Combinatorial Nested Effects Models

March 7, 2016

Contents

1 The package epiNEM

This package is an extension of the classic Nested Effects Models provided in package *nem*. Nested Effects Models is a pathway reconstruction method, which takes into account effects of downstream genes. Those effects are observed for every knockout of a pathway gene, and the nested structure of observed effects can then be used to reconstruct the pathway structure. However, classic Nested Effects Models do not account for double knockouts. In this package *epiNEM*, one additional layer of complexity is added. For every two genes, acting on one gene together, the relationship is evaluated and added to the model as a logic gate. Genetic relationships are represented by the logics OR (no relationship), AND (functional overlap), NOT (masking or inhibiting) and XOR (mutual prevention from acting on gene C).

1.1 Core functions

epiNEM() The main function *epiNEM* can be called using several different parameters. First of all, a filename is specified, containing a table with binary data. If no such data available, *random* needs to be specified. The vector *random* consists of the number of single as well as double knockouts, the number of reporter (downstream) genes, the false positive and false negative rate and lastly the number of replicates. With all those parameter, a corresponding dataset is simulated.

Furthermore, the user can choose whether to perform exhaustive search or use Greedy Hill Climbing (recommended for network > 5 genes),whereas for the latter a number of iterations has to be set as well.

epiNEM.PlotResults() In order to plot the winning pathway structure, *epiNEM.PlotResults* can be called. Its input is an object of epiNEM, containing information on the model structure, the introduced logics and its positions, the likelihood and the attachment of E-Genes.

```
## Warning: package 'tidyr' was built under R version 3.2.3
## Warning: package 'ggplot2' was built under R version 3.2.3
## Warning: package 'AnnotationDbi' was built under R version 3.2.3
## Warning: package 'BiocGenerics' was built under R version 3.2.2
## Warning: package 'Biobase' was built under R version 3.2.2
## Warning: package 'IRanges' was built under R version 3.2.2
## Warning: package 'S4Vectors' was built under R version 3.2.3
## Warning in .recacheSubclasses(def@className, def, doSubclasses, env): undefined
subclass "externalRefMethod" of class "expressionORfunction"; definition not updated
## Warning in .recacheSubclasses(def@className, def, doSubclasses, env): undefined
subclass "externalRefMethod" of class "functionORNULL"; definition not updated
```

