

Epistatic Nested Effects Models

Inferring mixed epistasis from indirect measurements of knockout screens.

Madeline Diekmann & Martin Pirkl

October 18, 2016

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).

Loading epiNEM

```
## install.packages("devtools", verbose = F, quiet = T)

library(devtools)

## install_github("cbg-ethz/epiNEM", quiet = T)

library(epiNEM)
```

Simulations

We compare epiNEM to several network inference methods.

```
library(bnem) # install_github("MartinFXP/B-NEM/package")

## Loading required package: CellNOptR
## Loading required package: RBGL
## Loading required package: graph
## Loading required package: BiocGenerics
## Loading required package: parallel
##
## Attaching package: 'BiocGenerics'
## The following objects are masked from 'package:parallel':
##
##   clusterApply, clusterApplyLB, clusterCall, clusterEvalQ, clusterExport,
```

```

##      clusterMap, parApply, parCapply, parLapply, parLapplyLB, parRapply,
##      parSapply, parSapplyLB
## The following objects are masked from 'package:igraph':
##
##      normalize, union
## The following objects are masked from 'package:stats':
##
##      IQR, mad, xtabs
## The following objects are masked from 'package:base':
##
##      Filter, Find, Map, Position, Reduce, anyDuplicated, append, as.data.frame,
##      cbind, colnames, do.call, duplicated, eval, evalq, get, grep, grepl,
##      intersect, is.unsorted, lapply, lengths, mapply, match, mget, order, paste,
##      pmax, pmax.int, pmin, pmin.int, rank, rbind, rownames, sapply, setdiff,
##      sort, table, tapply, union, unique, unsplit
##
## Attaching package: 'graph'
## The following objects are masked from 'package:igraph':
##
##      degree, edges, intersection
##
## Attaching package: 'RBGL'
## The following objects are masked from 'package:igraph':
##
##      bfs, dfs, transitivity
## The following object is masked from 'package:e1071':
##
##      extractPath
## Loading required package: hash
## hash-2.2.6 provided by Decision Patterns
## Loading required package: ggplot2
## Loading required package: RCurl
## Loading required package: bitops
## Loading required package: Rgraphviz
## Loading required package: grid
## Loading required package: XML
##
## Attaching package: 'XML'
## The following object is masked from 'package:graph':
##
##      addNode
## Loading required package: nem
##
## Attaching package: 'nem'
## The following object is masked from 'package:RBGL':
##
##      transitive.closure
## Loading required package: matrixStats
## matrixStats v0.50.2 (2016-04-24) successfully loaded. See ?matrixStats for help.
## Loading required package: snowfall
## Loading required package: snow
##
## Attaching package: 'snow'

```

```

## The following objects are masked from 'package:BiocGenerics':
##
##   clusterApply, clusterApplyLB, clusterCall, clusterEvalQ, clusterExport,
##   clusterMap, clustersSplit, parApply, parCapply, parLapply, parRapply,
##   parSapply
## The following objects are masked from 'package:parallel':
##
##   clusterApply, clusterApplyLB, clusterCall, clusterEvalQ, clusterExport,
##   clusterMap, clustersSplit, makeCluster, parApply, parCapply, parLapply,
##   parRapply, parSapply, splitIndices, stopCluster
## Loading required package: latticeExtra
## Loading required package: lattice
## Loading required package: RColorBrewer
##
## Attaching package: 'latticeExtra'
## The following object is masked from 'package:ggplot2':
##
##   layer

library(nem)

library(minet)

library(pcalg)

```

```

runs <- 100

noiselvls <- c(0.01, 0.025, 0.05, 0.1, 0.2, 0.3, 0.4, 0.5)

random <- list(FPrate = 0.1, FNrate = noiselvls, single = 4, double = 1, reporters = 100, replicates = 3)

spec <- sens <- logics <- array(0, dim = c(2, runs, length(noiselvls)))

sens2 <- spec2 <- time <- array(0, dim = c(5, runs, length(noiselvls)))

do <- c("n", "p", "a")

do <- c("e", "b", do)

popSize <- 100

maxTime <- F

forcelogic <- T

epinemsearch <- "greedy"

nIterations <- 3

bnemsearch <- "genetic"

parallel <- NULL

