Contents

1 Introduction

Single cell RNA-seq data sets from pooled CrispR screens provide the possibility to analyze heterogeneous cell populations. We extended the original Nested Effects Models (NEM) to Mixture Nested Effects Models (M&NEM) to simultaneously identify several causal signaling graphs and corresponding subpopulations of cells. The final result will be a soft clustering of the perturbed cells and a causal signaling graph, which describes the interactions of the perturbed signaling genes (S-genes) for each cluster of cells and the sub-topology for the observed genes (E-genes).

The M&NEM algorithm uses an expectation maximization (EM) algorithm to infer an optimum for k components. In the E-step the expectation of the hidden data (assignment of a cell to a component aka responsibilities) is calculated. Based on the responsibilities M&NEM weights the data for each component and the standard NEM approach is used to optimize the causal network for each weighted data set (M-step).

2 Installation and loading

Install the package with the bioconductor manager package.

```
if (!requireNamespace("BiocManager", quietly = TRUE)) install.packages("BiocManager")
BiocManager::install("mnem")
```

Load the package with the library function.

```
library(mnem)
```

3 Small example

The input data is an m times n matrix of log odds (e.g. for fold changes as in the R package 'limma', Ritchie *et al.*, 2015). The *i*th row of the *j*th column are the log odds for feature (E-gene) i in cell j. As in the case of the original NEM, the data consists of a multitude of E-genes. However, where for the original NEM only one column complies to each perturbed signaling gene (S-gene), M&NEM is designed for single cell data and therefore can handle and draws its power from multiple cells for each perturbed S-gene.

3.1 Data simulation

The M&NEM package includes a functions for the simulation of typical single cell log odds. We use this function to create data for three S-genes. Several E-genes for each S-gene including a small number of uninformative E-genes. Additionally we generate the data from a mixture of three components and sample a population of cells with different numbers of cells belonging to each component (approx. mixture weights: mw). The function simulates discrete data (1 for effect and 0 for no effect). We transform the discrete data to log odds by adding Gaussian noise with mean 1 to all 1s and with mean -1 to all 0s. Hint: If you use the 'mw' parameter, the 'Nems' parameter is not necessary.

Figure 1 shows a heatmap of our generated data. Since we used only mild noise, effects and no effects are still clearly distinguished into shades of blue and red. We can identify the uninformative E-gene (no rownames) by its random/unique patterns. There is a clear clustering noticeable for the knock-downs hinting at differential causal regulation within the cell population. The E-gene numbers denote the attachment of the E-genes in regards to the first component. For the other components attachments differ and do not agree with the E-gene name anymore.

```
seed <- 27
Sgenes <- 3
Egenes <- 10
nCells <- 100
uninform <- 1
mw <- c(0.4, 0.3, 0.3)
Nems <- 3
noise <- 0.5
set.seed(seed)
simmini <- simData(Sgenes = Sgenes, Egenes = Egenes, Nems = Nems, mw = mw, nCells = nCells,
    uninform = uninform)
data <- simmini$data
ones <- which(data == 1)
zeros <- which(data == 0)
data[ones] <- rnorm(length(ones), 1, noise)
data[zeros] <- rnorm(length(zeros), -1, noise)
epiNEM::HeatmapOP(data, col = "RdBu", cexRow = 0.75, cexCol = 0.75, bordercol = "transparent",
    xrot = 0, dendrogram = "both")
```

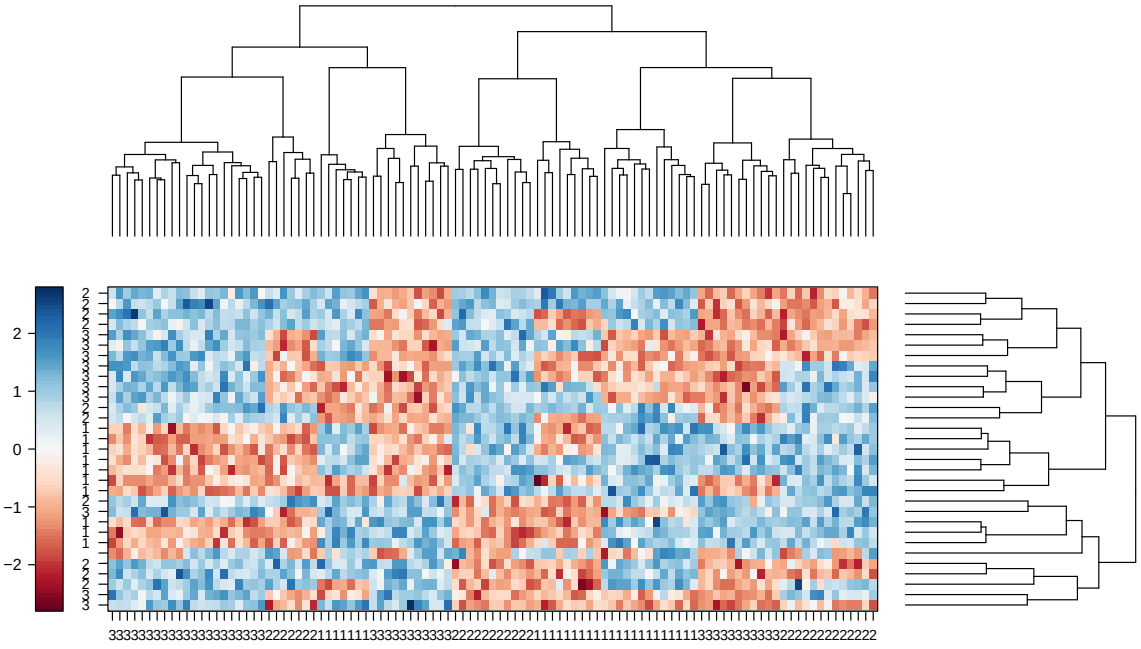

Figure 1: **Heatmap of the simulated log odds**
Effects are blue and no effects are red. Rows denote the observed E-genes and columns the S-genes. Each S-gene has been perturbed in many cells. The E-genes are annotated as how they are attached in the ground truth. E.g. E-genes named '1' are attached to S-gene '1' in the ground truth.

```
plot(simmini, data)
```

