1 Introduction

Tumors often consist of multiple distinct subpopulations or clones. Information about the number of clones present in a tumor can be inferred using either mutation allele frequency data, from sequencing studies, or from copy number variants (CNVs), derived either from sequencing or from SNP array data. The \texttt{CloneSeeker} package can be applied to SNP array data, sequencing data, or both, from tumor cells from a cancer patient. \texttt{CloneSeeker} can determine the number of clones, the distribution of cells among clones, and the copy number variations and mutations (depending on the available data sources) that occur in each clone. The presence of multiple detectable clones is called “clonal heterogeneity” in the literature.

Clonal heterogeneity likely plays an important role in the clinical course of a cancer. It is possible, for example, that the tumor cells that will eventually become the refractory cancer after treatment are present as a minor subclone in the tumor early on.

First, we load the \texttt{CloneSeeker} package:

\begin{verbatim}
> library(CloneSeeker)
\end{verbatim}

2 Simulated Tumor Containing Multiple Clones

In order to illustrate the algorithms, we are going to simulate data where we know the true structure. Specifically, we will simulate copy number and mutation data for a tumor with three clones. We start with an object that represents the Tumor at a somewhat abstract level.
```r
> set.seed(21303) # for reproducibility
> simTumor <- Tumor(c(5, 3, 2), rounds = 100,
+              nu = 10, pcnv = 0.8, norm.contam = FALSE)
```

The first argument to the `Tumor` constructor is a vector that specifies the relative proportions of cells belonging to each clone; the length of the vector determines the number of clones. These values are automatically converted to fractions that add up to one:

```r
> simTumor@psi
```

An object of class "WeightVector"
Slot "psi":
[1] 0.5 0.3 0.2

The second argument, `rounds`, specifies the number of generations through which the tumor clones are evolved. The idea is that new abnormalities, either in the form of mutations or copy number variants (CNVs), are acquired at each evolutionary step from some parent cell. The parameter `nu` is the expected number of new mutations and the parameter `pcnv` is the probability of a new CNV at each step. The final parameter, `norm.contam`, is a logical indicator of whether the tumor sample is assumed to include a subset of cells that represent non-cancerous "normal contamination".

The resulting simulated tumor contains descriptions of each individual clone. In the current implementation, these are stored as a list of clones.

```r
> class(simTumor@clones)
[1] "list"
> length(simTumor@clones)
[1] 3
```

Individual clones contain descriptions of both CNVs and mutations.

```r
> oneClone <- simTumor@clones[[1]]
> class(oneClone)
[1] "list"
> length(oneClone)
[1] 2
> names(oneClone)
[1] "cn" "seq"
```

The copy number data includes the chromosome, with start and end positions, the number of copies of the A and B alleles, an arbitrary “segment” identifier, and (as a residual from the simulated evolutionary history), a “parent” identifier.
> dim(oneClone$cn)

[1] 320  7

> summary(oneClone$cn)

<table>
<thead>
<tr>
<th></th>
<th>chr</th>
<th>start</th>
<th>end</th>
<th>A</th>
</tr>
</thead>
<tbody>
<tr>
<td>Min.</td>
<td>1.000</td>
<td>Min.</td>
<td>1</td>
<td>512228</td>
</tr>
<tr>
<td>1st Qu.:</td>
<td>4.000</td>
<td>1st Qu.:</td>
<td>41806795</td>
<td>1st Qu.:</td>
</tr>
<tr>
<td>Median :</td>
<td>9.000</td>
<td>Median :</td>
<td>117639946</td>
<td>Median :</td>
</tr>
<tr>
<td>Mean :</td>
<td>9.756</td>
<td>Mean :</td>
<td>114603107</td>
<td>Mean :</td>
</tr>
<tr>
<td>3rd Qu.:</td>
<td>15.000</td>
<td>3rd Qu.:</td>
<td>181197780</td>
<td>3rd Qu.:</td>
</tr>
<tr>
<td>Max. :</td>
<td>24.000</td>
<td>Max. :</td>
<td>248891168</td>
<td>Max. :</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>B</th>
<th>seg</th>
<th>parent.index</th>
</tr>
</thead>
<tbody>
<tr>
<td>Min.</td>
<td>1.000</td>
<td>Min.</td>
<td>1.00</td>
</tr>
<tr>
<td>1st Qu.:</td>
<td>1.000</td>
<td>1st Qu.:</td>
<td>80.75</td>
</tr>
<tr>
<td>Median :</td>
<td>1.000</td>
<td>Median :</td>
<td>160.50</td>
</tr>
<tr>
<td>Mean :</td>
<td>1.003</td>
<td>Mean :</td>
<td>160.50</td>
</tr>
<tr>
<td>3rd Qu.:</td>
<td>1.000</td>
<td>3rd Qu.:</td>
<td>240.25</td>
</tr>
<tr>
<td>Max. :</td>
<td>2.000</td>
<td>Max. :</td>
<td>320.00</td>
</tr>
</tbody>
</table>

