sequenza usage example

Francesco Favero*, Aron C. Eklund

August 30, 2013

Contents

*favero@cbs.dtu.dk

1 Abstract

Deep sequence of tumor DNA along with corresponding normal DNA can provide a rich picture of the mutations and aberrations that characterize the tumor. However, analysis of this data can be impeded by of tumor cellularity and heterogeneity and by unwieldy data. Here we describe the *sequenza* software system, which comprises a fast python-based pre-processor and an R-based analysis package. Sequenza enables the efficient estimation of tumor cellularity and ploidy, and generation of copy number, loss-of-heterozygosity, and mutation frequency profiles.

This document details a typical analysis of matched tumor-normal exome sequence data using *sequenza*.

2 Starting data

Sequenza requires: 1. A pileup file from the tumor specimen 2. A pileup file from the normal specimen 3. A FASTA genome sequence file (optional, for GC-content correction)

Pileup files can be generated using `samtools` (ref). The genome sequence file can be obtained from (url).

3 First the non-R part: preprocessing data

For convenience and efficiency we have implemented preprocessing algorithms in an external (not called from R) Python program. The program is provided with the package; it's exact location can be found like this:

```
> system.file("exec", "sequenza-utils.py", package="sequenza")
```

```
[1] ""
```

You may wish to copy this program to a location on your path. NOTE: this script requires several unix tools and thus probably not work on Windows.

Extract average GC content in 50-base genomic windows:

```
#  abfreqtools.py GC-windows -w 50 hg19.fa | gzip > hg19.gc50Base.txt.gz
```

Process the two pileup files to obtain an "abfreq" file containing alleles and mutation frequency.

```
#  abfreqtools.py pileup2tab -gc hg19.gc50Base.txt.gz -r 0001-normal_blood.pileup.gz
-s 0001-met2.pileup.gz -q 20 -n 10 -o 0001-met2.abfreq.txt.gz
```

— UPDATE ME UPDATE ME UPDATE ME UPDATE ME UPDATE ME UPDATE ME —

4 Read the preprocessed data (*abfreq* file) into R

The remainder of this example takes place in R.

Load the sequenza package:

```
> library("sequenza")
```

Find the example data file:

```
> data.file <-  system.file("data", "abf.data.abfreq.txt.gz", package = "sequenza")
> data.file
```

```
[1] "/Library/Frameworks/R.framework/Versions/3.0/Resources/library/sequenza/data/abf.
```

The abfreq file can be read all at once, but processing one chromosome at a time is less demanding on computational resources and might be preferable. (Note that the demo data included with sequenza is only chromosome 1)

Read only the data corresponding to chromosome 1:

```
> abf.data <- read.abfreq(data.file, chr.name = "1")
```

Alternatively, read all data at once (not run):

```
> abf.data <- read.abfreq(data.file)
```

```
> str(abf.data)
```

```
'data.frame':        5349 obs. of  13 variables:
 $ chromosome                : Factor w/ 1 level "1": 1 1 1 1 1 1 1 1 1 1 ...
 $ n.base                    : int  13116 13118 13327 881918 884091 884101 900298 9005
 $ base.ref                  : Factor w/ 5 levels "A","C","G","N",..: 5 1 3 3 2 1 2 1
 $ depth.normal              : int  53 51 48 55 85 76 108 106 31 41 ...
 $ depth.sample              : int  33 33 27 37 65 59 78 72 14 16 ...
 $ depth.ratio               : num  0.623 0.647 0.563 0.673 0.765 0.776 0.722 0.679 0.
 $ Af                        : num  0.645 0.606 0.577 0.514 0.597 0.558 0.5 0.564 0.38
 $ Bf                        : num  0.355 0.394 0 0.486 0.387 0.442 0.5 0.436 0 0.4 ..
 $ ref.zygosity              : Factor w/ 2 levels "het","hom": 1 1 2 1 1 1 1 1 2 1 ...
 $ GC.percent                : num  58 58 60 64 70 58 70 66 82 64 ...
 $ sample.reads.above.quality: num  0.94 1 0.96 1 0.95 0.88 0.92 0.54 0.93 0.94 ...
 $ AB.germline               : Factor w/ 10 levels "A","AC","AG",..: 9 3 8 3 6 2 6 2 8
 $ AB.sample                 : Factor w/ 64 levels ".","A0.004:C0.623",..: 1 1 23 1 1
```