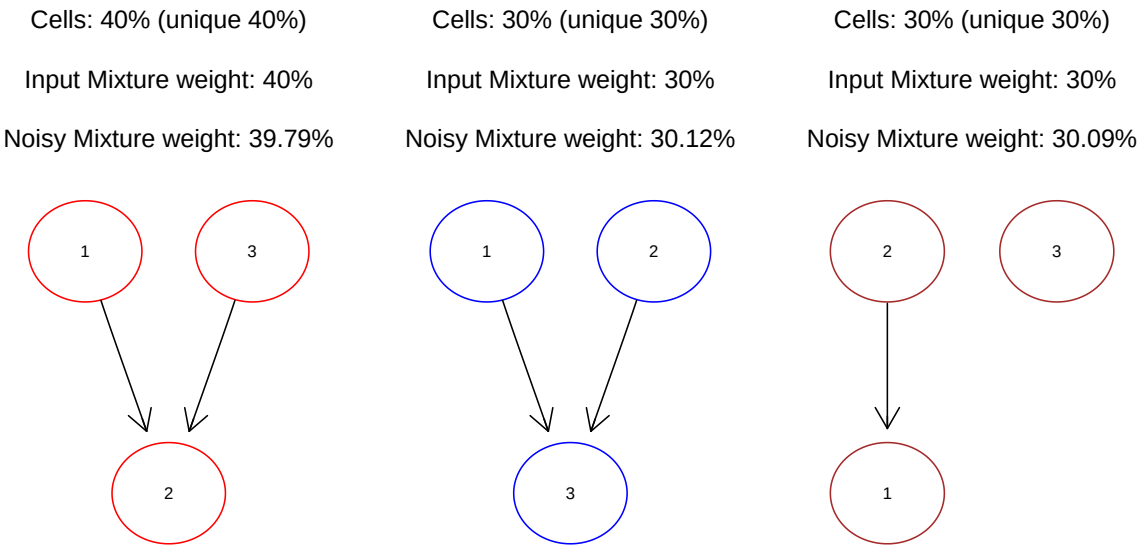

Figure 2: Ground truth causal networks of our simulated mixture of cells

Figure 2 shows three different causal networks for the population of simulated cells. The chosen mixture weights almost agree with the noisy mixture weights due to only little noise.

3.2 Network inference w.r.t. complexity

We forget the solution stored in 'simmini$Nem' and use the 'mnem' function to infer the network. We choose our a greedy search for the M-step and 3 independent starts for the EM algorithm to keep this vignette short. However, even for only three S-genes more starts are generally recommended.

Since the number of components k is a priori not known, we perform optimization for $k = 1, 2, 3, 4$ and use our penalized log likelihood to choose the best k. Figure 3 shows the best mixture of our causal networks with $k = 3$. We have found the same causal networks we used to generate the data.

We can speed up the computation with the 'parallel $=$ n' parameter with n threads.

```
starts <- 5
bestk <- mnemk(data, ks = 1:5, search = "greedy", starts = starts, parallel = NULL)
plot(bestk$best)
```

```
print(fitacc(bestk$best, simmini, strict = TRUE))
## [1] 1
```

```
par(mfrow = c(1, 2))
plot(bestk$lls, col = "blue", main = "raw log likelihood", type = "b", xlab = "number of components",
    ylab = "score")
plot(bestk$ics, col = "red", main = "penalized log likelihood", type = "b", xlab = "number of components",
    ylab = "score")
```

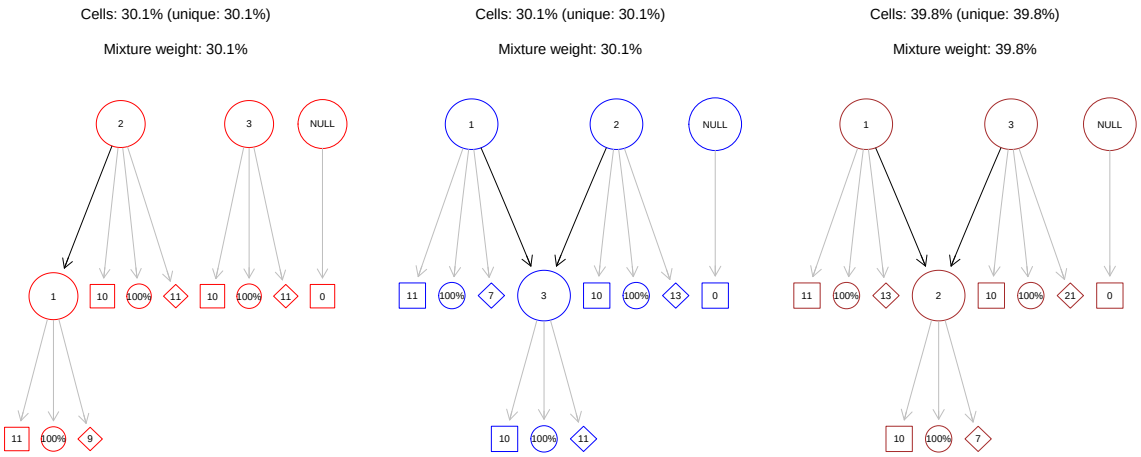

Figure 3: Mixture NEM result for the simulated data
On top we show the cell assignment percentages for hard clustering and the mixture weights. The large circular vertices depict the S-genes, the small ones the responsibility for the best fitting cell, the diamonds the number of assigned cells and the boxes the number of assigned E-genes.

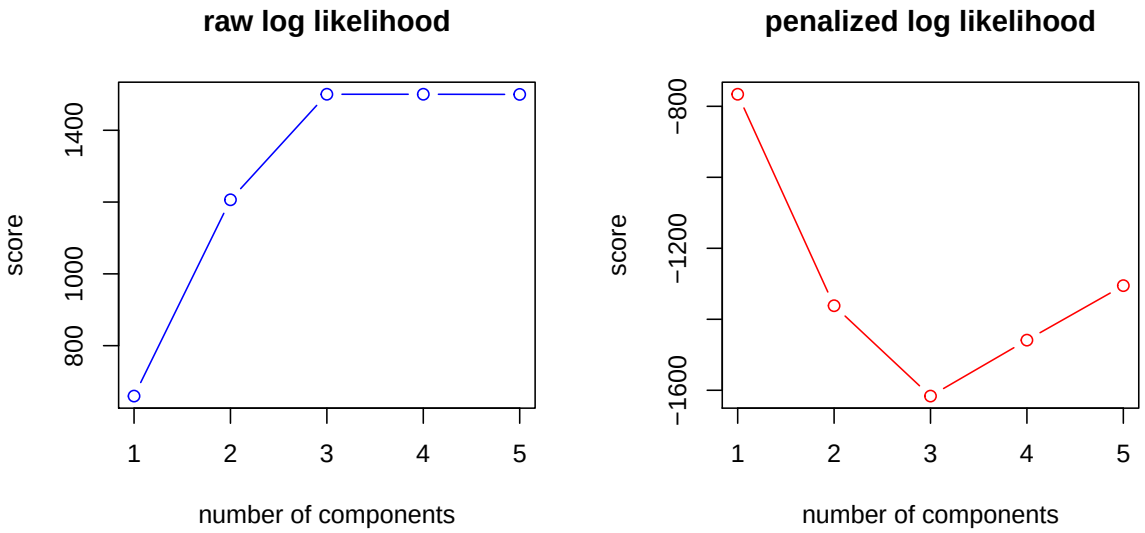

Figure 4: Model selection
Raw log likelihood of the models with different components (left, blue) and penalized log likelihood (right, red).