The mutation data has a chromosomal location, arbitrary segment and mutation identifiers, the number of mutated and wild type copies for each mutation, and the affected allele.

> dim(oneClone$seq)

[1] 13  7

> oneClone$seq

<table>
<thead>
<tr>
<th></th>
<th>chr</th>
<th>start</th>
<th>seg</th>
<th>mut.id</th>
<th>mutated.copies</th>
<th>allele</th>
<th>normal.copies</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>31289374</td>
<td>7</td>
<td>180</td>
<td>1</td>
<td>B</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>138387098</td>
<td>38</td>
<td>181</td>
<td>1</td>
<td>A</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>47508817</td>
<td>57</td>
<td>1</td>
<td>1</td>
<td>B</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>5</td>
<td>49397981</td>
<td>100</td>
<td>2</td>
<td>1</td>
<td>A</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>107243459</td>
<td>102</td>
<td>3</td>
<td>1</td>
<td>A</td>
<td>1</td>
</tr>
<tr>
<td>6</td>
<td>5</td>
<td>141225198</td>
<td>105</td>
<td>4</td>
<td>1</td>
<td>B</td>
<td>1</td>
</tr>
<tr>
<td>7</td>
<td>6</td>
<td>165607219</td>
<td>122</td>
<td>5</td>
<td>1</td>
<td>B</td>
<td>1</td>
</tr>
<tr>
<td>8</td>
<td>7</td>
<td>123361439</td>
<td>135</td>
<td>6</td>
<td>1</td>
<td>B</td>
<td>1</td>
</tr>
<tr>
<td>9</td>
<td>12</td>
<td>27339877</td>
<td>201</td>
<td>182</td>
<td>1</td>
<td>B</td>
<td>1</td>
</tr>
<tr>
<td>10</td>
<td>15</td>
<td>60248345</td>
<td>241</td>
<td>7</td>
<td>1</td>
<td>B</td>
<td>1</td>
</tr>
<tr>
<td>11</td>
<td>18</td>
<td>9110856</td>
<td>267</td>
<td>183</td>
<td>1</td>
<td>A</td>
<td>1</td>
</tr>
<tr>
<td>12</td>
<td>22</td>
<td>15038326</td>
<td>295</td>
<td>8</td>
<td>1</td>
<td>B</td>
<td>1</td>
</tr>
<tr>
<td>13</td>
<td>24</td>
<td>40007975</td>
<td>317</td>
<td>184</td>
<td>1</td>
<td>A</td>
<td>1</td>
</tr>
</tbody>
</table>

2.1 Simulating Tumor Data

Now that we have the tumor in place, we can simulate data arising from a study of that tumor.
> simData <- generateTumorData(simTumor,
+   snps.seq = 10000,
+   snps.cgh = 600000,
+   mu = 70,
+   sigma.reads = 25,
+   sigma0.lrr = 0.15,
+   sigma0.baf = 0.03,
+   density.sigma = 0.1)

For a description of the many parameters to the `generateTumorData` function, see the man page. The first two arguments are size parameters. The first, `snps.seq`, determines the number of germline variants to simulate; in the absence of separate copy number data, these are used to provide a crude estimate. The second, `snps.cgh`, represents the number of SNP locations on the simulated SNP chip from which copy number segments are derived. The remaining parameters control the simulated read depth and variability.

As with individual clones, the simulated data is structured as a list with separate data frames for the CNVs and mutations.

> class(simData)
[1] "list"

> length(simData)
[1] 2

> names(simData)
[1] "cn.data"  "seq.data"

The simulated copy number data includes chromosomal locations along with estimated log R ratios (LRR), B allele frequencies (BAF), separate intensity values for the two parental alleles (X and Y), and the number of SNPs in each segment (markers).

