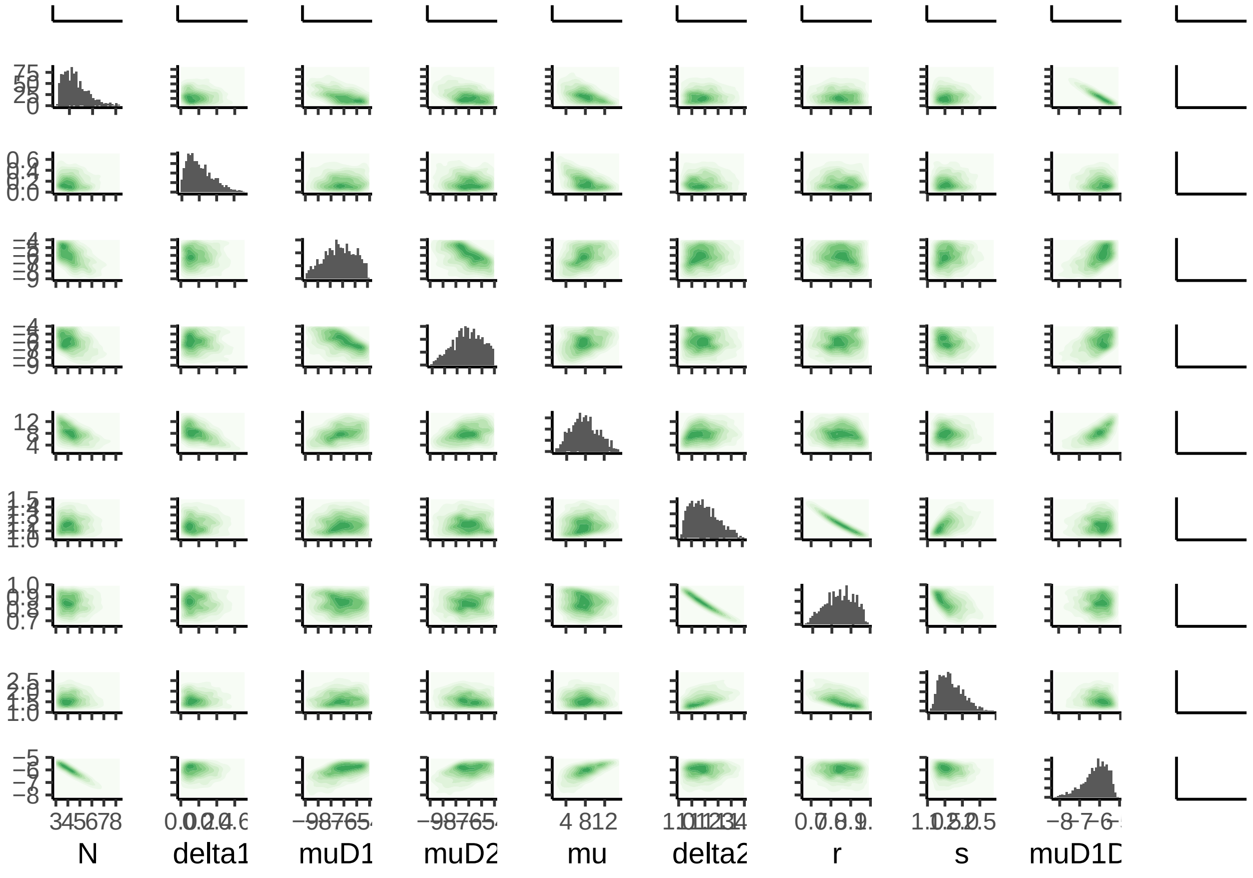


When looking into the parameters at the best fit, we notice that several of them, like the maximal number of neuroblasts, are well identifiable, whereas the selective advantage associated with the second driver is not:

```
## do all correlations
## compute selective advantage from survival probability
fits$par_s <- fits$par_delta2/(1-fits$par_psurv)
parameter.samples$s <- parameter.samples$delta2/(1-parameter.samples$psurv)
## compute geometric mean of mu1 and mu2
parameter.samples$muD1D2 <-
↪  log10(sqrt(10^parameter.samples$muD2*10^parameter.samples$muD1))

## parameters to plot
parameters <- c("par_N", "par_delta1", "par_delta2", "par_mu", "par_muD1", "par_muD2",
↪  "par_s", "par_r", "muD1D2")

## specify the variables I want to plot
meas_vars <- colnames(parameter.samples)
```

```r
## a data frame of all combinations of its arguments
controlTable <- data.frame(expand.grid(meas_vars, meas_vars, stringsAsFactors = F))

## rename the columns
colnames(controlTable) <- c("x", "y")

## add the key column
controlTable <- cbind(data.frame(par_key = paste(controlTable[[1]], controlTable[[2]]),
↪  stringsAsFactors = F), controlTable)

## create the new data frame
to.plot <- rowrecs_to_blocks(parameter.samples, controlTable)

## re-arrange with facet_grid
splt <- strsplit(to.plot$par_key, split=" ", fixed=TRUE)
to.plot$xv <- vapply(splt, function(si) si[[1]], character(1))
to.plot$yv <- vapply(splt, function(si) si[[2]], character(1))

to.plot$xv <- factor(as.character(to.plot$xv), meas_vars)
to.plot$yv <- factor(as.character(to.plot$yv), meas_vars)


## arrange manually

to.plot$xaxis <- F
to.plot$yaxis <- F
to.plot$xaxis[to.plot$yv == to.plot$xv[sqrt(length(unique(to.plot$par_key)))]] <- T
to.plot$yaxis[to.plot$xv==to.plot$xv[1]] <- T
to.plot$topm <- F
to.plot$rightm <- F
to.plot$topm[to.plot$yv == to.plot$xv[1]] <- T
to.plot$rightm[to.plot$xv==to.plot$xv[sqrt(length(unique(to.plot$par_key)))]] <- T

p <- list()

## introduce an artificial top row and right column

for(i in 1:(sqrt(length(unique(to.plot$par_key))))){
  p[[length(p)+1]] <- ggplot(data.frame()) + geom_point()+
    theme_bw() + theme(plot.margin = unit(c(-10, -10, -10, -10), "pt"),
                       panel.border = element_blank(), panel.grid.major =
                       ↪  element_blank(),
                       panel.grid.minor = element_blank(), axis.line =
                       ↪  element_line(colour = "black")) + theme(legend.position =
                       ↪  "none")

}

for(i in unique(to.plot$par_key)){


  tmp <- to.plot[to.plot$par_key==i,]
  if(tmp$xv=="psurv" | tmp$yv=="psurv"){next}

  if(tmp$xv[1]==tmp$yv[1]){
    p[[length(p)+1]] <- ggplot(tmp, aes(x=x)) +
      geom_histogram() + scale_x_continuous(name=tmp$xv[1]) +
↪   scale_y_continuous(name=tmp$yv[1])+
      theme_bw() + theme(panel.border = element_blank(), panel.grid.major =
↪   element_blank(),
                         panel.grid.minor = element_blank(), axis.line =
                         ↪   element_line(colour = "black")) +