5 Quality control step? (EXPLAIN)

NEED TO EXPLAIN THIS:

```
> abf.data$good.s.reads <- abf.data$depth.sample *
+                          abf.data$sample.reads.above.quality
```

6 GC-normalization

The number of reads at a given genomic position can be affected by the local GC content. We attempt to remove this bias as in (ref).

Gather gc-content information from the entire file (normally this would be the entire genome, but in our example it contains only chromosome 1):

```
> gc.stats <- gc.sample.stats(data.file)
```

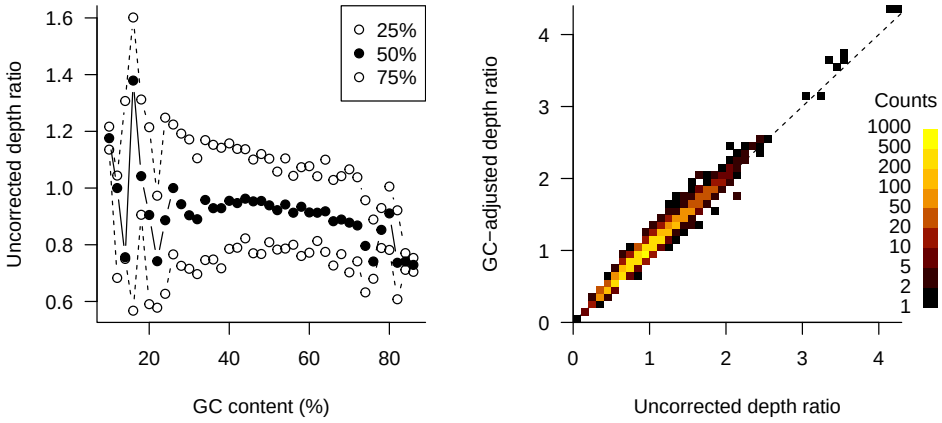
WHAT IS THIS? EXPLAIN... (not run)

```
> gc.stats <- gc.norm(ratio = abf.data$depth.ratio,
+                     gc = abf.data$GC.percent)
```

Calculate the GC-adjusted depth ratio:

```
> gc.vect  <- setNames(gc.stats$raw.mean, gc.stats$gc.values)
> abf.data$adjusted.ratio <- abf.data$depth.ratio /
+                            gc.vect[as.character(abf.data$GC.percent)]

> par(mfrow = c(1,2), cex = 1, las = 1, bty = 'l')
> matplot(gc.stats$gc.values, gc.stats$raw,
+         type = 'b', col = 1, pch = c(1, 19, 1), lty = c(2, 1, 2),
+         xlab = 'GC content (%)', ylab = 'Uncorrected depth ratio')
> legend('topright', legend = colnames(gc.stats$raw), pch = c(1, 19, 1))
> hist2(abf.data$depth.ratio, abf.data$adjusted.ratio,
+       breaks = prettyLog, key = vkey, panel.first = abline(0, 1, lty = 2),
+       xlab = 'Uncorrected depth ratio', ylab = 'GC-adjusted depth ratio')
```



7 Create genomic profiles

7.1 First, the depth ratio

Summarize the depth ratio in overlapping genomic windows:

```
> abf.r.win <- windowValues(x = abf.data$adjusted.ratio,
+                 positions = abf.data$n.base,
+                 chromosomes = abf.data$chromosome,
+                 window = 1e6, overlap = 1,
+                 weight = abf.data$depth.normal)
```

```
> plotWindows(abf.r.win[[1]], log2.plot = FALSE,
+             ylab = "Depth ratio", xlab = "Position (bases)",
+             main = names(abf.r.win)[1], las = 1, n.min = 1,
+             ylim = c(0, 2.5))
```

