InCaSCN

Introduction

This document provides a brief tutorial on using the InCaSCN package, which implements a constraint dictionary learning problem to recover subclones across several patients with SNP data. The InCaSCN model is designed to identify regions of copy number variation (CNV) in multi-sample SNPs data. In particular, it takes advantage of any similarities shared among samples while maintaining the ability to identify any potential heterogeneity by using parental copy number signals. The model is decribed in details below

Model

We model for minor and major DNA copy number profiles, denoted respectively by $y^1_{i\bullet}$ and $y^2_{i\bullet}$, as

$$y_{ils} \quad = \quad \sum_{k=1}^{p} w_{ik} z_{kls} + \epsilon_{is} \quad for \quad s = 1, 2\,, \quad and \quad l = 1, \dots L\,, \tag{1}$$

where z_{kl1} is the minor copy number for k-th subclone at location l, z_{kl2} is the major copy number for k-th subclone and location l, and w_{ik} is the weight associated to sample i and subclone k . The weights are assumed to be the same for minor and major copy numbers, as these weights correspond to the proportion of each subclone. Therefore, summing (1) for $s = 1$ and $s = 2$, we recover the latent model **??**.%, with $y_{i\bullet} = y^1_{i\bullet} + y^2_{i\bullet}$. To fit model 1, we estimate W, Z^1 and Z^2 by minimizing gaussian errors for $s = 1, 2$:

$$\sum_{s=1}^{2} F(W, Z_s) \quad = \quad \sum_{s=1}^{S} \sum_{i=1}^{n} \sum_{l=1}^{L} \left(y_{ils} - \sum_{k=1}^{p} w_{ik} z_{kls} \right)^2 \tag{2}$$

We constraint each subclone Z_1 and Z_2 to have only few breakpoints, this implies that subclones need to be piecewise constant in order to be interpreted easier (few alterations). Meaning of weights is also important, indeed, we can see them as proportion of subclones (heterogeneity components) which composes tumor samples, then their sum needs to be equal to one for each profile i and all coefficients need to be positive. [**?**]

General model of the minor and major copy number can be written as follows:

$$\text{argmin}_{w_{ik} \geq 0, z_{k\ell}, Z_s} \sum_{s=1}^{S} \left(F(W, Z_s) + \lambda_s \sum_{k=1}^{p} \sum_{l=1}^{m-1} |z_{kl+1s} - z_{kls}| \right)$$
$$\text{s.t.} \quad \sum_{k=1}^{p} w_{ik} = 1, \forall i = 1, \dots, n. \tag{3}$$

where p is a user-defined number of archetype for the dictionary $\{z_{k\ell s}\}$. Note that, the optimal penalty coefficients can be different for the minor and the major copy part. Indeed, the scales of the two dimensions (y_1, y_2) are not the same. In addition, breakpoints could occur more often in one of the minor or the major copy number signal than in the other one.