      theme(legend.position = "none")

  }else{
    p[[length(p)+1]] <- ggplot(tmp, aes(x=x, y=y)) +
      geom_density_2d_filled(col=NA, contour_var = "ndensity", aes( fill = ..level..)) +
      scale_fill_manual(values=colorRampPalette(brewer.pal(9, "Greens"))(15)) +
      scale_x_continuous(name=tmp$xv[1]) + scale_y_continuous(name=tmp$yv[1])+
      theme_bw() + theme(panel.border = element_blank(), panel.grid.major =
↪   element_blank(),
                         panel.grid.minor = element_blank(), axis.line =
                         ↪   element_line(colour = "black")) + theme(legend.position =
                         ↪   "none")

  }

  ## top-row and right column: adjust margins differently
  if(tmp$rightm[1] & tmp$topm[1]){
    p[[length(p)]] <-  p[[length(p)]] +  theme(plot.margin = unit(c(-10, -10, -10, -10),
↪   "pt"))
  }else if(tmp$rightm[1]){
    p[[length(p)]] <-  p[[length(p)]] +  theme(plot.margin = unit(c(-10, -10, -10, -10),
↪   "pt"))
  }else if(tmp$topm[1]){
    p[[length(p)]] <-  p[[length(p)]] +  theme(plot.margin = unit(c(-10, -10, -10, -10),
↪   "pt"))
  }else{
    p[[length(p)]] <-  p[[length(p)]] +  theme(plot.margin = unit(c(-10, -10, -10, -10),
↪   "pt"))
  }

  if(tmp$xaxis[1]==F){
    p[[length(p)]] <-  p[[length(p)]] +  theme(axis.title.x = element_blank(),
                                               axis.text.x = element_blank())
  }

  if(tmp$yaxis[1]==F){
    p[[length(p)]] <-  p[[length(p)]] +  theme(axis.title.y = element_blank(),
                                               axis.text.y = element_blank())
  }

  if(tmp$rightm[1]){
    p[[length(p)+1]] <- ggplot(data.frame()) + geom_point()+
      theme_bw() + theme(panel.border = element_blank(), panel.grid.major =
↪   element_blank(),
                              panel.grid.minor = element_blank(), axis.line =
                               ↪  element_line(colour = "black")) +
        theme(plot.margin = unit(c(-10, -10, -10, -10), "pt"),
              legend.position = "none")

  }

}

ggarrange(plotlist=p, nrow=10, ncol=10, align="hv")
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



Fit a population genetics model of tumor growth to the VAF distribution of an individual tumor

To gain additional insight into tumor evolution, we moreover, analyzed the variant allele frequency distribution of individual tumors with a population genetics model of tumor growth. The model is inspired by Williams et al., Nature Genetics, 2016 and Ohtsuki and Innan, Theoretical Population Biology, 2017 and is described in detail in the methods section of the paper. In brief, we stratify mutations by copy number state and then consider the VAF distribution to consist of a clonal peak, binomially distributed around the clonal VAF and a subclonal tail, arising from exponential expansion of the tumor. For each tumor, we fit the model to the variants on 1 to 4 copies and weight each fit by the fraction of the genome at the respective copy number. We will exemplarily look into tumor NBE11, provided as example data set on github.

We being by stratifying the mutation counts:

```r
tumor <- "NBE11"

## read in the mutation file
mutations <- list.files(paste0(data.directory, "/", tumor, "/", snv.directory, "/"),
↪  pattern="somatic_snvs_conf_8_to_10", full.names = T)[1]
mutations <- read.vcf(mutations)
```

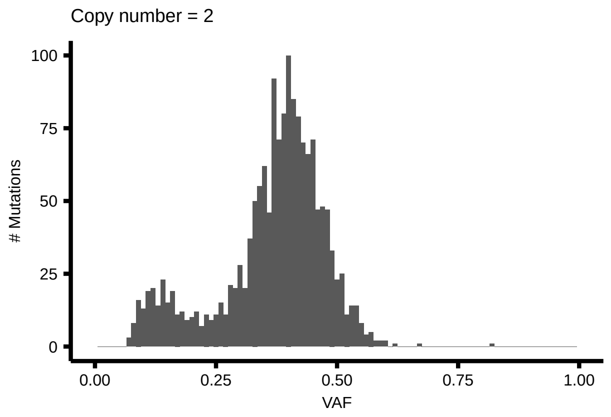
```
## READING VCF
##  * checking if file exists... PASS
##  * Reading vcf header...
##    Done
##  * Reading vcf body...
##    Done
##  * Parse vcf header...
##    Done
```

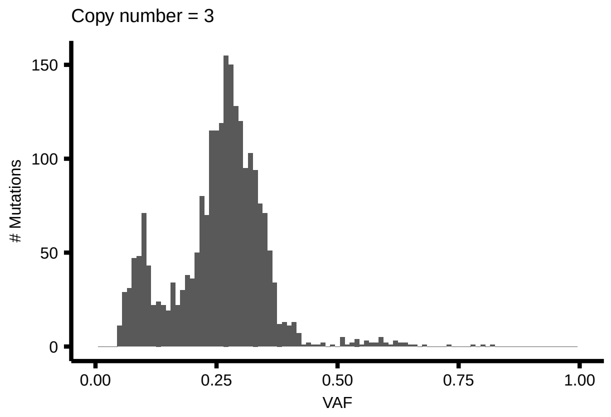
```r
## subset on chromosomes 1-22, X, Y
mutations$vcf <- mutations$vcf[!mutations$vcf$CHROM %in% c("X", "Y"), ]

## Read in copy number information and extract ploidy/purity, as before
aceseq <- list.files(paste0(data.directory, "/", tumor, "/", cnv.directory),
↪  pattern="comb_pro_extra", full.names = T)[1]

purity.ploidy <- Extract.purity.ploidy.from.ACEseq(aceseq)

purity <- purity.ploidy$purity
ploidy <- purity.ploidy$ploidy

copy.number.info <- read.delim(aceseq, sep="\t", stringsAsFactors = F)
## subset on chromosomes 1-22, X, Y
copy.number.info <- copy.number.info[!copy.number.info$X.chromosome %in% c("X", "Y"),]
## obtain the coverage ratios for the mutations of interest

## Extract copy number info for each mutation
cnv.info.per.mutation <- Extract.copy.number.info.per.SSNV(mutations, copy.number.info)
## obtain the coverage ratios at mutated loci
coverage.ratios <- cnv.info.per.mutation$coverage.ratio
bafs <- cnv.info.per.mutation$baf
genotype <- cnv.info.per.mutation$genotype
tcn <- cnv.info.per.mutation$tcn

## Extract readcounts of reference and alternative bases
readcounts <- Extract.info.from.vcf(mutations, info="readcounts")


 ########################################################################
```

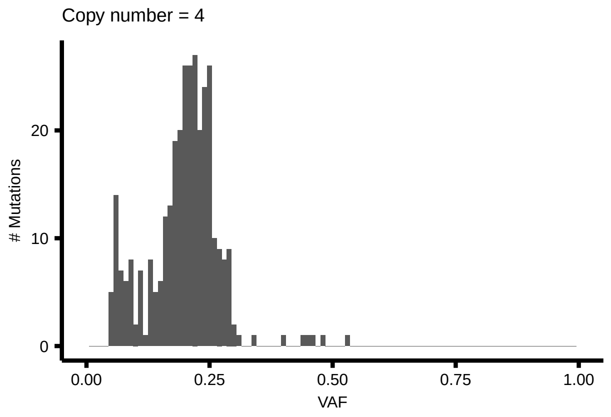
```
## Iterate through all copy number states and plot the VAF distribution separately for
↪  each copy number

vafs.this.tumor <- list()
genome.size.this.tumor <- list()
p <- list()