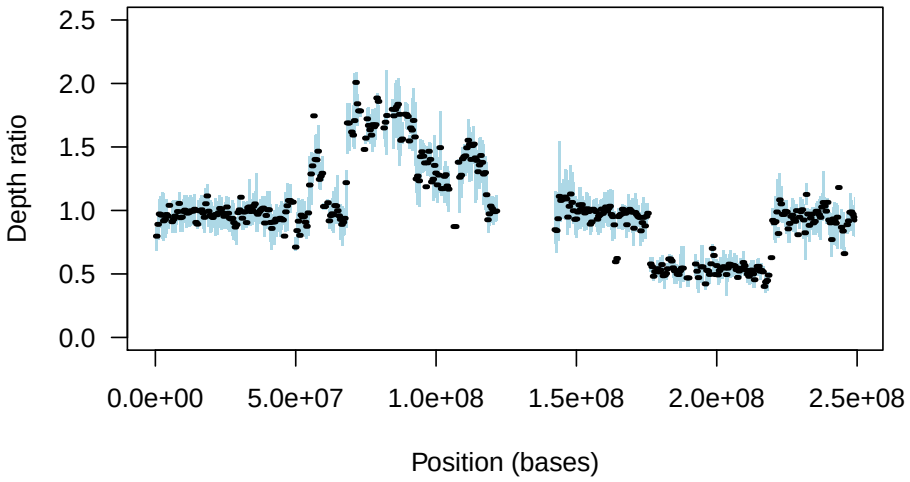


Figure 1: Depth ratio profile visualization over a single chromosome.

7.2 Next, the B-allele frequencies

```
> abf.hom  <- abf.data$ref.zygosity == 'hom'
> abf.het  <- abf.data[!abf.hom, ]
```

Summarize the BAF in overlapping genomic windows (including only those positions called heterozygous in the normal sample):

```
> abf.b.win <- windowValues(x = abf.het$Bf,
+              positions = abf.het$n.base,
+              chromosomes = abf.het$chromosome,
+              window = 1e6, overlap = 1,
+              weight = round(x = abf.het$good.s.reads, digits = 0))
```

```
> plotWindows(abf.b.win[[1]], ylim = c(0, 0.5),
+             main = names(abf.r.win)[1], xlab = "Position (bases)",
+             ylab = "B allele frequency", n.min = 10)
```

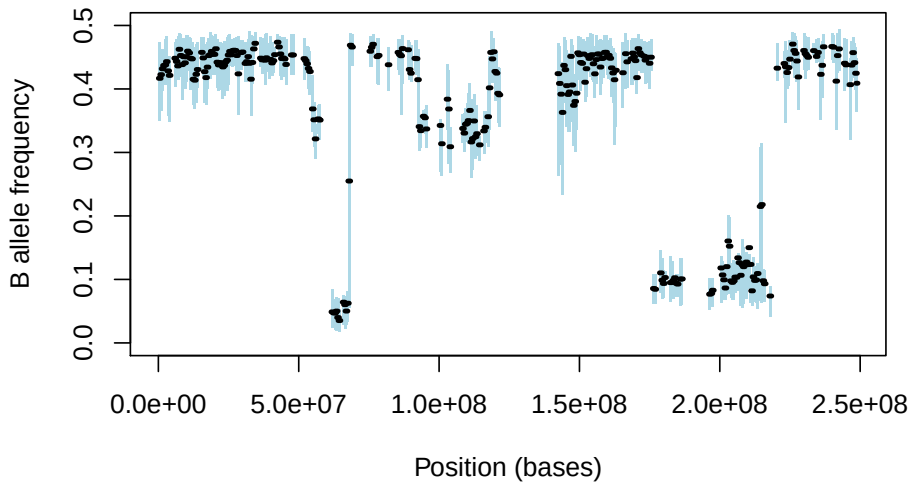


Figure 2: B-allele frequency profile visualization over a single chromosome.