Using InCaSCN package

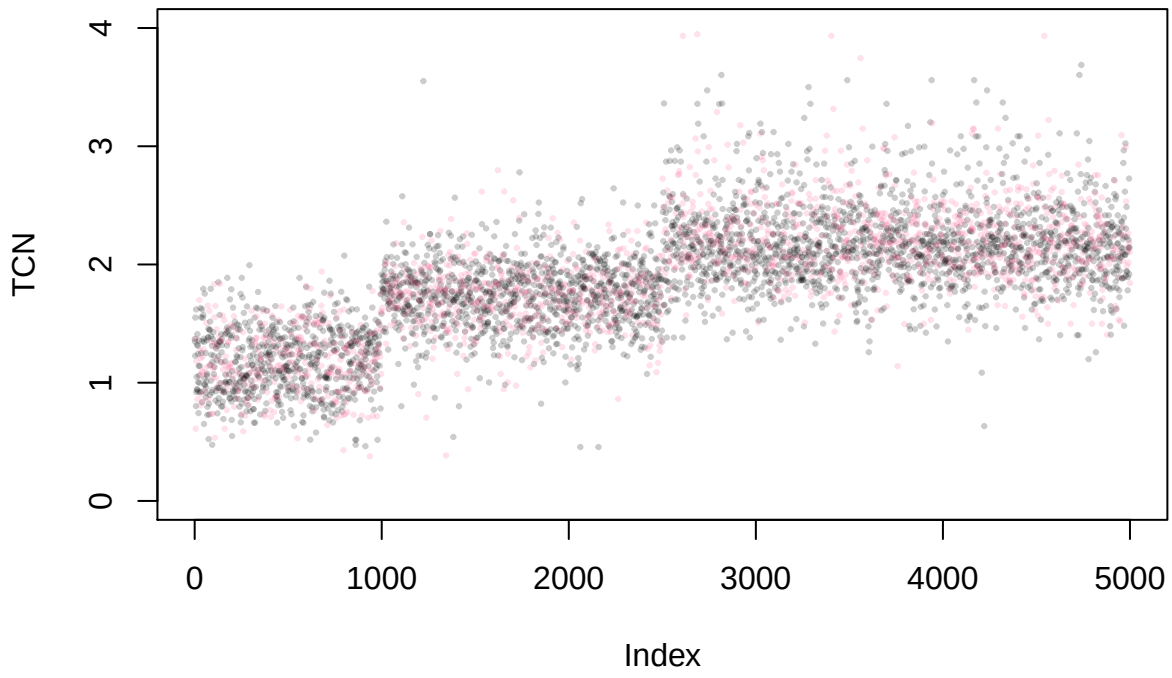
Create Simulations

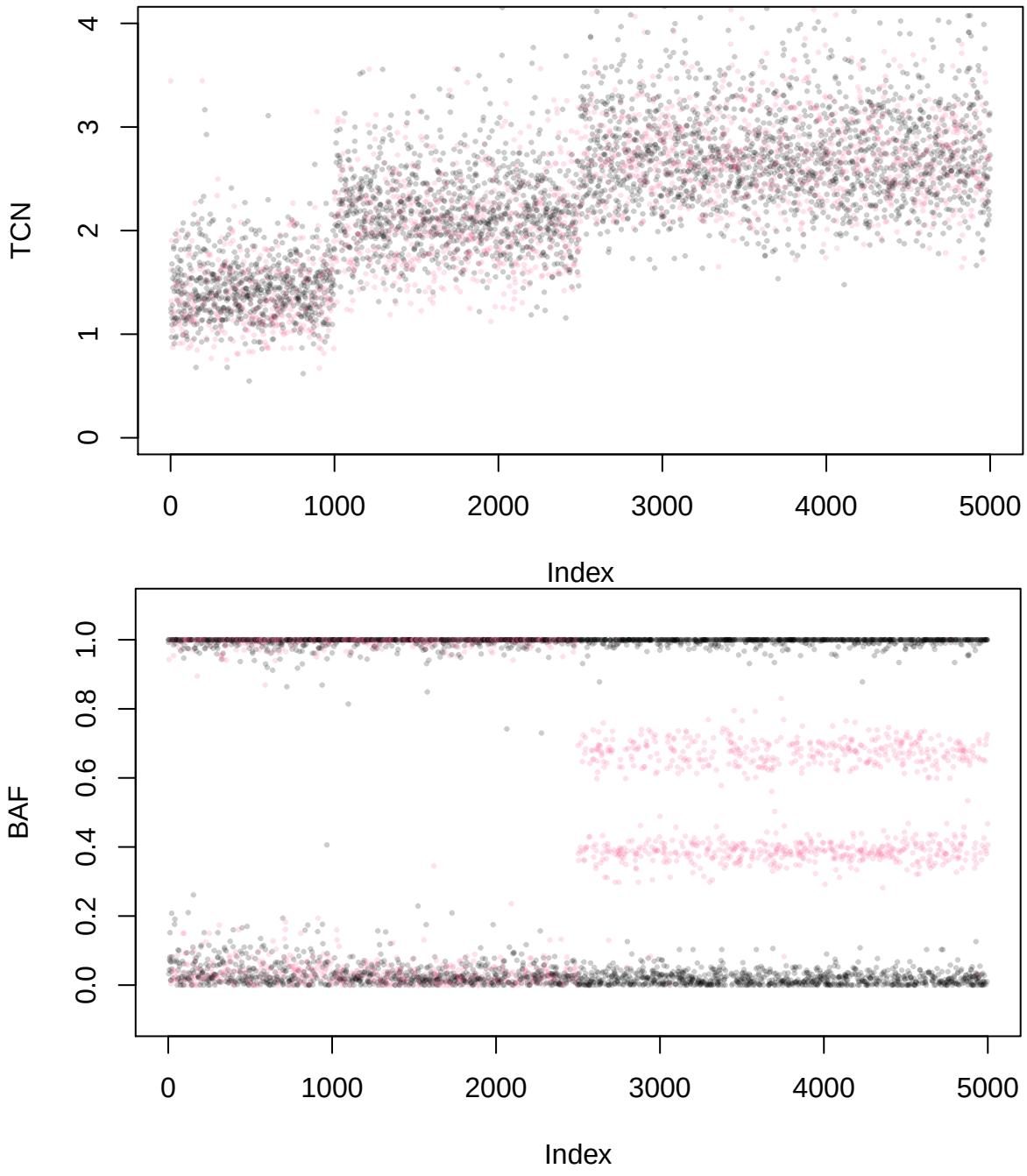
This package permits to create artificial dataset from real data. The first step is to load an annotated dataset (there exists two in this package). Then, after defining characteristics, we can create subclones.