## Plot separately for each copy number

for(k in 1:4){

  expected.coverage.ratio <- (k*purity + (1-purity)*2)/(ploidy*purity+(1-purity)*2)

  readcounts. <- readcounts[((coverage.ratios > (expected.coverage.ratio - 0.1) &
↪  coverage.ratios < (expected.coverage.ratio + 0.1) & !is.na(coverage.ratios)) |
                                (tcn ==k & !is.na(tcn))) ,,drop=F]


  vafs.this.tumor[[k]] <- readcounts.

  genome.size <-
↪  sum(as.numeric(copy.number.info[(copy.number.info$tcnMeanRaw>(expected.coverage.ratio-0.1)
↪  & copy.number.info$tcnMeanRaw<(expected.coverage.ratio+0.1)) |
↪  (as.numeric(copy.number.info$TCN)==k & !is.na(as.numeric(copy.number.info$TCN)))
↪  ,]$end)-

↪  as.numeric(copy.number.info[(copy.number.info$tcnMeanRaw>(expected.coverage.ratio-0.1)
↪  & copy.number.info$tcnMeanRaw<(expected.coverage.ratio+0.1)) |

↪  (as.numeric(copy.number.info$TCN)==k &
↪  !is.na(as.numeric(copy.number.info$TCN))),]$start))

   genome.size.this.tumor[[k]] <- genome.size

   if(nrow(readcounts.)==0){next}

    p[[length(p)+1]] <-  ggplot(data.frame(VAF=readcounts.[,2]/rowSums(readcounts.)),
↪  aes(x=VAF)) + geom_histogram(binwidth = 0.01) +
         theme(panel.grid.major = element_blank(), panel.grid.minor = element_blank(),
         panel.background = element_blank(), axis.line = element_line(colour = "black"))
          ↪  + scale_y_continuous(name="# Mutations") +
         scale_x_continuous(limits=c(0,1)) + ggtitle(paste0("Copy number = ", k))

   }
```

Look at the VAF distribution stratified by copy number

```
ggarrange(plotlist = p, nrow=2, ncol=2)
```







We did this for all tumors and stored the VAFs in the common list object ‘VAFs_all_tumors.RData’. We now use this data to learn the parameters of tumor expansion. The model has the following parameters

- n_clonal: number of clonal SNVs
- mu: neutral mutation rate (per division)
- delta: the loss rate during expansion (per division)

We fit the model to the data using approximate Bayesian computation. To this end, we determine the expected number of variants per VAF at a given parameter set. We then simulate sequencing by drawing from a binomial distribution with success probability according to the respective VAF and compare the simulated data to its measured correlate. To reproduce the analysis, perform the following steps:

- Adjust the directories in the script ‘Neutral_fit_pre_clonal_and_clonal.R’; the model will be run on copy number states with a total length of >10^8 bp only.
- Adjust the directories in the script ‘Neutral_fit.py’
- Ideally run the analysis on a cluster

We now look into the fit for tumor NBE11 and simulate the model with every parameter set sampled from the posterior distribution.

```r
fits <- list.files(paste0(fit.directory.growth), pattern=paste0(tumor, ".csv"),
↪  full.names = T)
fits <- read.csv(fits)

## Source the model file and input the data for tumor NBE11; the model file will also
↪  determine which copy number state was used for fitting.