> cnDat <- simData$cn.data
> dim(cnDat)
[1] 320  7

> summary(cnDat)

<table>
<thead>
<tr>
<th>chr</th>
<th>seg</th>
<th>LRR</th>
<th>BAF</th>
</tr>
</thead>
<tbody>
<tr>
<td>Min</td>
<td>1.00</td>
<td>Min. &lt;-0.1268032</td>
<td>Min. :0.4355</td>
</tr>
<tr>
<td>1st Qu.</td>
<td>4.000</td>
<td>1st Qu.:-0.0024181</td>
<td>1st Qu.:0.4996</td>
</tr>
<tr>
<td>Median</td>
<td>9.000</td>
<td>Median :160.50</td>
<td>Median :0.5000</td>
</tr>
<tr>
<td>Mean</td>
<td>9.756</td>
<td>Mean :0.0003409</td>
<td>Mean :0.5010</td>
</tr>
<tr>
<td>3rd Qu.:15.000</td>
<td>3rd Qu.:240.25</td>
<td>3rd Qu.:0.0027016</td>
<td>3rd Qu.:0.5005</td>
</tr>
<tr>
<td>Max</td>
<td>24.000</td>
<td>Max. :320.00</td>
<td>Max. :0.1739851</td>
</tr>
</tbody>
</table>

X

Y markers
The simulated sequencing data, in addition to chromosomal locations, has read counts for the number of reference alleles, alternate (meaning variant or mutated) alleles, total counts, the variant allele frequency (VAF), and a status indicator of whether the variant is believed to be germline or somatic.

```r
> dim(simData$seq.data)
[1] 10026  8

> seqDat <- simData$seq.data
> somatic <- seqDat[seqDat$status=='somatic',]
> dim(seqDat)
[1] 10026  8

> summary(seqDat)

chr    seg  mut.id  refCounts
  Min. : 1.000 Min. : 1.0 Min. : 1.00 Min. : 24.00
  1st Qu.: 4.000 1st Qu.: 80.0 1st Qu.: 20.50 1st Qu.: 60.00
  Median : 9.000 Median :160.0 Median : 95.10 Median : 70.00
  Mean : 9.681 Mean :159.6 Mean :260.38 Mean : 69.94
  3rd Qu.:14.000 3rd Qu.:237.0 3rd Qu.:183.75 3rd Qu.: 79.00
  Max. :24.000 Max. :320.0 Max. :951.00 Max. :166.00

varCounts  VAF  totalCounts  status
  Min. : 13.0 Min. :0.09783 Min. : 53.0 Length:10026
  1st Qu.: 60.0 1st Qu.:0.46988 1st Qu.:123.0 Class :character
  Median : 70.0 Median :0.50000 Median :140.0 Mode :character
  Mean : 69.8 Mean :0.49972 Mean :139.7
  3rd Qu.: 79.0 3rd Qu.:0.52991 3rd Qu.:156.0
  Max. :125.0 Max. :0.74336 Max. :238.0

> table(seqDat$status)
  germline somatic
     10000      26
```

3 Seeking Clones

To run CloneSeeker, we will need a starting set of \( \psi \) vectors as inputs, where \( \psi \) records the fraction of cells belonging to each clone. For each \( \psi \) vector, the algorithm will compute the most
probable copy number state for each clone at each segment. The maximum posterior probability is computed for each input \( \psi \) vector, and these probabilities are used to resample new potential \( \psi \) vectors. We usually start by considering every possible decomposition of the tumor into five clones, where the fraction assigned to each clone is a multiple of \( 1/20 = 0.05 \). We can generate this initial matrix of \( \psi \) vectors as follows:

\[
> \text{psis} <- \text{generateSimplex}(20, 5)
> \text{dim(psis)}
\]

\[
[1] 192 5
\]

\[
> \text{head(psis)}
\]

\[
[1,] 1.00 0.00 0.00 0 0
[2,] 0.95 0.05 0.00 0 0
[3,] 0.90 0.10 0.00 0 0
[4,] 0.90 0.05 0.05 0 0
[5,] 0.85 0.15 0.00 0 0
[6,] 0.85 0.10 0.05 0 0
\]

\[
> \text{tail(psis)}
\]

\[
[187,] 0.25 0.25 0.25 0.20 0.05
[188,] 0.25 0.25 0.25 0.15 0.10
[189,] 0.25 0.25 0.20 0.20 0.10
[190,] 0.25 0.25 0.20 0.15 0.15
[191,] 0.25 0.20 0.20 0.20 0.15
[192,] 0.20 0.20 0.20 0.20 0.20
\]