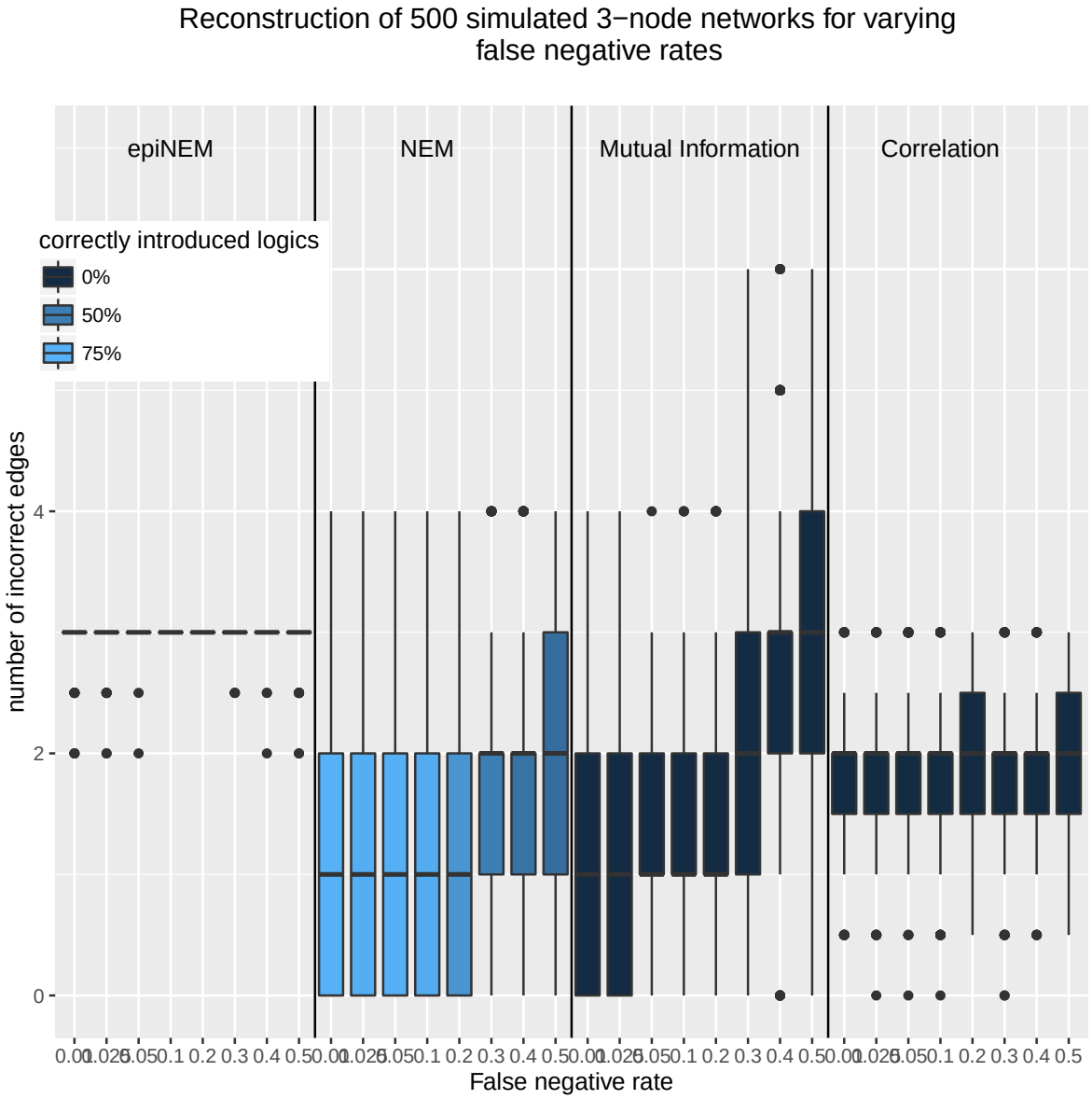
2 Validation studies

2.1 Comparing against other methods

In this section, we compare epiNEMs against the classic Nested Effects Models, as well as against network reconstruction by using Correlation and Mutual Information. We performed 500 simulations of 3 node networks for different false negative rates.

```
simul500 <- epiNEM.Simulations(random, 500)
```

Those networks were then learned by the above mentioned methods. For each method, we plot the number of wrong edges for each false negative rate.



3 Case Study

3.1 Data overview

The data provided in this package is based on the study of van Wageningen [reference]. The dataset consists of in total 150 viable protein kinase and phosphatase perturbations, as well as 15 viable double knockouts. DNA microarray expression profiles were generated under a single growth condition. For each mutant, two independent cultures have been measured and p-values

according to relative expression change as well as mutant/wildtype ratios recorded. The raw data input consists of a matrix of p-values indicating the relative change of gene expression for each perturbation. The columns indicate gene deletions, the rows represent all downstream genes that were investigated for exhibiting effects.

3.2 Running the main function

First we load the R packages and load the example data file from `http://www.holstegelab.nl/publications/sv/signaling_redundancy/`. The data file contains a gene expression matrix of all kinase phosphatase mutants. For the analysis, a small number of genes can be selected, which we specify in a 'genelist'.

```
#library(epiNEM)
library(magrittr)
library(tidyr)
library(ggplot2)
library(data.table)
library(BoolNet)
library(igraph)
library(e1071)
library(grid)
library(org.Sc.sgd.db)
devtools::load_all()
library(dplyr)
```

```
## load data file from local copy or from URL
address <-
  "http://www.holstegelab.nl/publications/sv/signaling_redundancy/downloads/DataS1.txt"
data <- data.table(read.csv(url(address), sep="\t", header=TRUE))
genelist <- toupper(c('bck1', 'slt2', 'ptp2', 'bck1.slt2'))

## read in genes from user and prepare for further analysis
read_in_genes <- function(genes){
  return(unlist(lapply(genes, function(x) {paste(x, '.del.vs..wt.1', sep='')})))
}
single <- read_in_genes(genelist)
```