8 Allele-specific segmentation

8.1 Find genomic breakpoints

To find breakpoints we use the allele-specific segmentation algorithm from the *copynumber* package [1].

```
> breaks <- find.breaks(abf.het, gamma = 80, kmin = 10, baf.thres = c(0, 0.5))
> head(breaks)
```

```
  chrom start.pos   end.pos
1     1     13116  17013363
2     1  17013750  55100328
3     1  55183366  60381491
4     1  61743160  67960720
5     1  68151685  92445257
6     1  92568263 118165328
```

Now obtain the segment values:

```
> seg.s1 <- segment.breaks(abf.data, breaks = breaks)
```

9 Select mutations by mutation frequency

```
> mut.tab <- mutation.table(abf.data, mufreq.treshold = 0.15,
+                           min.reads = 40, max.mut.types = 1,
+                           min.type.freq = 0.9, segments = seg.s1)
```

NEEDS EXPLANATION HERE... what is the following object?

```
> mut.tab.no.seg <- mutation.table(abf.data, mufreq.treshold = 0.15,
+                           min.reads = 40, max.mut.types = 1,
+                           min.type.freq = 0.9)

> dim(mut.tab)

[1] 22  7

> head(mut.tab)

     chromosome   n.base GC.percent good.s.reads adjusted.ratio     F mutation
286           1 10436585         50       207.58      0.9982462 0.382      C>T
496           1 13111750         44        48.02      0.9982462 0.417      A>C
637           1 15821826         54       497.97      0.9982462 0.398      G>T
1077          1 19983391         72        77.08      0.9813842 0.558      G>C
1364          1 26878353         60        50.00      0.9813842 0.520      C>A
1504          1 32627966         54       120.78      0.9813842 0.413      C>T

> head(mut.tab.no.seg)

     chromosome   n.base GC.percent good.s.reads adjusted.ratio     F mutation
286           1 10436585         50       207.58      1.0785423 0.382      C>T
496           1 13111750         44        48.02      1.2353979 0.417      A>C
637           1 15821826         54       497.97      0.8704422 0.398      G>T
1077          1 19983391         72        77.08      0.9201950 0.558      G>C
1364          1 26878353         60        50.00      1.2760504 0.520      C>A
1504          1 32627966         54       120.78      1.0526037 0.413      C>T
```

10 Plot chromosome view with mutations, BAF, depth ratio and segments

```
> chromosome.view(mut.tab = mut.tab, baf.windows = abf.b.win[[1]],
+                 ratio.windows = abf.r.win[[1]], min.N.ratio = 1,
+                 segments = seg.s1, main = "Chromosome 1")
```