For SNP array data, we also need, as input, a set of possible clonal segment copy number states. If none exists the function will automatically generate one. The version used here considers all possible copy number states from 0 to 5 copies, but it imposes a strong prior belief that two different clones cannot both gain and lose the same segment.

\[
> \text{cnmodels} <- \text{expand.grid(rep(list(0:5),5))}
> \text{include} <- \text{sapply}(1:nrow(cnmodels), function(i) {
+   \text{length(which(cnmodels[i,] >= 1))==5} \text{ | length(which(cnmodels[i,] <= 1)) == 5}
+ })
> \text{cnmodels} <- \text{cnmodels[include,]}
\]

Now we will define the other algorithm parameters:

\[
> \text{pars} <- \text{list(sigma0 = 1, \# SNP-wise standard deviation}
+   \text{ktheta = 0.3, \# geometric prior parameter on number of clones}
+   \text{theta = 0.9, \# geometric prior parameter on copy number changes}
+   \text{mtheta = 0.9, \# geometric prior parameter on point mutations}
\]

6
3.1 Seeking Clones from Copy Number Data

The seekClones function can estimate the clonal architecture from copy number data, or from mutation and variant data, or jointly from both kinds of data. In this section, we will run the algorithm using only the copy number data. To do that, we set the varData argument to NULL.

```r
> resCN <- seekClones(cndata = cnDat, varData = NULL,
+ cnmodels = cnmodels, psiset = psis, pars = pars)
```

Here are the results of the “CNV only” analysis of this sample:

```r
> resCN$psi

[1] 0.5 0.3 0.2 0.0 0.0

> simTumor@psi

An object of class "WeightVector"
Slot "psi":
[1] 0.5 0.3 0.2
```

In this case, CloneSeeker accurately estimates not only the number of clones but also the clonal fractions. Let’s look at the clonal copy number assignments as well:

```r
> trueCN_Assignments <- t(sapply(1:nrow(resCN$filtered.data$cndata.filt),
+ function(i) {
+ index <- rownames(simTumor@clones[[1]]$cn) ==
+ rownames(resCN$filtered.data$cndata.filt)[i]
+ sapply(1:length(simTumor@clones),function(j){
+ simTumor@clones[[j]]$cnA[index] + simTumor@clones[[j]]$cnB[index]
+ })
+ })
> inferredCN_Assignments <- (resCN$A+resCN$B)[1:length(simTumor@clones)]
> colnames(inferredCN_Assignments) <- colnames(trueCN_Assignments) <-
+ paste("C", 1:3)
> data.frame(Truth = trueCN_Assignments,
+ Infer = inferredCN_Assignments)

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>22</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>24</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>2</td>
</tr>
</tbody>
</table>
```

7
Although not perfect, the algorithm managed to correctly estimate most of the segment-wise allelic copy numbers of different clones.

### 3.2 Sequencing Data

Now, let’s illustrate the use of CloneSeeker in analyzing mutation data (by which we mean variant data such as one would find in a .vcf file) to seek clones. This time, we run the CloneSeeker algorithm with the `cndata` argument set to `NULL`.

```r
> resMut <- seekClones(cndata = NULL, vardata = seqDat, + cnmodels = cnmodels, psiset = psis, pars = pars)
```

Here the results aren’t as good; at least one of the actual clones has been split into separate pieces.

```r
> resMut$psi
[1] 0.53443247 0.19437358 0.10946443 0.10387835 0.05785117
```

```r
> simTumor@psi
An object of class "WeightVector"
Slot "psi":
[1] 0.5 0.3 0.2
```

### 3.3 Both Sequencing and SNP Array Data

Finally, we illustrate running CloneSeeker on a sample for which there is both SNP array and mutation data.

```r
> resBoth <- seekClones(cndata = cnDat, vardata = somatic, + cnmodels = cnmodels, psiset = psis, pars = pars)
```

And we can look at the inferred allocation of tumor fraction to clones:

```r
> resBoth$psi
[1] 0.45 0.30 0.15 0.05 0.05
```

```r
> simTumor@psi
```
An object of class "WeightVector"
Slot "psi":
[1] 0.5 0.3 0.2

Surprisingly, the results here are similar to the overaggressive results obtained using just the sequencing data rather than the simpler and correct results obtained when using just the copy number data.

In conclusion, CloneSeeker can be applied effectively to cases where one has SNP array data, (processed) sequencing data, or both.