Also, we only consider changes in expression which result in a p-value < 0.05. All those are set to 1, the other ones to 0.

```
## numericalize factors
as.numeric.factor <- function(x) {as.numeric(levels(x))[x]}
```

Last, we put all constraints together, meaning, we select and binarize the data.

```r
data %<>% dplyr::select(-c(systematicName, geneSymbol)) %>%
  filter(-1) %>%
  mutate_each(funs(droplevels)) %>%
  mutate_each(funs(as.numeric.factor)) %>%
  data.frame()

# consider foldchange and gene expression
# 0.73 = log2(1.7)
new <- matrix(0, ncol=dim(data)[2]/2, nrow=dim(data)[1])
colnames(new) <- colnames(data)[1:(dim(data)[2]/2)]
n=1
  for (i in seq(1, dim(data)[2], by=2)){
    colnames(new)[n] <- colnames(data)[i+1]
    for (k in 1:dim(data)[1]){
      if (((abs(data[k,i])) >= 0.76) & (data[k,i+1] < 0.05)) # & data[k,i+1] > 0))
        new[k,n] <- 1
      else new[k,n] <- 0
    }
    n = n+1
  }
# filter out 0-rows and set new colnames
new3 <- data.table(new[,which(colSums(new)!=0)])
#new5 <- data.table(new3[which(rowSums(new3)!=0),])
new4 <- new3 %>% (function(x){setnames(x, names(x), toupper(names(x)) %>%
  gsub('.DEL.VS..WT.1', '', .) %>%
  gsub('.VS..WT', '',.) %>%
  gsub('.DEL.', '.', .)); return(x)})

mutants     <- names(new4)
experiments <- mutants[!grepl("\\.", mutants)]
dKO         <- mutants[which(!mutants %in% experiments)]
dKO         <- dKO[!grepl('\\..*\\.', dKO)]
sKO <- unlist(strsplit(dKO, ".", fixed=TRUE))

sKO_add <- sKO[which(!sKO %in% names(new4))]

for (i in 1:length(sKO_add)){
  new4[, sKO_add[i]] <- c(rep(0,dim(new4)[1]))
}

PWadd <- genelist[which(!genelist %in% names(new4))]
for( i in 1:length(PWadd)){
  new4[, PWadd[i]] <- c(rep(0, dim(new4)[1]))
}

pathwayData <- new4 %>%
```

```
  select_(.dots = genelist) %>%
  filter(rowSums(.) > 0)

# short glimpse of the data
head(pathwayData)

##    BCK1 SLT2 PTP2 BCK1.SLT2
## 1:    1    0    0         0
## 2:    1    0    0         0
## 3:    0    1    0         0
## 4:    1    0    0         0
## 5:    1    0    0         0
## 6:    1    0    0         0
```

Rownames correspond to sample identifiers. Columns indicate the mutants. A 1 corresponds to a change in expression for an identifier at a knockdown, a 0 represents no effect.

An already processed version of the complete dataset is provided in this package and accessible by calling *epiNEMdata*.

```
lognem <- epiNEM(filename=pathwayData, method = "exhaustive")#, nIterations=3)
```

epiNEM introduces a logic gate for every node, which has exactly two parents. This logic gate represents the relationship of the parents, the interaction. If there is no interaction, as assumed in the classic NEMs, an OR-gate is introduced. Functional overlap of two genes is represented by an AND-gate, meaning both parents must be knocked out at the same time, in order for the signal to be interrupted. The logics NOTA and NOTB stand for a so called masking effect and the XOR-gate can be interpreted as some kind of mutual prevention of the parents from acting on the child gene.