i <- tumor
source(paste0(custom.script.directory, "Neutral_fit_pre_clonal_and_clonal.R"))
```

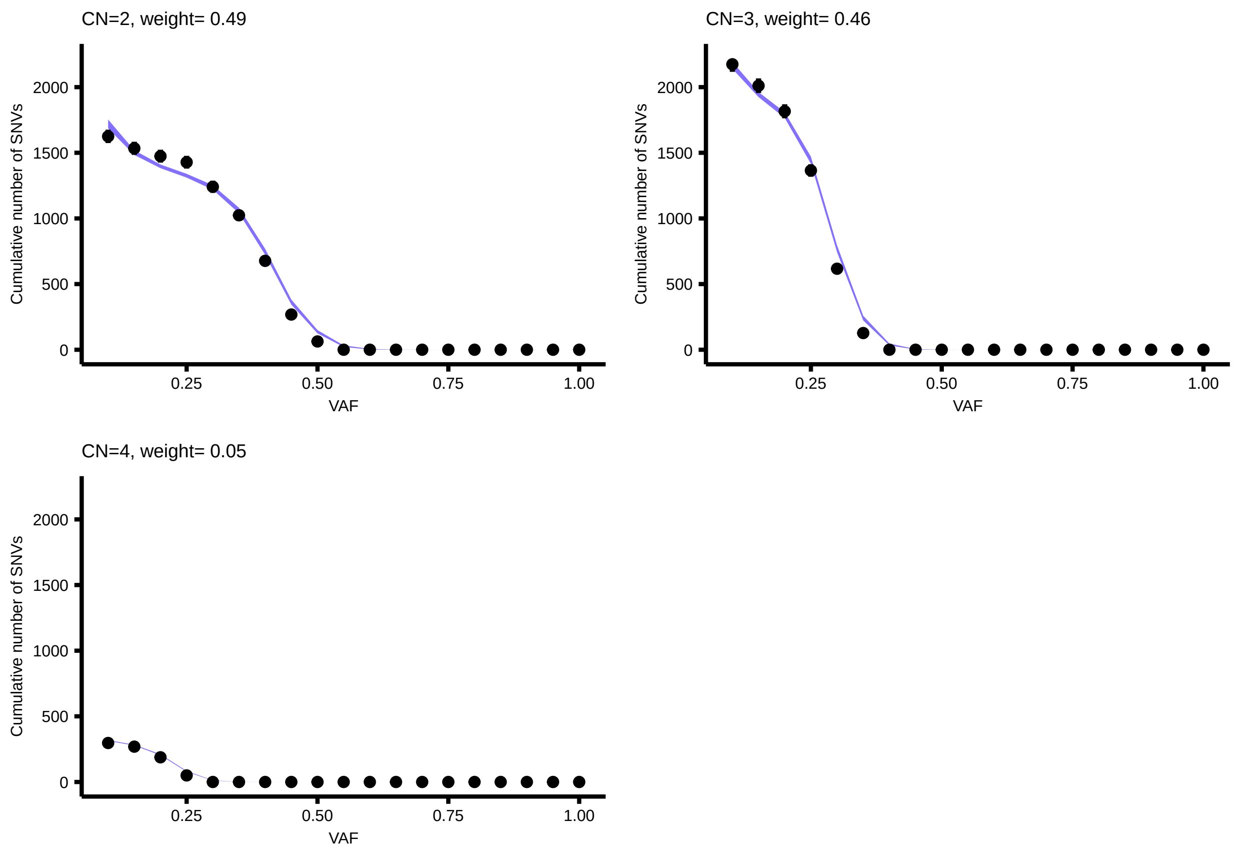
```r
## Simulate the model for the copy number states that went into the fit
if(fit.haploid){
  sim.haploid <- matrix(0, nrow=1000, ncol=length(mySumStatData$haploid))
}
if(fit.diploid){
  sim.diploid <- matrix(0, nrow=1000, ncol=length(mySumStatData$diploid))
}
if(fit.triploid){
  sim.triploid <- matrix(0, nrow=1000, ncol=length(mySumStatData$triploid))
}
if(fit.tetraploid){
  sim.tetraploid <- matrix(0, nrow=1000, ncol=length(mySumStatData$tetraploid))
}

## Run a model simulation for each parameter set obtained from the posterior sample. In
↪  the model, the mutation counts were extrapolated to the haploid genome
for(j in 1:nrow(fits)){
  parms <- list(delta=fits$par_delta[j], n_clonal=fits$par_n_clonal[j], mu =
↪  fits$par_mu[j])
  output <- myModel(parms)

  if(fit.haploid){
    sim.haploid[j,] <- output$haploid
    max.haploid <- max(apply(sim.haploid, 2, max)*haploid.genome.fraction/(3.3*10^9))
  }else{
    max.haploid <- 0
  }
  if(fit.diploid){
    sim.diploid[j,] <- output$diploid
    max.diploid <- max(apply(sim.diploid, 2, max)*diploid.genome.fraction/(3.3*10^9))
  }else{
    max.diploid <- 0
  }
  if(fit.triploid){
    sim.triploid[j,] <- output$triploid
    max.triploid <- max(apply(sim.triploid, 2,
↪  max)*triploid.genome.fraction/(3.3*10^9))
  }else{
    max.triploid <- 0
  }
  if(fit.tetraploid){
    sim.tetraploid[j,] <- output$tetraploid
    max.tetraploid <- max(apply(sim.tetraploid, 2,
↪  max)*tetraploid.genome.fraction/(3.3*10^9))
  }else{
    max.tetarploid <- 0
  }

}
```

Finally, we plot the measured and simulated cumulative VAF distribution for each copy number state. In the model fit, the data were extrapolated to the haploid genome, but we plot as was measured.

```
 y.max <- max(max.haploid, max.diploid, max.triploid, max.tetraploid)

 p <- list()

 if(fit.haploid){
  to.plot <- data.frame(VAF=seq(0.1, 1, 0.05), Data =
↪  mySumStatData$haploid*haploid.genome.fraction/(3.3*10^9),
             Mmin=apply(sim.haploid, 2, quantile,
             ↪  p=0.025)*haploid.genome.fraction/(3.3*10^9),
             Mmax=apply(sim.haploid, 2, quantile,
             ↪  p=0.975)*haploid.genome.fraction/(3.3*10^9))
  p[[length(p)+1]] <- eval(substitute(ggplot(to.plot, aes(x=VAF, y=Data,
                      ymin = Data - sqrt(Data),
                      ymax = Data + sqrt(Data)
                      )) +
   geom_ribbon(aes(ymin=Mmin,
           ymax=Mmax),
         fill="lightslateblue")+ geom_point() + geom_errorbar(width=0.01) +
         ↪ ggtitle(paste("CN=1, weight=",
         ↪ round(haploid.genome.fraction/(haploid.genome.fraction+diploid.genome.fraction+tri
         ↪ digits=2))) +  scale_y_continuous(limits=c(0, y.max),
         ↪ name="Cumulative number of SNVs"),
         ↪ list(haploid.genome.fraction=haploid.genome.fraction,
         ↪ diploid.genome.fraction=diploid.genome.fraction,
         ↪ triploid.genome.fraction=triploid.genome.fraction,
         ↪ tetraploid.genome.fraction=tetraploid.genome.fraction))) }