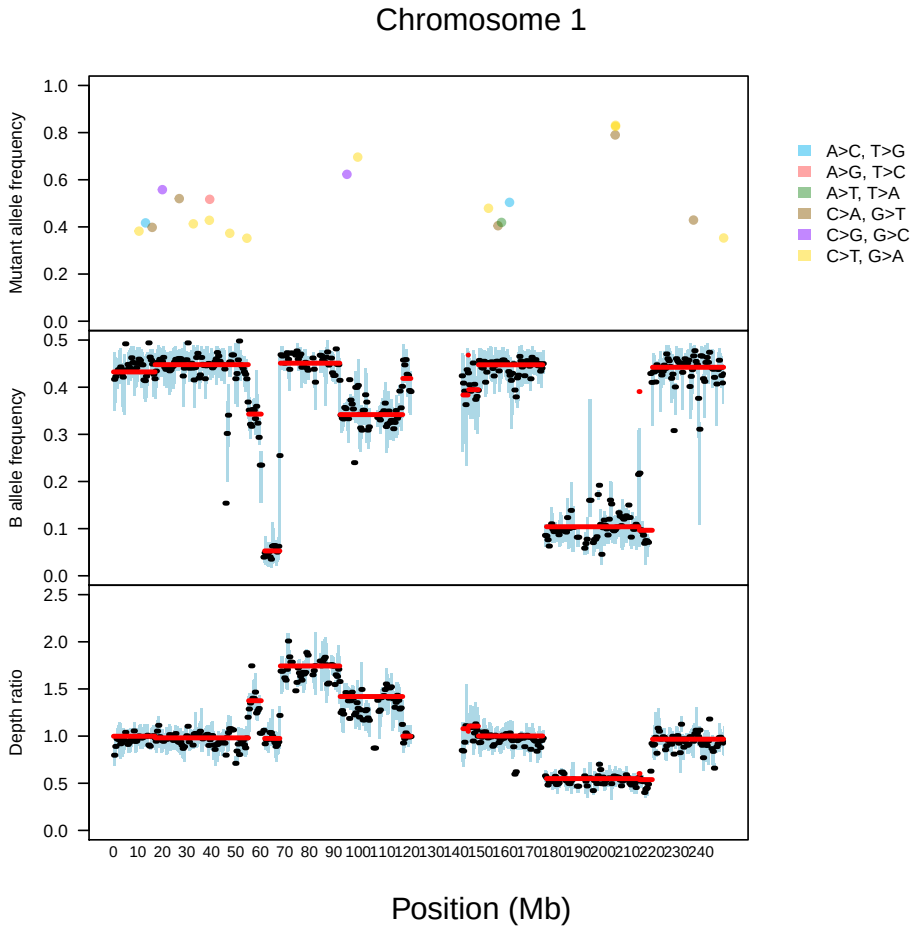


Figure 3: Plots of Mutation (top), B-allele frequencies (middle) and depth ratio (bottom) for chromosome position.

11 Inference of cellularity and DNA-index

NEEDS EXPLANATION:

```
> seg.filtered <- seg.s1[(seg.s1$end.pos - seg.s1$start.pos) > 5e6, ]
```

NEEDS EXPLANATION:

```
> weights.seg  <- 150 + round((seg.filtered$end.pos - seg.filtered$start.pos) / 1e6,
```

```
> avg.depth.ratio <- mean(gc.stats$adj[,3])
> avg.depth.ratio
```

```
[1] 1.202572
```

I DON'T UNDERSTAND WHY "avg.depth.ratio = 1" when we have calculated a different number above?:

```
> CP <- baf.model.fit(Bf = seg.filtered$Bf, depth.ratio = seg.filtered$depth.ratio,
+                     weight.ratio = weights.seg,
+                     weight.Bf = weights.seg,
+                     avg.depth.ratio = 1,
+                     cellularity = seq(0.1,1,0.01),
+                     dna.index = seq(0.5,3,0.05), mc.cores = 4)
```

WOULD IT MAKE MORE SENSE FOR THIS FUNCTION TO RETURN A MATRIX?

NEED EXPLANATIONS HERE:

```
> cint <- get.ci(CP)
```

```
> cp.plot(CP)
> cp.plot.contours(CP, add = TRUE, likThresh = c(0.5, 0.75, 0.95, 0.99))
```