```

```

logicgate <- matrix("", runs, length(noiselvls))

edgenr <- matrix(0, runs, length(noiselvls))

## for (i in 1:runs) {

##     print(paste("run ", i, sep = ""))

##     for (j in 1:length(noiselvls)) {

##         print(paste("noiselvl ", j, sep = ""))

##         topology <- CreateTopology(random$single, random$double, force = forcelogic)

##         topology <- unlist(unique(topology), recursive = FALSE)

##         extTopology <- ExtendTopology(topology$model, random$reporters)

##         sortedData <- GenerateData(topology$model, extTopology, random$FPrate, random$FNrate[j], ran

##         logicgate[i, j] <- paste(topology$logics, collapse = "_")

##         edgenr[i, j] <- sum(topology$origModel == 1)

##         if ("e" %in% do) {
##             print("epiNEM")

##             start <- Sys.time()
##             TriplModel <- epiNEM(filename = sortedData, method = epinemsearch, nIterations = nIterat
##             time[1, i, j] <- difftime(Sys.time(), start, units = "secs")
##             print(time[1, i, j])

##             tp <- sum(topology$model == 1 & TriplModel$model == 1)
##             tn <- sum(topology$model == 0 & TriplModel$model == 0)
##             fp <- sum(topology$model == 0 & TriplModel$model == 1)
##             fn <- sum(topology$model == 1 & TriplModel$model == 0)
##             sens[1, i, j] <- tp/(tp+fn)
##             spec[1, i, j] <- tn/(tn+fp)
##             tp <- sum(topology$origModel == 1 & TriplModel$origModel == 1)
##             tn <- sum(topology$origModel == 0 & TriplModel$origModel == 0)
##             fp <- sum(topology$origModel == 0 & TriplModel$origModel == 1)
##             fn <- sum(topology$origModel == 1 & TriplModel$origModel == 0)
##             sens2[1, i, j] <- tp/(tp+fn)
##             spec2[1, i, j] <- tn/(tn+fp)
##             tp <- 0
##             for (k in 1:length(topology$column)) {
##                 for (l in 1:length(TriplModel$column)) {
##                     if (topology$column[k] == TriplModel$column[l]) {
##                         if (topology$logics[k] %in% TriplModel$logics[l]) {
##                             tp <- tp + 1
##                         }
##                     }
##                 }
##             }
##         }

```

```

##      }
##      logics[1, i, j] <- tp/(length(topology$logics) + length(TriplModel$logics) - tp)
##      print(sens[1, i, j])
##      print(spec[1, i, j])
##      print(sens2[1, i, j])
##      print(spec2[1, i, j])
##      print(logics[1, i, j])
##
##    }

##    if ("b" %in% do) {
##      print("B-NEM")

##      gtn <- epi2bg(topology)

##      fc <- cbind(Ctrl_vs_S = -1, epi2bg(sortedData))*(-1)

##      bnemnoise <- sample(1:nrow(fc), floor(nrow(fc)*random$FNrate[j]))

##      fc[bnemnoise, 1] <- 0

##      ers <- t(topology$model)*(-1)
##      colnames(ers) <- paste("S_vs_S_", gsub("\\.", "_", colnames(ers)), sep = "")
##      ers <- cbind(Ctrl_vs_S = 1, ers)
##      ers <- ers[, order(colnames(ers))]

##      CNOList <- dummyCNOList(stimuli = "S", inhibitors = LETTERS[1:random$single], maxStim = 1)

##      parents <- unique(unlist(strsplit(colnames(sortedData)[grep("\\.", colnames(sortedData))], "\\."))

##      nodes <- unique(colnames(sortedData)[-grep("\\.", colnames(sortedData))])

##      child <- nodes[-which(nodes %in% parents)]

##      sifMatrix <- NULL
##      for (k in LETTERS[1:random$single]) {
##        sifMatrix <- rbind(sifMatrix, c("S", "1", k))#, c("S", "-1", k)) # bnem can set a priori
##        for (l in LETTERS[1:random$single]) {
##          if (k %in% l) { next() }
##          if (k %in% parents) {
##            sifMatrix <- rbind(sifMatrix, c(k, "1", l), c(k, "-1", l))
##          } else {
##            sifMatrix <- rbind(sifMatrix, c(k, "1", l))
##          }
##        }

##      randfile <- paste("pkn_", as.numeric(Sys.time()), sep = "")
##      write.table(sifMatrix, file = randfile, sep = "\t",
##                row.names = FALSE, col.names = FALSE, quote = FALSE)
##      PKN <- readSIF(randfile)
##      unlink(randfile)

##      model <- preprocessing(CNOList, PKN)