 if(fit.diploid){
  to.plot <- data.frame(VAF=seq(0.1, 1, 0.05), Data =
↪  mySumStatData$diploid*diploid.genome.fraction/(3.3*10^9),
             Mmin=apply(sim.diploid, 2, quantile,
             ↪  p=0.025)*diploid.genome.fraction/(3.3*10^9),
             Mmax=apply(sim.diploid, 2, quantile,
             ↪  p=0.975)*diploid.genome.fraction/(3.3*10^9))
  p[[length(p)+1]] <- eval(substitute(ggplot(to.plot, aes(x=VAF, y=Data,
                      ymin = Data - sqrt(Data),
                      ymax = Data + sqrt(Data)
                      )) +
   geom_ribbon(aes(ymin=Mmin,
           ymax=Mmax),
         fill="lightslateblue")+ geom_point() + geom_errorbar(width=0.01) +
         ↪ ggtitle(paste("CN=2, weight=",
         ↪ round(diploid.genome.fraction/(haploid.genome.fraction +
         ↪ diploid.genome.fraction + triploid.genome.fraction +
         ↪ tetraploid.genome.fraction), digits=2))) +
         ↪ scale_y_continuous(limits=c(0, y.max), name="Cumulative number of
         ↪ SNVs"),list(haploid.genome.fraction=haploid.genome.fraction,
         ↪ diploid.genome.fraction=diploid.genome.fraction,
         ↪ triploid.genome.fraction=triploid.genome.fraction,
         ↪ tetraploid.genome.fraction=tetraploid.genome.fraction))) }

 if(fit.triploid){
  to.plot <- data.frame(VAF=seq(0.1, 1, 0.05), Data =
↪  mySumStatData$triploid*triploid.genome.fraction/(3.3*10^9),
                          Mmin=apply(sim.triploid, 2, quantile,
                           ↪  p=0.025)*triploid.genome.fraction/(3.3*10^9),
                          Mmax=apply(sim.triploid, 2, quantile,
                           ↪  p=0.975)*triploid.genome.fraction/(3.3*10^9))
    p[[length(p)+1]] <- eval(substitute(ggplot(to.plot, aes(x=VAF, y=Data, ymin = Data -
↪   sqrt(Data), ymax = Data + sqrt(Data))) +
      geom_ribbon(aes(ymin=Mmin, ymax=Mmax),
                  fill="lightslateblue")+ geom_point() + geom_errorbar(width=0.01)  +
                  ↪  ggtitle(paste("CN=3, weight=",
                  ↪  round(triploid.genome.fraction/(haploid.genome.fraction +
                  ↪  diploid.genome.fraction + triploid.genome.fraction +
                  ↪  tetraploid.genome.fraction), digits=2))) +
                  ↪  scale_y_continuous(limits=c(0, y.max), name="Cumulative number of
                  ↪  SNVs"),
                  ↪  list(haploid.genome.fraction=haploid.genome.fraction,diploid.genome.fraction=dipl
                  ↪  triploid.genome.fraction=triploid.genome.fraction,
                  ↪  tetraploid.genome.fraction=tetraploid.genome.fraction))) }