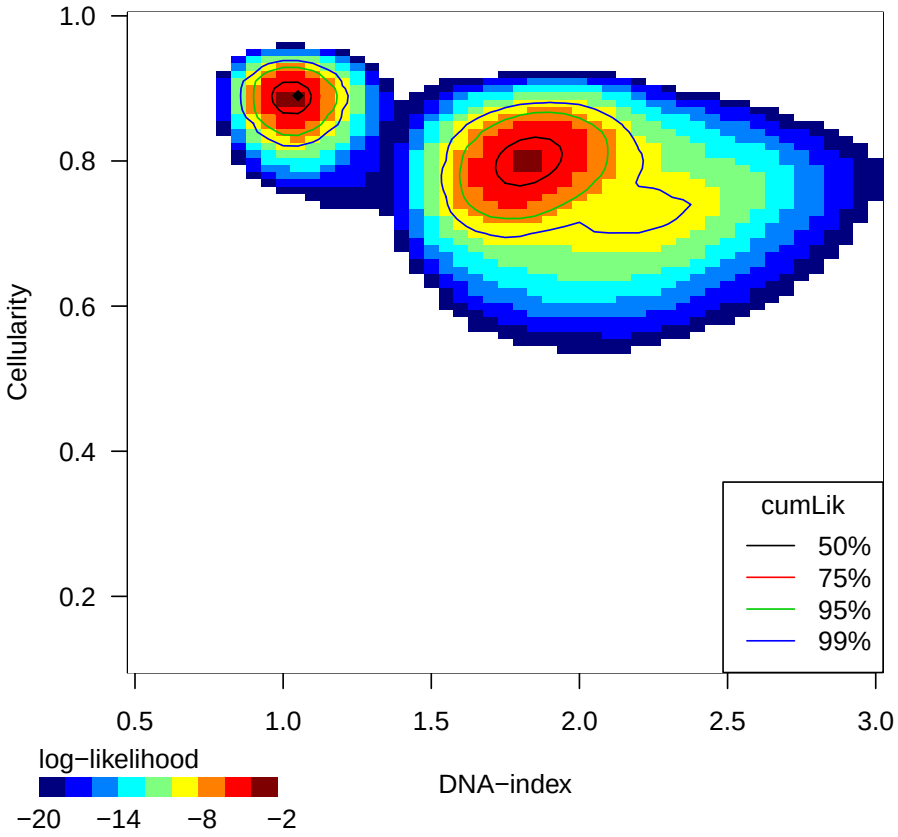


Figure 4: Result from the Bayesian inference over the defined range of cellularity and DNA-index. The color indicates the log-likelihood of the corresponding cellularity/DNA-index values.

```
> par(mfrow = c(2,2))
> cp.plot(CP)
> plot(cint$values.y, ylab = "Cellularity",
+      xlab = "likelihood", type = "n")
> select <- cint$confint.y[1] <= cint$values.y[,2] & cint$values.y[,2] <= cint$confi
> polygon(y = c(cint$confint.y[1], cint$values.y[select, 2], cint$confint.y[2]),
+         x = c(0, cint$values.y[select, 1], 0), col='red', border=NA)
> lines(cint$values.y)
> abline(h = cint$max.y, lty = 2, lwd = 0.5)
> plot(cint$values.x, xlab = "DNA index",
+      ylab = "likelihood", type = "n")
> select <- cint$confint.x[1] <= cint$values.x[,1] & cint$values.x[,1] <= cint$confi
> polygon(x = c(cint$confint.x[1], cint$values.x[select, 1], cint$confint.x[2]),
+         y = c(0, cint$values.x[select, 2], 0), col='red', border=NA)
> lines(cint$values.x)
> abline(v = cint$max.x, lty = 2, lwd = 0.5)
>
```