We compare the actual log likelihood shown in figure 4 (left, blue) to the penalized log likelihood (right, red). Notice that the raw log likelihood of $k = 4, 5$ is the same as for the $k = 3$ component model. However, if we penalize complexity the three component model gives us the best score (minimum). The penalized like the raw log likelihood is not guaranteed to have only one global optimum, but can have several local ones. Hence, in practice when there are little time constraints larger $k > 4$ should also be tested.

We can also look at the convergence of the expectation maximization algorithm. Since we started the EM 5 times, we have as many convergences.

```
par(mfrow = c(2, 2))
plotConvergence(bestk$best, cex.main = 0.7, cex.lab = 0.7)
```

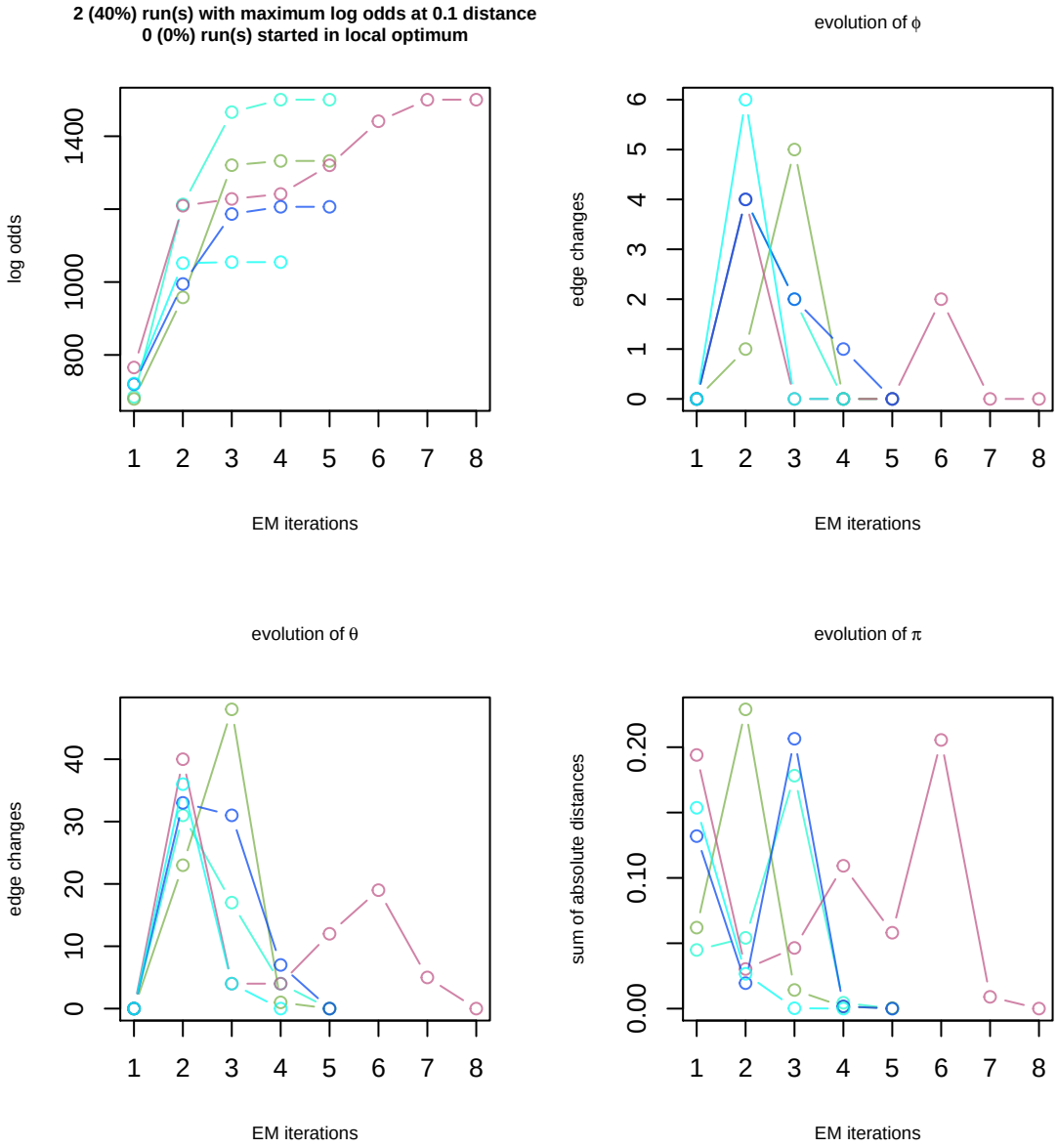

Figure 5: EM Convergence
Convergence plot for the different starts of the EM algorithm.

3.3 Cell responsibilities

Figure 5 shows the actual responsibilities of the best scoring model. They denote how well each cell fits to each component. e.g. when a cell has a responsibility of 100% for one component, then the responsibility for the other two is 0%.

```
postprobs <- getAffinity(bestk$best$probs, mw = bestk$best$mw)
hist(postprobs, xlab = "responsibilities", main = "")
```

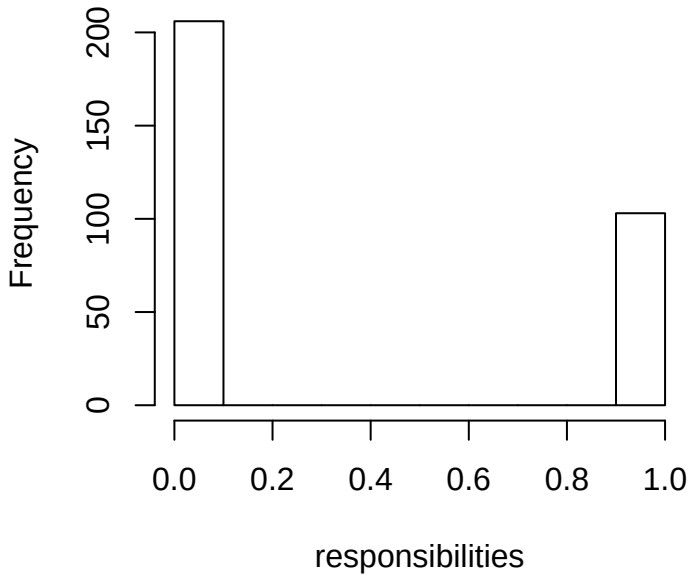

Figure 6: Responsibility distributions
Histograms of the responsibilities for the best fitting model.

3.4 Discrete data

For certain data sets calculating binary data (1 for effect and 0 for no effect) might be more straightforward (e.g. Markowetz *et al.*, 2005) than log odds. Thus mixture NEM also supports discrete data as input for the EM algorithm. To do inference on discrete data, we just have to set the method parameter to "disc". However, for likelihood calculations we need to set false positive and false negative rates. I.e. how many effects we assume are coming from noise and how many real effects have been turned to zeros. These are typically not known, but sometimes can be estimated. Although, Markowetz *et al.*, 2007 have shown, that the causal network is stable across a variety of different rates and only breaks down for very unreasonable (i.e. high) values.