The search space of epiNEM is now extended by those logic gates, and each model evaluated. Output of the algorithm is all information on the highest-scoring pathway structure.

```
#lognem <- unlist(lognem, recursive = FALSE)
lognem

## [[1]]
## [[1]]$origModel
##      BCK1 PTP2 SLT2
## BCK1    0    1    1
## PTP2    0    0    0
## SLT2    0    1    0
##
## [[1]]$model
##           BCK1 PTP2 SLT2
## BCK1         1    1    1
## SLT2         0    1    0
## PTP2         0    0    1
## BCK1.SLT2    1    0    1
```

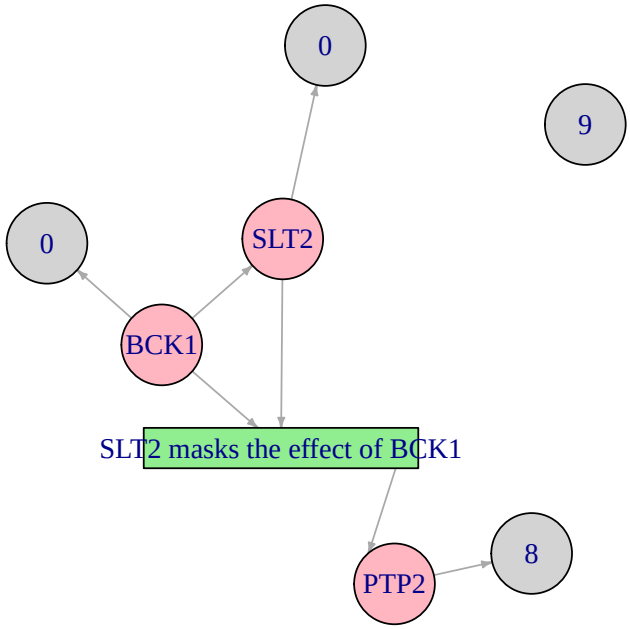
```
##
## [[1]]$logics
##                             Var1
## "SLT2 masks the effect of BCK1"
##
## [[1]]$column
##      c
## [1,] 2
##
## [[1]]$score
## [1] -41.14147
##
## [[1]]$EGeneset
##      noE
## BCK1   0
## PTP2   8
## SLT2   0
## null   9
```

```
lognem <- unlist(lognem, recursive = FALSE)
```

3.3 Plotting the result

This winning pathway structure can further be visualized by the function PlotResults(), which is also available with the package epiNEM. The grey circles, which are attached to the genes, denote the number of E-genes attached to them. The unconnected circle includes all those, which could not uniquely be attached to a single S-gene.

```
epiNEM.PlotResults(lognem)
```



3.4 Triplet screening

For analysing the mixed epistasis found by van Wageningen et. al., we constructed networks of only three nodes for every double knockout. Or, we combined each pair with any remaining gene and evaluate each of those small networks. Depending on the introduced logic, we were able to confirm most experimental results. First we need to define the double knockout of interest – in this example we choose ptp2 and ptp3 – and combine it with all remaining single experiments in a vector called currentSet.

```
#  parents     <- 'PTP2.PTP3'
#  parents2    <- unlist(strsplit(parents, ".", fixed=TRUE))
#  mutants     <- names(full_matrix)
#  experiments <- mutants[!grepl("\\.", mutants)]
```

```
#  dKO          <- mutants[which(!mutants %in% experiments)]
#  dKO          <- dKO[!grepl('\\..*\\.', dKO)]
#  NOT2 <- paste(parents2[2], "masks the effect of", parents2[1])
#  NOT1 <- paste(parents2[1], "masks the effect of", parents2[2])

#  currentSet <- c(experiments, parents)
```