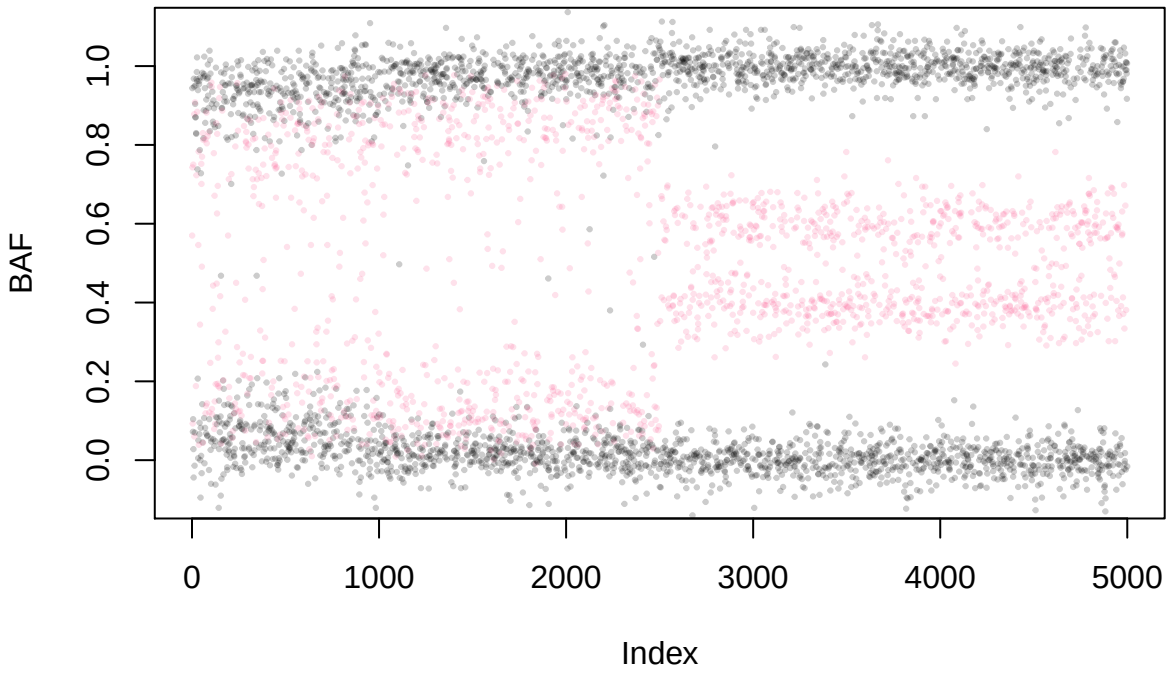
```
dataAnnotTP <- loadCnRegionData(dataSet="GSE11976", tumorFrac=1)
dataAnnotN <- loadCnRegionData(dataSet="GSE11976", tumorFrac=0)
len <- 500*10
nbClones <- 3
bkps <- list(c(100,250)*10, c(150,400)*10,c(150,400)*10)
regions <-list(c("(0,1)", "(0,2)","(1,2)"), c("(1,1)", "(0,1)","(1,1)"), c("(0,2)", "(0,1)","(1,1)"))
datSubClone <- buildSubclones(len, dataAnnotTP, dataAnnotN, nbClones, bkps, regions)
```

Example with the second dataset.

```
dataAnnotTP <- loadCnRegionData(dataSet="GSE13372", tumorFraction=1)
dataAnnotN <- loadCnRegionData(dataSet="GSE13372", tumorFraction=0)
datSubClone2 <- buildSubclones(len, dataAnnotTP, dataAnnotN, nbClones, bkps, regions)
```