We choose false positive and false negative rates of 10% each and apply them to our simulated data set.

```
set.seed(seed)
datadisc <- simmini$data
fp <- sample(which(datadisc == 0), floor(0.1 * sum(datadisc == 0)))
fn <- sample(which(datadisc == 1), floor(0.1 * sum(datadisc == 1)))
datadisc[c(fp, fn)] <- 1 - datadisc[c(fp, fn)]
epiNEM::HeatmapOP(datadisc, col = "RdBu", cexRow = 0.75, cexCol = 0.75, bordercol = "transparent",
    xrot = 0, dendrogram = "both")
```

Similar to the log odds matrix, the structure of the simulated mixture of causal networks is still recognizable even with the addition of our (mild) noise. As a shortcut, we directly set the number of components to 3.

```
result_disc <- mnem(datadisc, k = length(mw), search = "greedy", starts = starts,
    method = "disc", fpfn = c(0.1, 0.1))
plot(result_disc)
```

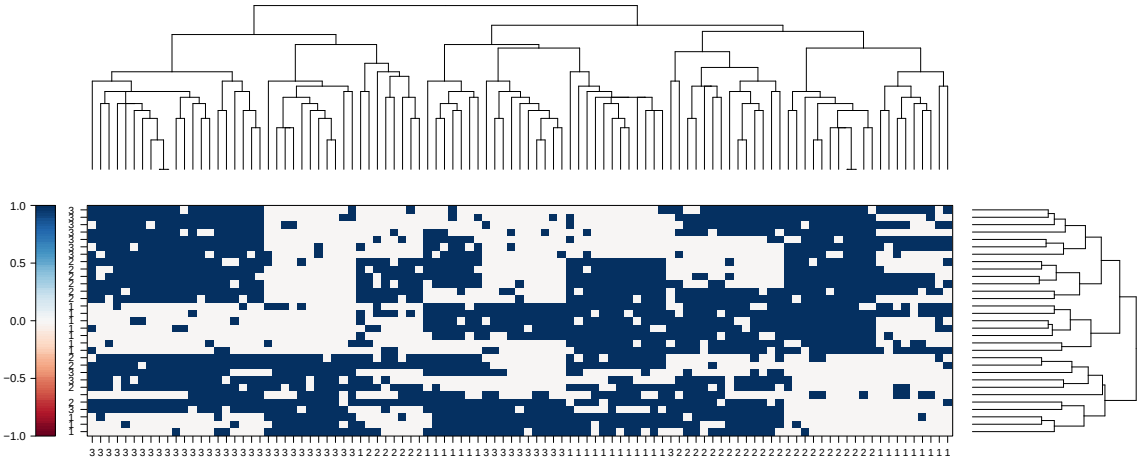

Figure 7: **Heatmap of the simulated binary data**
Effects are blue and no effects are white. Rows denote the observed E-genes and columns the S-genes. Each S-gene has been perturbed in many cells. The E-genes are annotated as how they are attached in the ground truth. E.g. E-genes named '1' are attached to S-gene '1' in the ground truth.

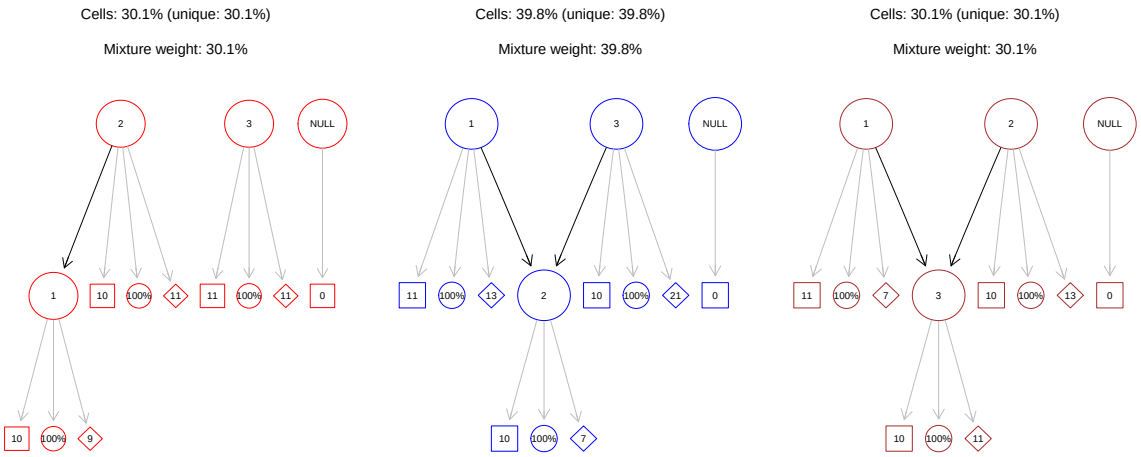

Figure 8: **Binary data**
Mixture NEM result for a binary data set.

3.5 Multiple perturbations

Mixtures are in general very complex. Currently M&NEM only supports the standard NEM approach and no other extensions, which would increase complexity even further. However, M&NEM supports multiple perturbation per sample. The information of multiple perturbations is not used to infer more complex signaling structures like B-NEM (Pirkl *et al.*, 2016). M&NEM just uses the information to increase accuracy of the standard NEM approach. E.g. let's assume during a knock-down of gene A genes E1 to E10 react and during a knock-down of gene B genes E11 to E20 react. A sample with a perturbation of genes A and B will show a reaction of genes E1 to E20 (assuming perfect data), even though A and B are not linearly connected.

Again, we simulate data this time including 20% double and 10% triple knock-downs. The inference is done the same way, except we specifically tell the algorithm, that we have data with multiple perturbations with 'multi = TRUE'. Take a look at how the samples are named. The "_" character separates different S-genes in the data column names.

```
set.seed(seed)
simmini2 <- simData(Sgenes = Sgenes, Egenes = Egenes, Nems = Nems, mw = mw, nCells = nCells,
    uninform = uninform, multi = c(0.2, 0.1))
data <- simmini2$data
ones <- which(data == 1)
zeros <- which(data == 0)
data[ones] <- rnorm(length(ones), 1, noise)
data[zeros] <- rnorm(length(zeros), -1, noise)
epiNEM::HeatmapOP(data, col = "RdBu", cexRow = 0.75, cexCol = 0.75, bordercol = "transparent",
    dendrogram = "both")
```