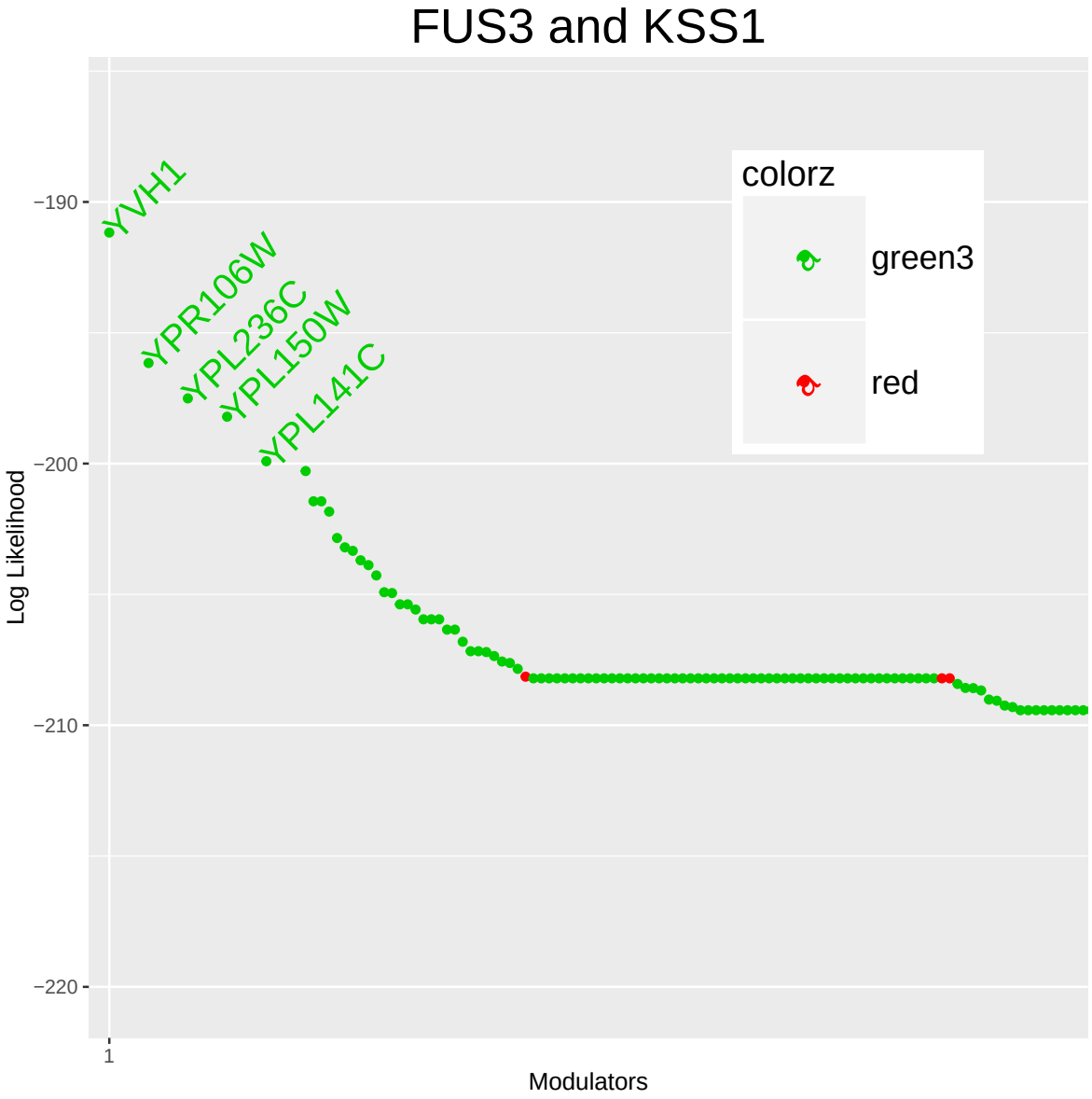
We modify our dataset in a way that it only contains the double knockout of interest plus single knockouts. In addition, we keep a common E-Geneset for better overall comparison of our results. We keep all E-Genes showing effect in either single or double mutant in combination of interest.