TCN
0
1
2
3
4
0
1000
2000
3000
4000
5000
Index
BAF
0.0
0.2
0.4
0.6
0.8
1.0
0
1000
2000
3000
4000
5000
Index



Once subclones are created, it is also easy to generate a matrix W in order to build mixtures.

```
W = getWeightMatrix(70,30, nb.arch = 3, nb.samp = 20)
dat <- apply(W, 1, mixSubclones, subClones=datSubClone, fracN=NULL)
str(dat[[1]])
```

```
## 'data.frame':    5000 obs. of  7 variables:
##  $ c1      : num  NA NA NA NA 0.566 ...
##  $ c2      : num  NA NA NA NA 1.22 ...
##  $ tcn     : num  1.72 1.86 1.6 1.63 1.78 ...
##  $ dh      : num  NA NA NA NA 0.366 ...
##  $ genotype: num  1 1 1 0 0.5 0.5 0 1 0 0.5 ...
##  $ chr     : num  1 1 1 1 1 1 1 1 1 1 ...
##  $ pos     : int  1 2 3 4 5 6 7 8 9 10 ...
```

Note that dat is a list of data frame with the following necessary columns : `c1,c2,tcn,dh,genotype`

Run InCaSCN model

Then it is easy to run InCaSCN model on the data. Let choose the same grid for λ_1 and λ_2 and a grid from 2 to 6 for the number of subclones.

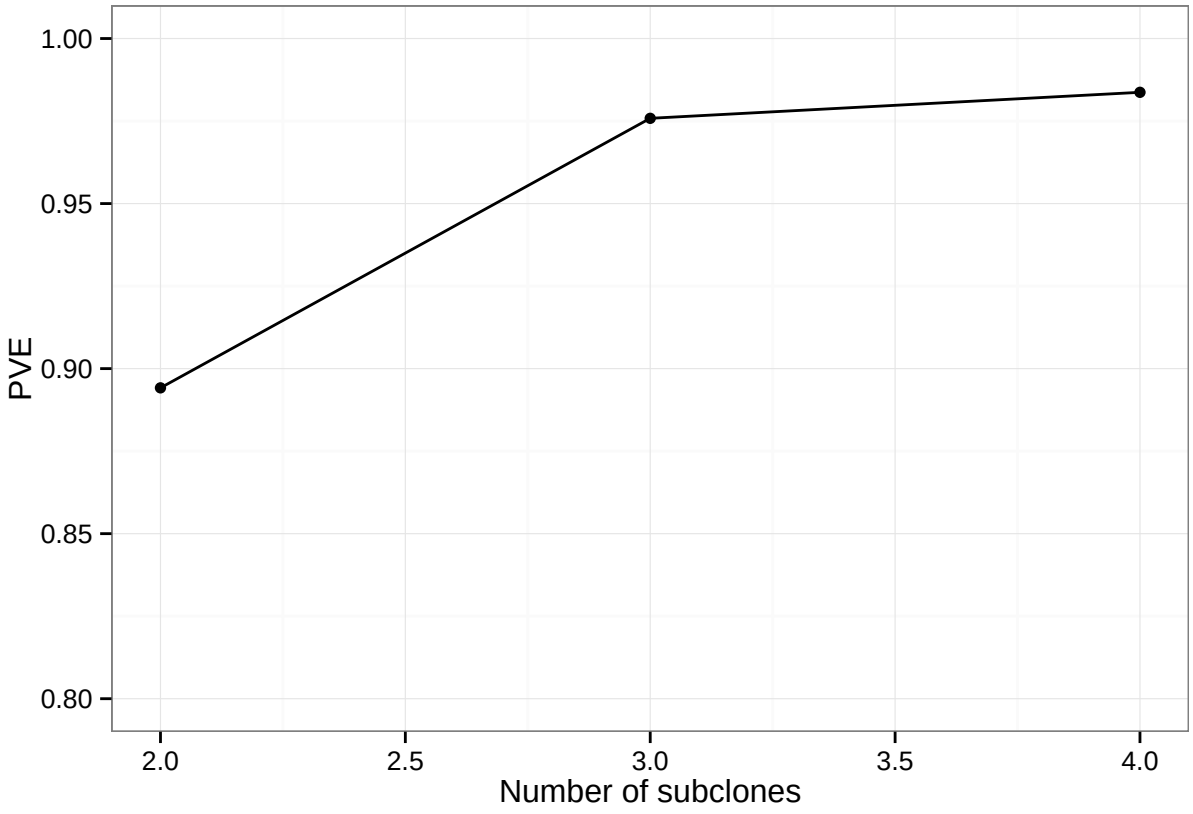
```
lambda1.grid <- lambda2.grid <- c(0.005,0.001)
casRes <- InCaSCN(dat,lambda1.grid, lambda2.grid, nb.arch.grid = 2:6)
```