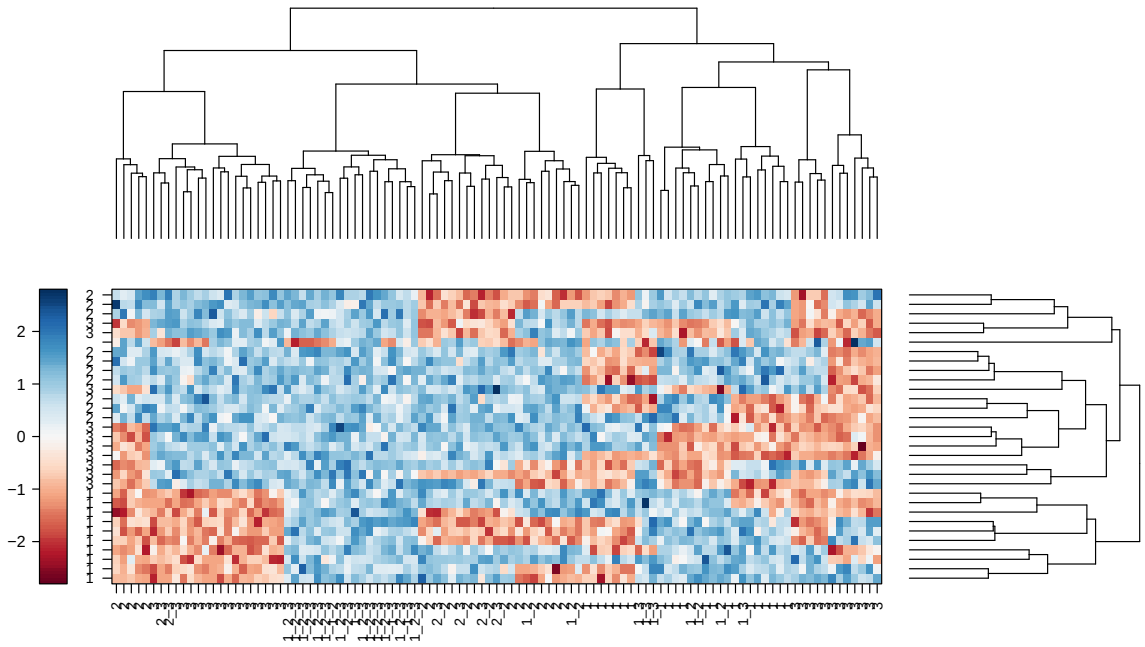

Figure 9: Multiple perturbations
Simulated data including samples in which more than one S-gene has been perturbed.

```
result <- mnem(data, k = length(mw), search = "greedy", multi = TRUE, starts = starts)
plot(result)
```

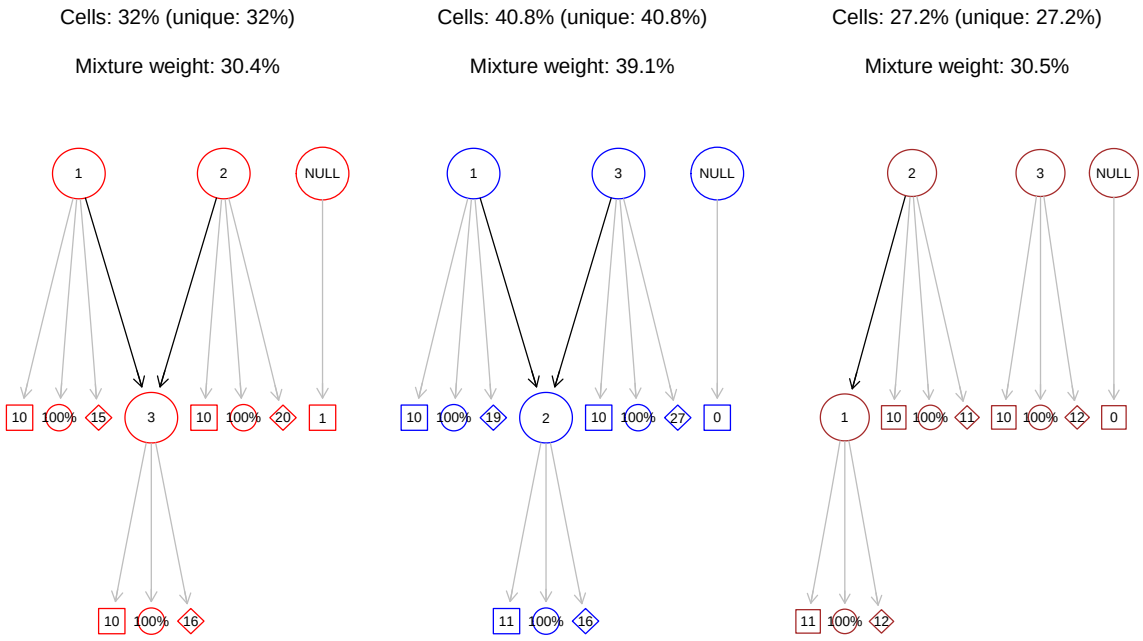

Figure 10: **Multi sample result**
The result of the mixture NEM inference on data including samples with multiple perturbations.

4 Application to pooled CRISPR screens

We apply M&NEM to three different data sets from two different pooled CRISPR screens, Crop-seq (Datlinger *et al.*, 2017) and Perturb-seq (Dixit *et al.*, 2016). Those screens consists of single cell RNA-seq data. Each cell has been perturbed by a knock-down of a gene (S-gene) and each cell is observed by its gene expression profile (E-genes).

We optimized the mixture for components $k = 1, 2, 3, 4, 5$. Hence, we use our penalized likelihood to find the optimal number of components without over-fitting.

The data object 'app' consists of three lists containing the results for Crop-seq, Perturb-seq (transcription factors) and Perturb-seq (cell cycle regulators). Each list contains the results for $k = 1, 2, 3, 4, 5$ (number of components).

The object 'app' was generated by the following code (do not run before you've read the function's help page):

```
app <- createApp()
```

4.1 Cell cycle regulators (Perturb-seq)

As an example we show the results for the cell cycle regulators of the Perturb-seq data set. We show the raw log likelihood together with the penalized likelihood. We choose the optimal k at the minimum of the penalized log likelihood.

```
data(app)
res2 <- app
```

```
maxk <- 5
j <- 2
res <- res2[[j]]
for (i in 2:5) {
    res[[i]]$data <- res[[1]]$data
}
bics <- rep(0, maxk)
ll <- rep(0, maxk)
for (i in seq_len(maxk)) {
    bics[i] <- getIC(res[[i]])
    ll[i] <- max(res[[i]]$ll)
}
ll2 <- ll
ll <- (ll/(max(ll) - min(ll))) * (max(bics) - min(bics))
ll <- ll - min(ll) + min(bics)
ll3 <- seq(min(bics), max(bics[!is.infinite(bics)]), length.out = 5)
par(mar = c(5, 5, 2, 5))
plot(bics, type = "b", ylab = "", col = "red", xlab = "", yaxt = "n", ylim = c(min(min(bics,
    ll)), max(max(bics, ll))), xaxt = "n")
lines(ll, type = "b", col = "blue")
axis(4, ll3, round(seq(min(ll2), max(ll2), length.out = 5)), cex.axis = 0.5)
axis(2, ll3, round(ll3), cex.axis = 0.5)
axis(1, 1:maxk, 1:maxk)
mtext("penalized", side = 2, line = 3, cex = 1.2)
mtext("raw", side = 4, line = 3, cex = 1.2)
```