  if(fit.tetraploid){
    to.plot <- data.frame(VAF=seq(0.1, 1, 0.05), Data =
↪   mySumStatData$tetraploid*tetraploid.genome.fraction/(3.3*10^9),
                          Mmin=apply(sim.tetraploid, 2, quantile,
                           ↪  p=0.025)*tetraploid.genome.fraction/(3.3*10^9),
                          Mmax=apply(sim.tetraploid, 2, quantile,
                           ↪  p=0.975)*tetraploid.genome.fraction/(3.3*10^9))
    p[[length(p)+1]] <- eval(substitute(ggplot(to.plot, aes(x=VAF, y=Data, ymin = Data -
↪   sqrt(Data), ymax = Data + sqrt(Data))) +
      geom_ribbon(aes( ymin=Mmin,  ymax=Mmax),fill="lightslateblue")+ geom_point() +
↪   geom_errorbar(width=0.01) + ggtitle(paste("CN=4, weight=",
↪   round(tetraploid.genome.fraction/(haploid.genome.fraction + diploid.genome.fraction +
↪   triploid.genome.fraction + tetraploid.genome.fraction), digits=2))) +
↪   scale_y_continuous(limits=c(0, y.max), name="Cumulative number of SNVs"),
↪   list(haploid.genome.fraction=haploid.genome.fraction,
↪   diploid.genome.fraction=diploid.genome.fraction,
↪   triploid.genome.fraction=triploid.genome.fraction,
↪   tetraploid.genome.fraction=tetraploid.genome.fraction))) }
```

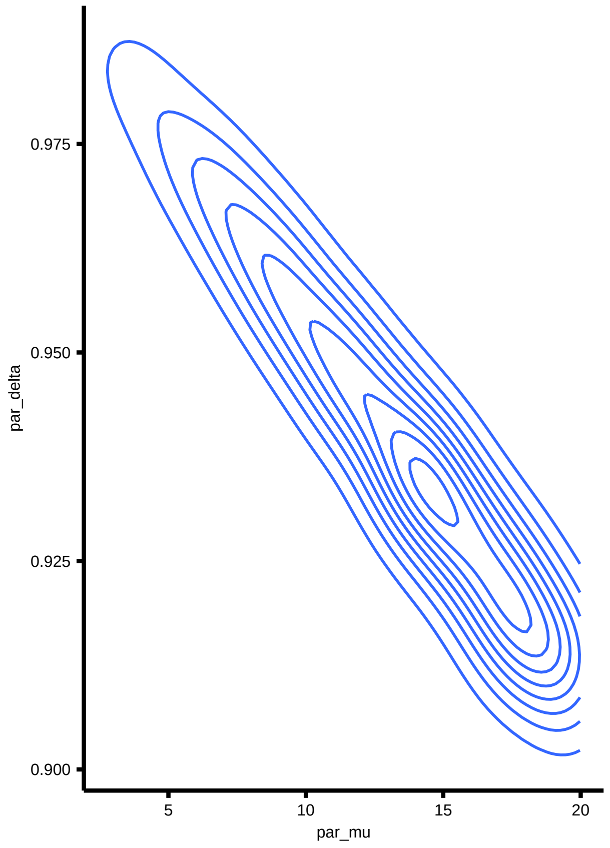
Let's look into the result

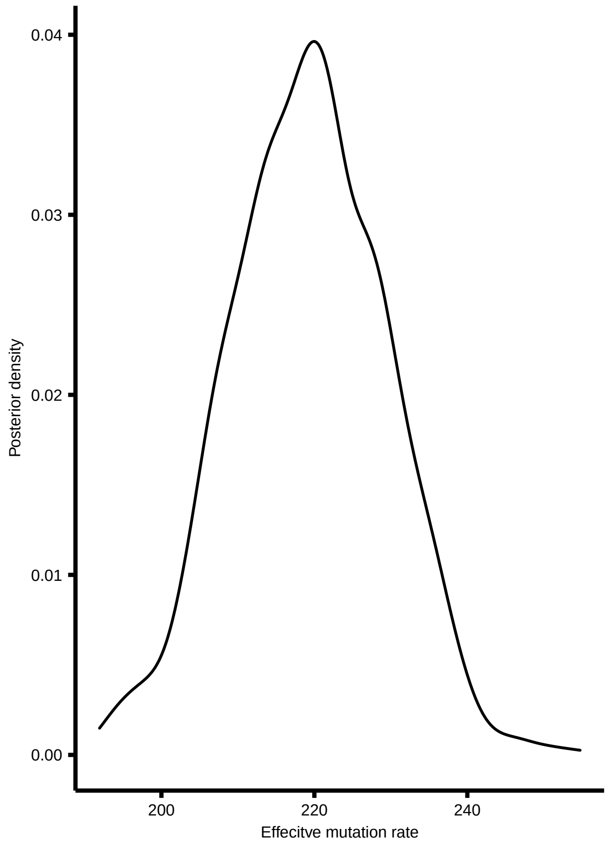
```
ggarrange(plotlist=p, nrow=2, ncol=2)
```



From the model fit, we cannot resolve mu and delta, but only the quotient thereof:

```r
p1 <- ggplot(fits, aes(x=par_mu, y=par_delta)) + geom_density_2d()

p2 <- ggplot(fits, aes(x=par_mu/(1-par_delta))) + geom_density()+
↪  scale_y_continuous("Posterior density") + scale_x_continuous(name="Effecitve mutation
↪  rate")

ggarrange(p1, p2, nrow=1, ncol=2)
```





Estimate mutation and division rates in actual time by combining the fitting results and using the age at diagnosis.

In the last step, we will now combine the model results from neruoblastoma initiation with the model results from neuroblastoma growth along with the age of the patients to translate the mutation and division rates into real time. To this end, we use the estimate for mu from the population genetics model of tumor initiation to estimate delta from the population genetics model of tumor growth. We then compute the number of generations between conception and diagnosis as the sum between the cell divisions until MRCA and the generations until growing a tumor of size 10^9 cells. Together with the age at diagnosis this yield estimates for mu and lambda in actual time for each analyzed patient. Eventually, we take the average across patients to reduce uncertainties in the individual estimates.