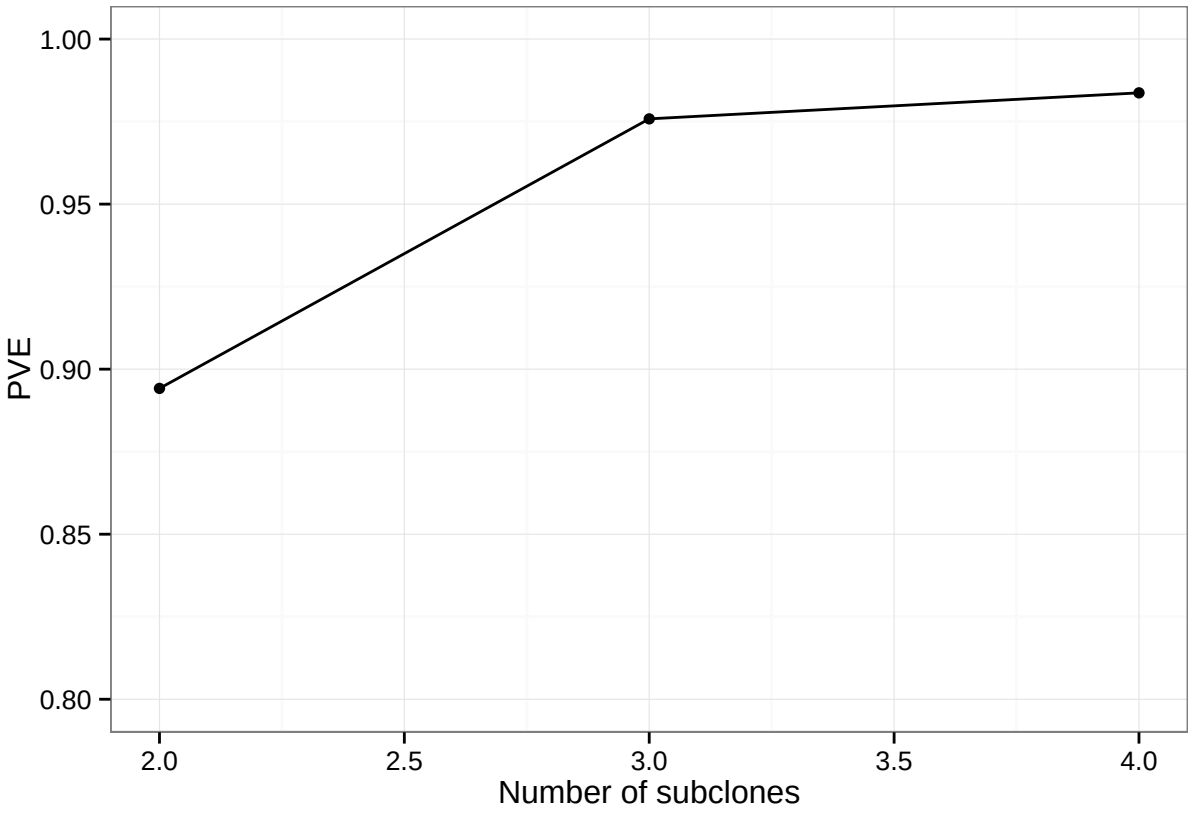
```
## [1] 2
## [1] 3
## [1] 4
```

```
casResTCN <- InCaSCN(dat,lambda1.grid, lambda2.grid, nb.arch.grid = 2:6, stat="TCN")
```

```
## [1] 2
## [1] 3
## [1] 4
```