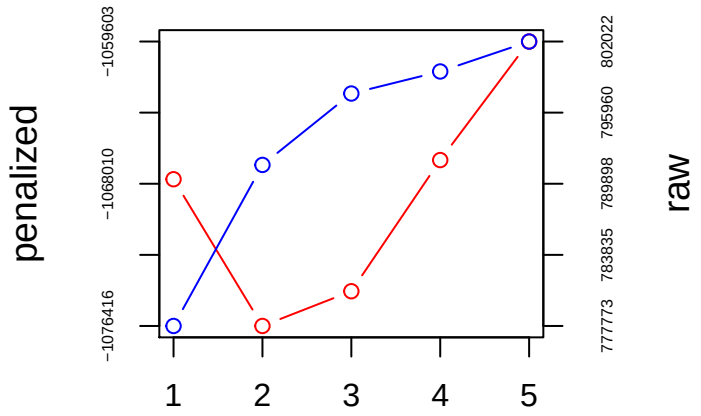

Figure 11: **Penalized and raw log likelihood ratios**
Blue denotes the raw log likelihood and red the negative penalized for complexity.

Figure 10 shows that the optimal mixture is $k = 2$ according to our penalized log likelihood. Even $k = 3$ beats just a single network.

Each cell has a certain probability (responsibility) to have been generated by a components. The histograms show the distribution of the responsibilities for all cells.

```
j <- 2
res <- res2[[j]]
for (i in 2:5) {
```

```
    res[[i]]$data <- res[[1]]$data
}
bics <- rep(0, maxk)
ll <- rep(0, maxk)
for (i in seq_len(maxk)) {
    bics[i] <- getIC(res[[i]])
    ll[i] <- max(res[[i]]$ll)
}
ll2 <- ll
ll <- (ll/(max(ll) - min(ll))) * (max(bics) - min(bics))
ll <- ll - min(ll) + min(bics)
ll3 <- seq(min(bics), max(bics[!is.infinite(bics)]), length.out = 5)
i <- which.min(bics)
gamma <- getAffinity(res[[i]]$probs, mw = res[[i]]$mw)
par(mar = c(5, 5, 2, 5))
hist(gamma, main = "Histogram of responsibilities", xlab = "responsibilities")
```

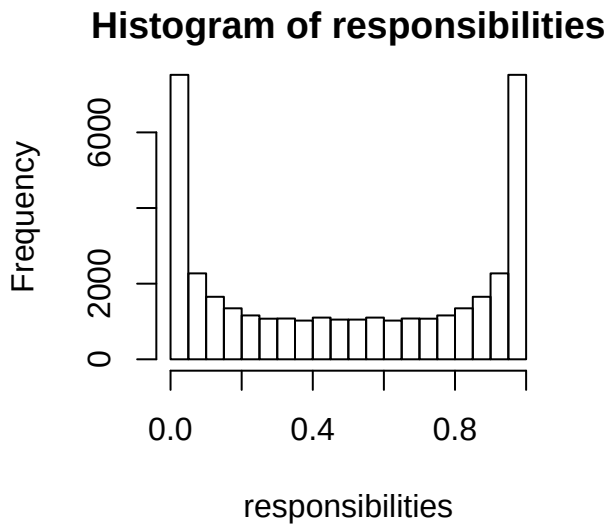

Figure 12: Histograms of the responsibilities
The responsibilities for the highest scoring mixture according to the penalized log likelihood.

Figure 11 shows the distribution of responsibilities. There are many cells with a responsibility of 100%, but many cells have one around 50% and are therefore ambiguous to which component they belong.

We show the highest and second highest scoring mixtures. On top is the percentage of cells assigned to each network by hard clustering and the mixture weights. The connected large circles depict the causal network of S-genes. The small circles show the responsibility of the best fitting cell. The boxes show the number E-genes assigned to each S-gene and the diamonds show the numbers of cells assigned to each S-gene. The NULL node is assigned E-genes, which fit best to an S-gene (NULL) with no effects at all.

The full methodology this package is based on, a simulation study and the rest of our application are shown in Pirkl & Beerenwinkel (2018).

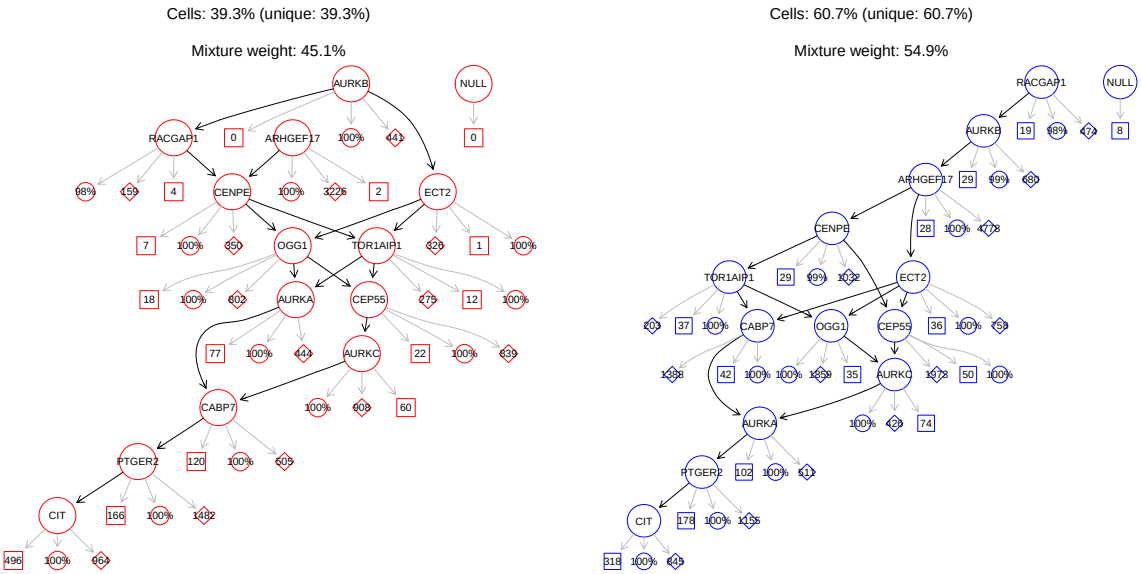

Figure 13: **Highest scoring mixture for the Perturb-Seq dataset of cell cycle regulators**
On top we show the cell assignments for hard clustering and the mixture weights. The large circular vertices depict the S-genes, the small ones the responsibility for the best fitting cell, the diamonds the number of assigned cells and the boxes the number of assigned E-genes.