Then, we call epiNEM for every triple in the dataset.

```
mutants     <- names(new4)
experiments <- mutants[!grepl("\\.", mutants)]
dKO         <- mutants[which(!mutants %in% experiments)]
dKO         <- dKO[!grepl('\\..*\\.', dKO)]

#for(tri in 1:length(dKO)){
parents     <- dKO[5]
parents2    <- unlist(strsplit(parents, ".", fixed=TRUE))
NOT2 <- paste(parents2[2], "masks the effect of", parents2[1])
NOT1 <- paste(parents2[1], "masks the effect of", parents2[2])

currentSet <- c(experiments, parents)

trip_data   <- new4 %>% #tbl_df() %>%
    select_(., .dots = currentSet)

EGcols <- which(colnames(trip_data) %in% c(parents, parents2))

trip_data$ID <- as.numeric(rownames(trip_data))

## common EGeneset containing only E-genes that are affected at
## either double or single mutant
EG <- trip_data %>%
   dplyr::select(EGcols, ID) %>%
   filter((rowSums(.)-ID) > 0)

EGdata <- as.data.frame(trip_data)[which(trip_data$ID %in% EG$ID),]

  scores <- c()
  singles <- experiments[which(!experiments %in% parents2)]
for (i in 1:(length(singles)-1)){
  mutants1    <- c(singles[i], parents2, parents)
  experiments1 <- mutants1[1:length(mutants1)-1]
```

```
  modelsTrip1 <- EnumerateModels(length(experiments1), experiments1)
  EGdata2     <- EGdata[,which(colnames(trip_data) %in% mutants1)]
  scores[[i]] <- epiNEM(filename = EGdata2, method = "exhaustive")
  #load('SAgain_scores6_12.RData')
}
```

And finally we can plot the marginal likelihood and the logic included for evaluation. In this example it shows clearly, that only AND-logic gates were introduced to every single model. This overlaps with the results from van Wageningen et. al. who found out experimentally, that ptp2 with ptp3 always behave in complete redundancy.



4 Session Info

```
sessionInfo()

## R version 3.2.1 (2015-06-18)
## Platform: x86_64-apple-darwin13.4.0 (64-bit)
## Running under: OS X 10.10.4 (Yosemite)
##
## locale:
## [1] C
##
## attached base packages:
##  [1] parallel  stats4    grid      stats     graphics  grDevices utils
##  [8] datasets  methods   base
##
## other attached packages:
##  [1] epiNEM_0.0.0.9000    dplyr_0.4.3          org.Sc.sgd.db_3.2.3
##  [4] RSQLite_1.0.0        DBI_0.3.1            AnnotationDbi_1.32.3
##  [7] IRanges_2.4.6        S4Vectors_0.8.7      Biobase_2.30.0
## [10] BiocGenerics_0.16.1  e1071_1.6-7          igraph_1.0.1
## [13] BoolNet_2.1.1        data.table_1.9.6     ggplot2_2.0.0
## [16] tidyr_0.4.0          magrittr_1.5         knitr_1.12
##
## loaded via a namespace (and not attached):
##  [1] Rcpp_0.12.3      formatR_1.2.1    plyr_1.8.3       highr_0.5.1
##  [5] class_7.3-14     tools_3.2.1      digest_0.6.9     evaluate_0.8
##  [9] memoise_0.2.1    gtable_0.1.2     stringr_1.0.0    roxygen2_5.0.1
## [13] devtools_1.9.1   R6_2.1.1         XML_3.98-1.3     scales_0.3.0
## [17] assertthat_0.1   colorspace_1.2-6 labeling_0.3     stringi_1.0-1
## [21] lazyeval_0.1.10  munsell_0.4.2    chron_2.3-47
```