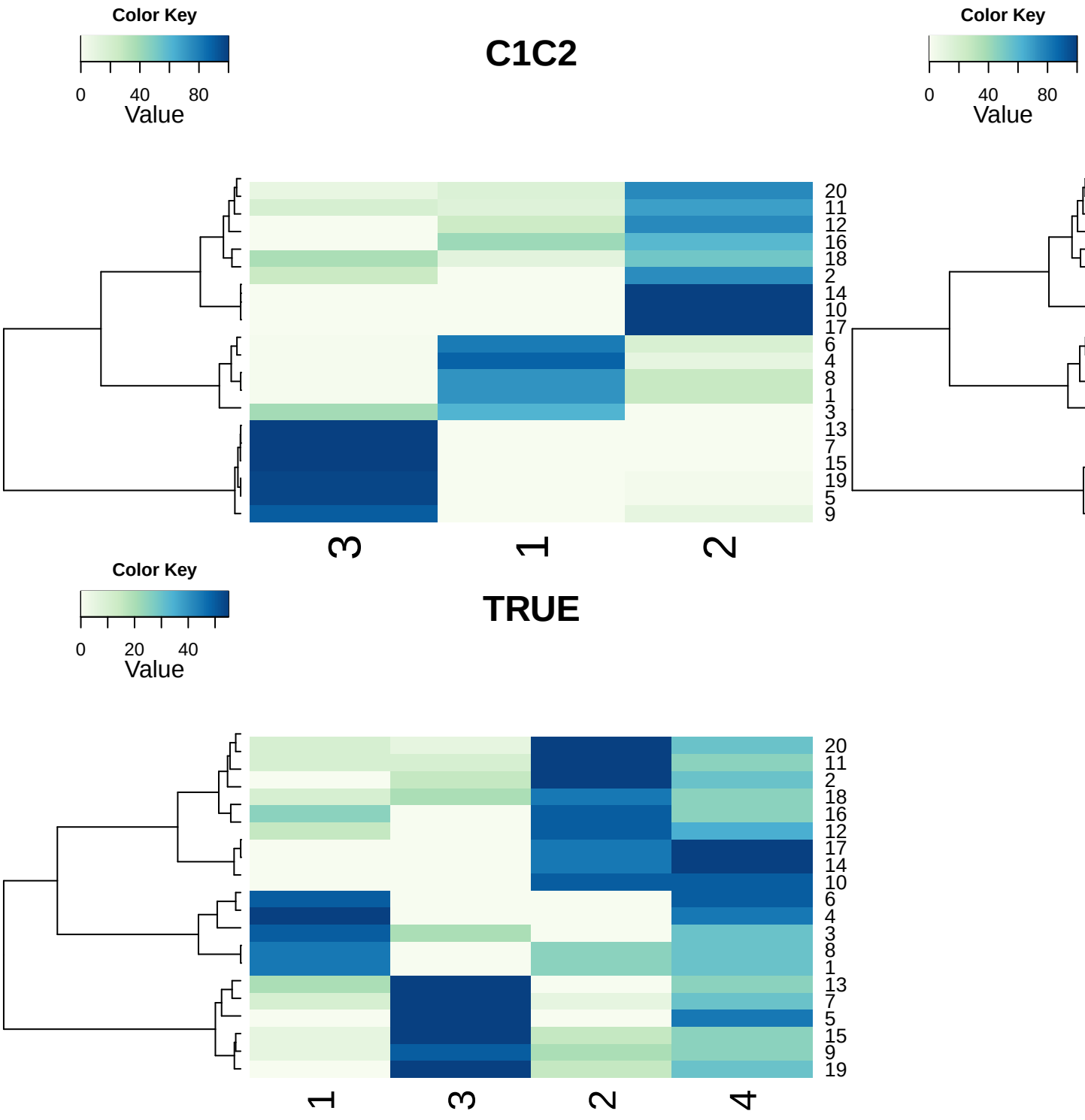
For each p `InCaSCN` keep only the combination (λ_1, λ_2) which minimize the BIC. The next step is to choose the best p (number of subclones). In this example, it seems that the best is $\hat{p} = 4$ (which is the true number of subclones).