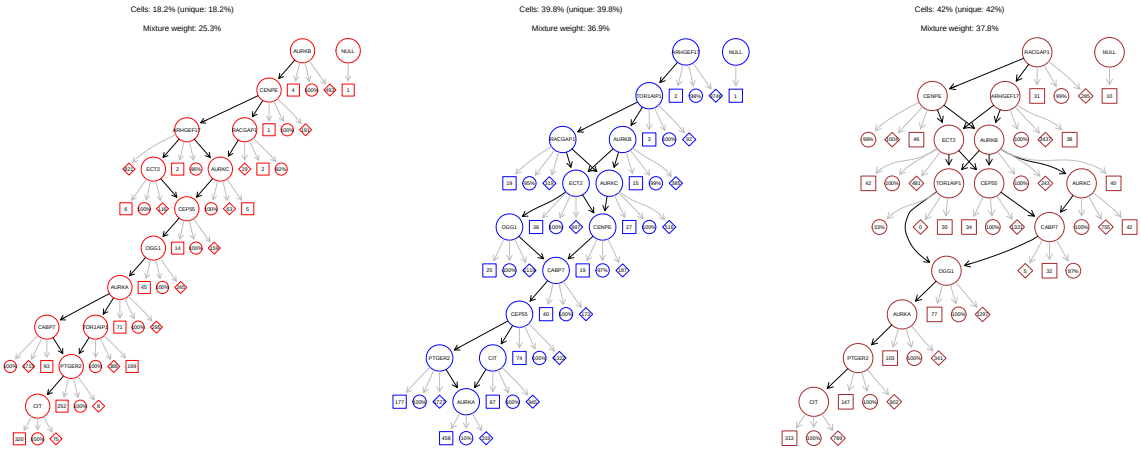

Figure 14: **Second highest scoring mixture for the Perturb-Seq dataset of cell cycle regulators**
On top we show the cell assignments for hard clustering and the mixture weights. The large circular vertices depict the S-genes, the small ones the responsibility for the best fitting cell, the diamonds the number of assigned cells and the boxes the number of assigned E-genes.

5 Session information

```
sessionInfo()
## R version 3.6.0 (2019-04-26)
## Platform: x86_64-apple-darwin15.6.0 (64-bit)
## Running under: OS X El Capitan 10.11.6
##
## Matrix products: default
```

```
## BLAS:   /Library/Frameworks/R.framework/Versions/3.6/Resources/lib/libRblas.0.dylib
## LAPACK: /Library/Frameworks/R.framework/Versions/3.6/Resources/lib/libRlapack.dylib
##
## locale:
## [1] C/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base
##
## other attached packages:
## [1] mnem_1.0.0       BiocStyle_2.12.0
##
## loaded via a namespace (and not attached):
##   [1] amap_0.8-16               Rtsne_0.15
##   [3] colorspace_1.4-1          RcppEigen_0.3.3.5.0
##   [5] class_7.3-15              modeltools_0.2-22
##   [7] mclust_5.4.3              Linnorm_2.8.0
##   [9] corpcor_1.6.9             RcppArmadillo_0.9.400.2.0
##  [11] ggdendro_0.1-20           clue_0.3-57
##  [13] flexmix_2.3-15            mvtnorm_1.0-10
##  [15] splines_3.6.0             robustbase_0.93-4
##  [17] knitr_1.22                jsonlite_1.6
##  [19] apcluster_1.4.7           cluster_2.0.9
##  [21] kernlab_0.9-27            sfsmisc_1.1-3
##  [23] snowfall_1.84-6.1         graph_1.62.0
##  [25] BiocManager_1.30.4        compiler_3.6.0
##  [27] assertthat_0.2.1          Matrix_1.2-17
##  [29] lazyeval_0.2.2            limma_3.40.0
##  [31] formatR_1.6               htmltools_0.3.6
##  [33] tools_3.6.0               igraph_1.2.4.1
##  [35] BoolNet_2.1.5             gtable_0.3.0
##  [37] glue_1.3.1                epiNEM_1.8.0
##  [39] dplyr_0.8.0.1             gmodels_2.18.1
##  [41] V8_2.2                    Rcpp_1.0.1
##  [43] trimcluster_0.1-2.1       gdata_2.18.0
##  [45] nlme_3.1-139              fpc_2.1-11.2
##  [47] xfun_0.6                  fastcluster_1.1.25
##  [49] stringr_1.4.0             ggm_2.3
##  [51] gtools_3.8.1              statmod_1.4.30
##  [53] XML_3.98-1.19             DEoptimR_1.0-8
##  [55] MASS_7.3-51.4             zoo_1.8-5
##  [57] scales_1.0.0              minet_3.42.0
##  [59] dagitty_0.2-2             parallel_3.6.0
##  [61] RBGL_1.60.0               RColorBrewer_1.1-2
##  [63] yaml_2.2.0                curl_3.3
##  [65] ggplot2_3.1.1             bdsmatrix_1.3-3
##  [67] naturalsort_0.1.3         fastICA_1.2-1
##  [69] latticeExtra_0.6-28       stringi_1.4.3
##  [71] nem_2.58.0                plotrix_3.7-5
##  [73] e1071_1.7-1               permute_0.9-5
##  [75] BiocGenerics_0.30.0       boot_1.3-22
```

```
##  [77] matrixStats_0.54.0        pcalg_2.6-2
##  [79] rlang_0.3.4               pkgconfig_2.0.2
##  [81] prabclus_2.2-7            evaluate_0.13
##  [83] lattice_0.20-38           purrr_0.3.2
##  [85] tidyselect_0.2.5          plyr_1.8.4
##  [87] magrittr_1.5              bookdown_0.9
##  [89] R6_2.4.0                  pillar_1.3.1
##  [91] mgcv_1.8-28               abind_1.4-5
##  [93] nnet_7.3-12               tibble_2.1.1
##  [95] tsne_0.1-3                crayon_1.3.4
##  [97] ellipse_0.4.1             rmarkdown_1.12
##  [99] grid_3.6.0                flexclust_1.4-0
## [101] data.table_1.12.2         vegan_2.5-4
## [103] Rgraphviz_2.28.0          digest_0.6.18
## [105] diptest_0.75-7            stats4_3.6.0
## [107] munsell_0.5.0
```

6 Discrete data model

Log odds are a reasonable way for the data normalization, if the perturbation experiments are accompanied by a single control (Pirkl & Beerenwinkel 2018). If the experimental setup includes positive control (e.g. stimulated cells) and negative control (unstimulated), discretizing the data to $0, 1$ is also feasible (Markowetz *et al.*, 2007).

Let Φ be the transitively closed adjacency matrix of the causal network connecting the S-genes and Θ the E-gene attachment with $\theta_{ij} = 1$, if E-gene j is attached to S-gene i. Let ρ be the perturbation map with $\rho_{ij} = 1$, if cell j has been perturbed by a knock-down of S-gene i.