```r
## take the mutation rate from the model of tumor initiation and combine this with the
↪  model fits for tumor growth.
fits <- read.csv(paste0(fit.directory.initiation,
↪  "/Expansion_decay_continuous_evol.csv"))
mutation.rate <- c(mean(fits$par_mu*2), sd(fits$par_mu*2))

## subset of tumors used to learn the parameters of tumor growth
subset <- sample.information.discovery[
↪  sample.information.discovery$Use.to.fit.tumor.expansion=="T" ,]

## from these, extract estimates of the effective mutation rate (number of mutations per
↪  effective division) and subsequently of the loss rate
deltas <- c()
effective.mutation.rates <- c()
division.rate <- c()
```

```r
mutation.rate.per.day <- c()
for(i in rownames(subset)){

  if(!file.exists(paste0(fit.directory.growth, i, ".csv"))){
    print(i)
    next}

  fits <- read.csv(paste0(fit.directory.growth, i, ".csv"))

  ## in the growth model, the mutation rate is per 2 cells, but per haploid genome, thus
  ↪  take it as it is (*2/2=1) per cell
  ## The effective mutation rate is per effective division.
  ## store mean, sd
  effective.mutation.rate <- c(mean(fits$par_mu/(1-fits$par_delta)),
↪  sd(fits$par_mu/(1-fits$par_delta)))
  effective.mutation.rates <- cbind(effective.mutation.rates, effective.mutation.rate)
  colnames(effective.mutation.rates)[ncol(effective.mutation.rates)] <- i

  ## From this, compute delta; mean and sd
  delta <- 1 - mutation.rate[1]/effective.mutation.rate[1]
  ## we obtain the error by propagation of uncertainties
  delta[2] <- abs(1/effective.mutation.rate[1] *mutation.rate[2] +
↪  mutation.rate[1]/effective.mutation.rate[1]^2 *effective.mutation.rate[2])
  deltas <- cbind(deltas, delta)
  colnames(deltas)[ncol(deltas)] <- i

  mutational.burden.at.mrca <- mutation.time.mrca[i,]$Mean
  ## assume that mutation times are roughly normally distributed. Thus the standard
  ↪  deviation would correspond to 1/3.84 of the 95% CI
  mutational.burden.at.mrca[2] <- (mutation.time.mrca[i,]$Max -
↪  mutation.time.mrca[i,]$Min)/(2*1.96)
  age <- sample.information.discovery[rownames(sample.information.discovery)==i, "Age"]

  n.generations <- mutational.burden.at.mrca*2/mutation.rate[1] + 9*log(10)/(1-delta[1])
  n.generations[2] <- 2/mutation.rate[1]*mutational.burden.at.mrca[2] +
    mutational.burden.at.mrca[1]*2/mutation.rate[1]^2*mutation.rate[2] +
↪  9*log(10)/(1-delta[1])^2*delta[2]

  ## generations until MRCA
  n.generations.1 <- mutational.burden.at.mrca*2/mutation.rate[1]
  n.generations.1[2] <- 2/mutation.rate[1]*mutational.burden.at.mrca[2]+
    mutational.burden.at.mrca[1]*2/mutation.rate[1]^2*mutation.rate[2]
  ## generations during tumor growth
  n.generations.2 <- 9*log(10)/(1-delta[1])
  n.generations.2[2] <-  9*log(10)/(1-delta[1])^2*delta[2]


  ## t.total = age + pregnancy
  t.total <- age + 250
  ## initiation is the number of generations until tumor initiation divided by the total
  ↪  number of generations times the total time
  t.init <- mutational.burden.at.mrca*2/mutation.rate[1]/n.generations[1]*t.total
  t.init[2] <- t.total*(n.generations.1[1] * n.generations.2[1] +
↪  n.generations.2[1]*n.generations.1[2])/ (n.generations.1[1] + n.generations.2[1])^2
```

```r
  division.rate <- cbind(division.rate, c(n.generations[1]/(age + 250),
↪  n.generations[2]/(age + 250)))
  colnames(division.rate)[ncol(division.rate)] <- i

  ## the mutation rate per day is the product of the division rate and the mutation rate
  mut.per.day <-  division.rate[1]*mutation.rate[1]
  mut.per.day[2] <- division.rate[1]*mutation.rate[2] + division.rate[2]*mutation.rate[1]
  mutation.rate.per.day <-  cbind(mutation.rate.per.day, mut.per.day)

}

estimated.mutation.rate.per.day <- c(mean(mutation.rate.per.day[1,subset$Sample.type %in%
↪  c("Primary", "Metastasis")]),
                                     sqrt(sum((mutation.rate.per.day[2,subset$Sample.type
                                     ↪  %in% c("Primary", "Metastasis")])^2))/
                                       sum(subset$Sample.type %in% c("Primary",
↪  "Metastasis")))

estimated.mutation.rate.per.day <- c(estimated.mutation.rate.per.day[1] -
↪  1.96*estimated.mutation.rate.per.day[2],
                                     estimated.mutation.rate.per.day[1],
                                     estimated.mutation.rate.per.day[1] +
↪  1.96*estimated.mutation.rate.per.day[2])


median(effective.mutation.rates[1,])
```

```
## [1] 128.1901
```