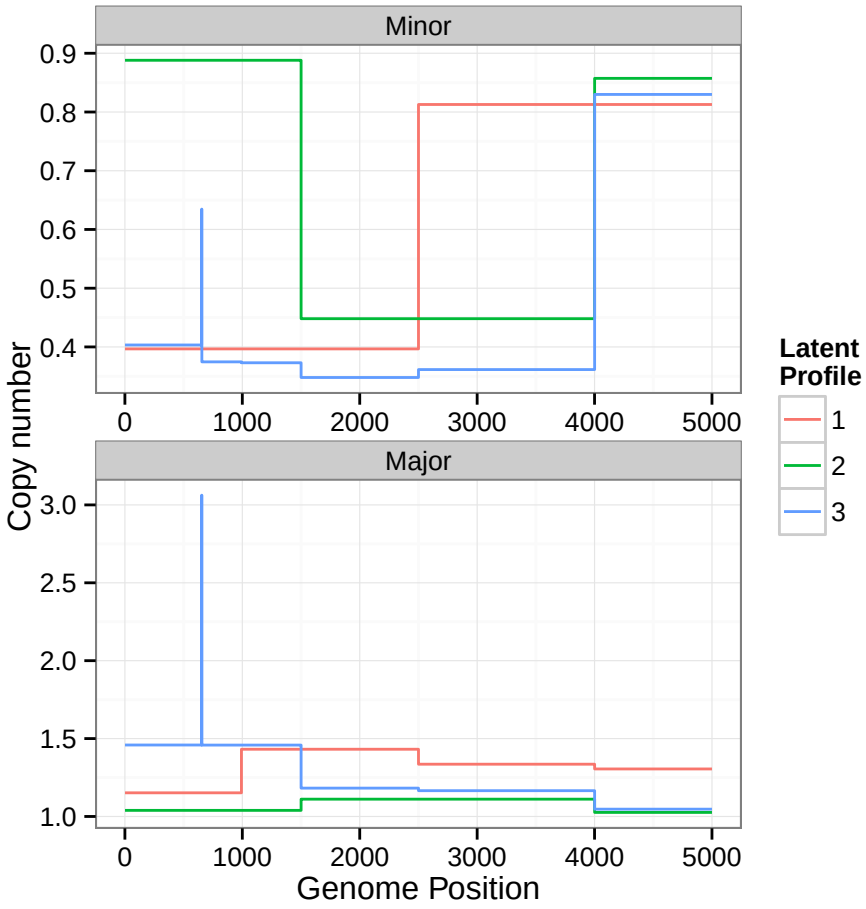
We can compare the true and the estimated matrices of the weights. Even if the computation is not perfect, we can easily recover a classification close to the truth with the inferred weight matrix.

```
## List of 5
##  $ BIC  : num -171
##  $ PVE  : num 0.976
##  $ res  :List of 5
##   ..$ Z    : num [1:7, 1:3] 1.55 1.55 1.55 1.83 1.83 ...
##   ..$ Z1   : num [1:7, 1:3] 0.397 0.397 0.397 0.397 0.397 ...
##   ..$ Z2   : num [1:7, 1:3] 1.15 1.15 1.15 1.43 1.43 ...
##   ..$ W    : num [1:20, 1:3] 0.724 0 0.619 0.885 0 ...
##   ..$ Y.hat:List of 2
##   .. ..$ Y1: num [1:20, 1:7] 0.525 0.762 0.399 0.444 0.417 ...
##   .. ..$ Y2: num [1:20, 1:7] 1.13 1.15 1.27 1.15 1.45 ...
##  $ param:List of 3
##   ..$ nb.arch: int 3
##   ..$ lambda1: num 0.001
##   ..$ lambda2: num 0.001
##  $ bkp  :List of 1
##   ..$ : num [1:6] 654 656 992 1500 2500 ...
```



If we look at the subclones in the dimension of parental copy numbers, we can recover the simulated alterations.

```
z1 <- rbind(casRes[[idxBestC1C2]]$res$Z1,casRes[[idxBestC1C2]]$res$Z1[length(bkp),])
z2 <- rbind(casRes[[idxBestC1C2]]$res$Z2,casRes[[idxBestC1C2]]$res$Z2[length(bkp),])
dfZ <- data.frame(Cn=c(as.numeric(z1),as.numeric(z2)),stat=factor(rep(c("Minor", "Major"), each = prod(
gC <- ggplot(dfZ, aes(y=Cn, x=bkps))+geom_step(aes(col=latent), direction="hv")+ facet_wrap(~stat, ncol=
gC
```


Minor
Major
Copy number
Genome Position
0.9
0.8
0.7
0.6
0.5
0.4
3.0
2.5
2.0
1.5
1.0
0
1000
2000
3000
4000
5000
Latent Profile
1
2
3