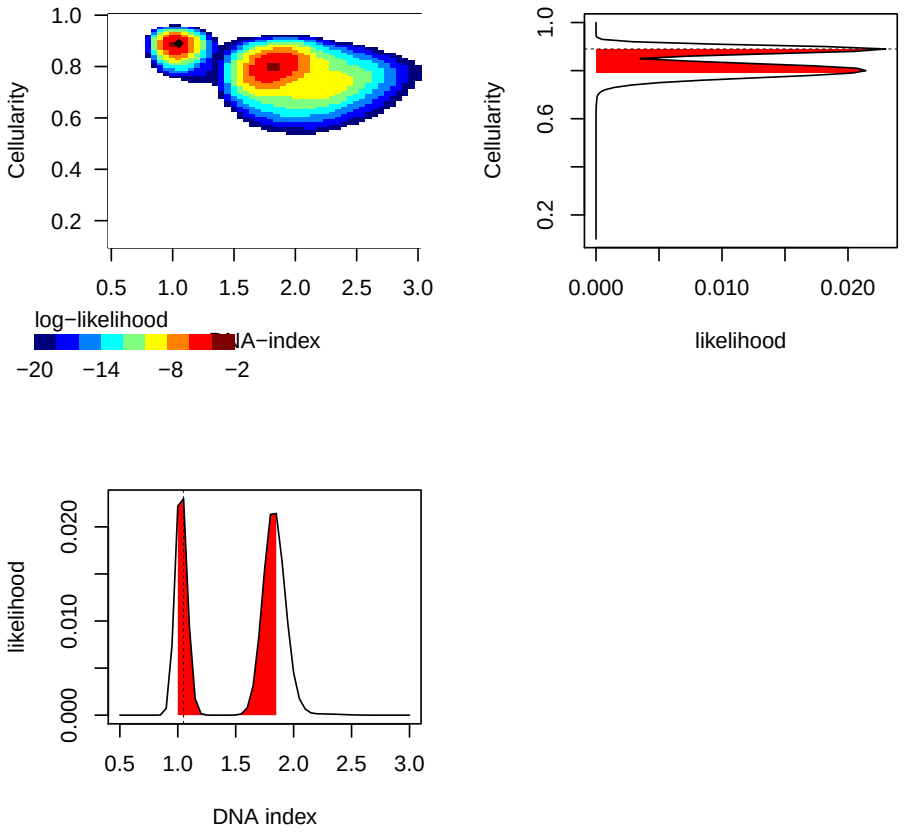


Figure 5: Plot of the log likelihood with respective cellularity and DNA-index probability distribution and confidence intervals.

12 Call CNVs and mutations using the estimated parameters

```
> cellularity <- cint$max.y
> cellularity

[1] 0.89

> dna.index <- cint$max.x
> dna.index

[1] 1.05
```

12.1 Detect mutated alleles

```
> mut.tab.clean <- na.exclude(mut.tab)
> mut.alleles <- mufreq.bayes(mufreq = mut.tab.clean$F, depth.ratio = mut.tab.clean$
+                             cellularity = cellularity, dna.index = dna.index,
+                             avg.depth.ratio = 1)
> head(mut.alleles)

   CNr CNt Mt         L
4    2   2  1 -13.68204
41   2   2  1 -12.63950
42   2   2  1 -13.04180
43   2   2  1 -16.29588
44   2   2  1 -14.10459
45   2   2  1 -12.81114

> head(cbind(mut.tab.clean[,c("chromosome","n.base","F","adjusted.ratio", "mutation"

     chromosome   n.base     F adjusted.ratio mutation CNr CNt Mt         L
286           1 10436585 0.382      0.9982462      C>T   2   2  1 -13.68204
496           1 13111750 0.417      0.9982462      A>C   2   2  1 -12.63950
637           1 15821826 0.398      0.9982462      G>T   2   2  1 -13.04180
1077          1 19983391 0.558      0.9813842      G>C   2   2  1 -16.29588
1364          1 26878353 0.520      0.9813842      C>A   2   2  1 -14.10459
1504          1 32627966 0.413      0.9813842      C>T   2   2  1 -12.81114
```

12.2 Detect Copy number variation

```
> cn.alleles <- baf.bayes(Bf = seg.s1$Bf, depth.ratio = seg.s1$depth.ratio,
+                         cellularity = cellularity, dna.index = dna.index,
```

```
+                         avg.depth.ratio = 1)
> seg.s1.cn <- cbind(seg.s1, cn.alleles)
> head(seg.s1.cn)

  chromosome start.pos   end.pos         Bf N.BAF depth.ratio N.ratio CNt A B
1          1     13116  17013363 0.43238002   854   0.9982462     866   2 1 1
2          1  17013750  55100328 0.44782729  1056   0.9813842    1072   2 1 1
3          1  55183366  60381491 0.34292696    68   1.3751212      68   3 2 1
4          1  61743160  67960720 0.05276295   120   0.9751264     122   2 2 0
5          1  68151685  92445257 0.45096790   262   1.7436966     265   4 2 2
6          1  92568263 118165328 0.34186189   348   1.4197246     355   3 2 1
         L
1 -12.07834
2 -11.75661
3 -11.54033
4 -10.48384
5 -11.65777
6 -11.56998
```