In the case of discrete data we need to account for false positives and negatives with rates α and β. In this case we calculate the responsibilities by

$$\gamma_{ik} = \frac{\pi_k \prod_{j=1}^{m} P\left(d_{ji} = a \mid f_{k,ij} = b\right)}{\sum_s \pi_s \prod_{j=1}^{m} P\left(d_{ji} = a \mid f_{s,ij} = b\right)}$$

with the discrete data $D = (d_{ij})$, $F_k = \rho^T \Phi_k \Theta_k = (f_{k,ij})$ and

$$P\left(d_{ji} = a \mid f_{k,ij} = b\right) = \begin{cases} \alpha & \text{if } a = 1, b = 0 \\ 1 - \alpha & \text{if } a = 0, b = 0 \\ \beta & \text{if } a = 0, b = 1 \\ 1 - \beta & \text{if } a = 1, b = 1 \end{cases}.$$

Analogously we calculate the likelihood of the mixture by

$$\mathcal{L} = \prod_{i=1}^{l} \sum_k \pi_k \prod_{j=1}^{m} P\left(d_{ji} \mid f_{k,ij}\right).$$

In contrast to the log odds we cannot just weight the data matrix by the responsibilities. However, we can use the algorithm for the log odds by turning the error probabilities and the discrete data into log odds. We turn all ones in the data matrix into $log\left(\frac{1-\beta}{\alpha}\right)$ and all zeros into $log\left(\frac{\beta}{1-\alpha}\right)$.

If cells exist, which have been perturbed by a knock-down of more than one S-gene, values greater than 1 in F_k have to be normalized to 1. This is independent of discrete or continuous data.

7 Expectation of the complete data log likelihood for large data sets

The standard mixture NEM uses the EM algorithm to optimize the observed data log odds 1 of the model compared to the null model N (no effects).

$$\mathcal{L} = \sum_{i=1}^{l} \log \sum_{k=1}^{K} \pi_k \prod_{j=1}^{m} \frac{P\left(d_{ji} \mid f_{k,ij}\right)}{P\left(d_{ji} \mid N\right)}. \qquad 1$$

However, our data $R = (r_{ji})$ is already in log odds to make computation more efficient.

$$r_{ji} = \log \frac{P\left(d_{ji} \mid 1\right)}{P\left(d_{ji} \mid N\right)}.$$

Hence, we need to apply the exponential function $\exp$ for the sum of fractions in the observed log odds 1.

$$\mathcal{L} = \sum_{i=1}^{l} \log \sum_{k=1}^{K} \pi_k \exp \sum_{j=1}^{m} \log \frac{P\left(d_{ji} \mid f_{k,ij}\right)}{P\left(d_{ji} \mid N\right)} = \sum_{i=1}^{l} \log \sum_{k=1}^{K} \pi_k \exp \sum_{j=1}^{m} f_{ij,k} r_{ji}.$$

However, the summation of the log odds can lead to such large numbers, that using $\exp$ on them returns infinity in practical applications. This is mostly the case for data sets with many E-genes and/or large effect sizes. To avoid that problem, we optimize the expectation of the complete data log odds 2 instead.

$$\begin{aligned} \mathcal{L} &= \sum_{i=1}^{l} \sum_{k=1}^{K} \gamma_{ik} \left(\log \pi_k + \sum_{j=1}^{m} \log \frac{P\left(d_{ji} \mid f_{k,ij}\right)}{P\left(d_{ji} \mid N\right)} \right) \\ &= \sum_{i=1}^{l} \sum_{k=1}^{K} \gamma_{ik} \left(\log \pi_k + \sum_{j=1}^{m} f_{ij,k} r_{ji} \right). \end{aligned} \qquad 2$$

This way we never have to use $\exp$ on the sum of log odds. However, to get γ_{ik} we do still have to apply $\exp$, since

$$\gamma_{ki} = \frac{\pi_k \exp\overbrace{\left(\sum_{j=1}^{m} f_{ij,k} r_{ji}\right)}^{=l_{ii,k}}}{\sum_{s=1}^{K} \pi_s \exp\underbrace{\left(\sum_{j=1}^{m} f_{ij,s} r_{ji}\right)}_{=l_{ii,s}}}.$$

To solve that problem, we use the properties of $\exp$.

$$\gamma_{ki} = \frac{\pi_k \exp\left(l_{ii,k}\right)}{\sum_{s=1}^{K} \pi_s \exp\left(l_{ii,s}\right)} = \frac{C_i \pi_k \exp\left(l_{ii,k}\right)}{C_i \left(\sum_{s=1}^{K} \pi_s \exp\left(l_{ii,s}\right)\right)} = \frac{\pi_k \exp\left(l_{ii,k} + \log C_i\right)}{\sum_{s=1}^{K} \pi_s \exp\left(l_{ii,s} + \log C_i\right)}.$$

C_i should be chosen as such, that any $\exp\left(l_{ii,t} + \log C_i\right)$ returns a finite real number. In practice we do this by normalising, such that

$$\max_s \left\{\exp\left(l_{ii,s} + \log C_i\right)\right\} \approx \frac{10^3 08}{K}.$$

In theory this could lead to very small numbers and push several of the responsibilities to 0. However, in practice we don't consider this a problem. We reason, that if the cell log odds for one component k are very large, then they should either be large for all components or already close to 0 before normalisation.

8 References:

Datlinger, P., Rendeiro, A., Schmidl, C., Krausgruber, T., Traxler, P., Klughammer, J., Schuster, L. C., Kuchler, A., Alpar, D., and Bock, C. (2017). Pooled crispr screening with single-cell transcriptome readout. Nature Methods, 14, 297-301.

Dixit, A., Parnas, O., Li, B., Chen, J., Fulco, C. P., Jerby-Arnon, L., Marjanovic, N. D., Dionne, D., Burks, T., Raychowdhury, R., Adamson, B., Norman, T. M., Lander, E. S., Weissman, J. S., Friedman, N., and Regev, A. (2016). Perturb-seq: Dissecting molecular circuits with scalable single-cell rna profiling of pooled genetic screens. Cell, 167(7), 1853-1866.e17.

Markowetz, F., Bloch, J., and Spang, R. (2005). Non-transcriptional pathway features reconstructed from secondary effects of rna interference. Bioinformatics, 21(21), 4026–4032.

Markowetz, F., Kostka, D., Troyanskaya, O. G., and Spang, R. (2007). Nested effects models for high-dimensional phenotyping screens. Bioinformatics, 23(13), i305–i312.

Pirkl, M., Beerenwinkel, N.; Single cell network analysis with a mixture of Nested Effects Models, Bioinformatics, Volume 34, Issue 17, 1 September 2018, Pages i964–i971, https://doi.org/10.1093/bioinformatics/bty602.

Pirkl, M., Hand, E., Kube, D., Spang, R.; Analyzing synergistic and non-synergistic interactions in signalling pathways using Boolean Nested Effect Models, Bioinformatics, Volume 32, Issue 6, 15 March 2016, Pages 893–900, https://doi.org/10.1093/bioinformatics/btv680

Ritchie ME, Phipson B, Wu D, Hu Y, Law CW, Shi W, Smyth GK (2015). "limma powers differential expression analyses for RNA-sequencing and microarray studies." Nucleic Acids Research, 43(7), e47.