13 Visualize detected copy number

```
> chromosome.view(mut.tab = mut.tab, baf.windows = abf.b.win[[1]],
+                 ratio.windows = abf.r.win[[1]],  min.N.ratio = 1,
+                 segments = seg.s1.cn, main = "Chromosome 1",
+                 cellularity = cellularity, dna.index = dna.index,
+                 avg.depth.ratio = 1)
```

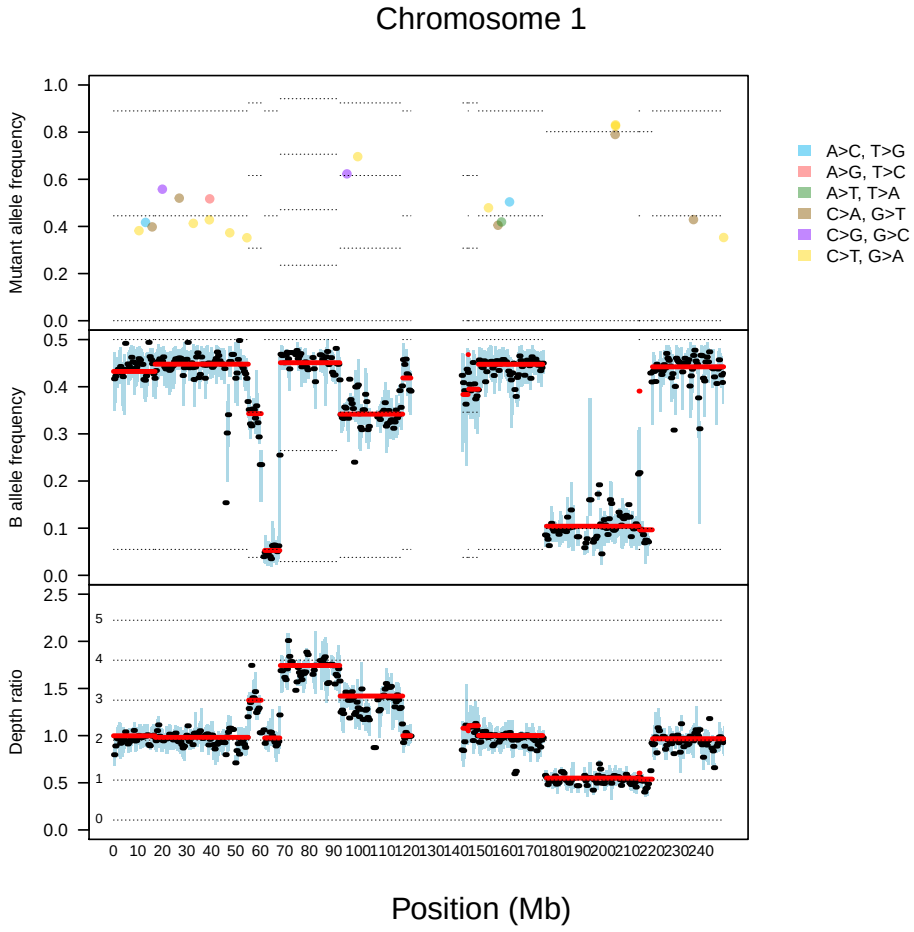


Figure 6: Plots of Mutation (top), B-allele frequencies (middle) and depth ratio (bottom) for chromosome position. Horizontal dotted line indicate different copy number/allelic state.

References

[1] Gro Nilsen, Knut Liestø l, Peter Van Loo, Hans Kristian Moen Vollan, Marianne B Eide, Oscar M Rueda, Suet-Feung Chin, Roslin Russell, Lars O Baumbusch, Carlos

Caldas, Anne-Lise Bø rresen Dale, and Ole Christian Lingjaerde. Copynumber: Efficient algorithms for single- and multi-track copy number segmentation. *BMC genomics*, 13:591, January 2012.