```

```

##      initBstring <- absorption(rep(1, length(model$reacID)), model)

##      if (maxTime) { maxTime2 <- time[1, i, j] } else { maxTime2 <- Inf }

##      start <- Sys.time()
##      bga <- bnem(search = bnemsearch,
##                fc=fc,
##                CN0list=CN0list,
##                model=model,
##                initBstring=initBstring,
##                draw = F,
##                verbose = F,
##                popSize = popSize,
##                maxTime = maxTime2,
##                parallel = parallel
##                )
##      time[2, i, j] <- difftime(Sys.time(), start, units = "secs")
##      print(time[2, i, j])

##      ers2 <- computeFc(CN0list, t(simulateStatesRecursive(CN0list, model, bga$bString)))
##      ers2 <- ers2[, unique(colnames(fc))]
##      ers2 <- ers2[, order(colnames(ers2))]

##      tp <- sum(ers == -1 & ers2 == -1)
##      tn <- sum(ers == 0 & ers2 == 0)
##      fn <- sum(ers == -1 & ers2 == 0)
##      fp <- sum(ers == 0 & ers2 == -1)
##      sens[2, i, j] <- tp/(tp+fn)
##      spec[2, i, j] <- tn/(tn+fp)
##      gtn2 <- abs(dnf2adj(gtn))
##      if (length(grep("S", rownames(gtn2))) > 0) {
##        gtn2 <- gtn2[-grep("S", rownames(gtn2)), -grep("S", colnames(gtn2))]
##      }
##      gtn2 <- gtn2[order(rownames(gtn2)), order(colnames(gtn2))]
##      res <- abs(dnf2adj(bga$graph))
##      if (length(grep("S", rownames(res))) > 0) {
##        res <- as.matrix(res[-grep("S", rownames(res)), -grep("S", colnames(res))])
##      }
##      if (dim(res)[1] == 1) {
##        colnames(res) <- rownames(res) <- gsub(".*=", "", bga$graph)
##      } else {
##        res <- res[order(rownames(res)), order(colnames(res))]
##      }
##      if (nrow(res) < nrow(gtn2)) {
##        res2 <- rbind(cbind(res, matrix(0, nrow(res), nrow(gtn2) - nrow(res))), matrix(0, nrow(res), nrow(gtn2) - nrow(res)))
##        colnames(res2)[(ncol(res)+1):ncol(res2)] <- colnames(gtn2)[which(!(colnames(gtn2) %in% colnames(res2)))]
##        rownames(res2)[(nrow(res)+1):nrow(res2)] <- rownames(gtn2)[which(!(rownames(gtn2) %in% rownames(res2)))]
##        res2 <- res2[order(rownames(res2)), order(colnames(res2))]
##        res <- res2
##      }
##      diag(gtn2) <- diag(res) <- 0
##      tp <- sum(gtn2 == 1 & res == 1)
##      tn <- sum(gtn2 == 0 & res == 0)

```

```

##      fn <- sum(gtn2 == 1 & res == 0)
##      fp <- sum(gtn2 == 0 & res == 1)
##      sens2[2, i, j] <- tp/(tp+fn)
##      spec2[2, i, j] <- tn/(tn+fp)
##      tp <- sum(bga$graph %in% gtn)
##      logics[2, i, j] <- tp/(length(gtn) + length(bga$graph) - tp) # (tp/(tp+fn) + tn/(tn+fp))
##      print(sens[2, i, j])
##      print(spec[2, i, j])
##      print(sens2[2, i, j])
##      print(spec2[2, i, j])
##      print(logics[2, i, j])

##      print(bga$graph)
##      print(gtn)

##    }

##    if (any(c("n", "p", "a") %in% do)) {

##      reddata <- sortedData[, -grep("\\.", colnames(sortedData))]
##      gtnadj <- topology$origModel
##      gtnadj <- gtnadj[order(apply(gtnadj, 1, sum), decreasing = T), order(apply(gtnadj, 2, sum))]
##      gtnadj[lower.tri(gtnadj)] <- gtnadj[upper.tri(gtnadj)]
##      gtnadj <- gtnadj[order(rownames(gtnadj)), order(colnames(gtnadj))]
##      eadj <- topology$origModel
##      eadj <- eadj[order(rownames(eadj)), order(colnames(eadj))]
##      reddata2 <- matrix(0, nrow(reddata)*random$replicates, length(unique(colnames(reddata))))
##      for (k in 1:length(unique(colnames(reddata)))) {
##        reddata2[, k] <- as.vector(reddata[, which(colnames(reddata) %in% unique(colnames(reddata))))
##      }
##      colnames(reddata2) <- unique(colnames(reddata))

##    }

##    if ("n" %in% do) {
##      print("NEM")

##      start <- Sys.time()
##      if (epinemsearch %in% "greedy") {
##        nemres <- nem(reddata, inference = "nem.greedy")
##      } else {
##        nemres <- nem(reddata, inference = "search")
##      }
##      nadj <- transitive.reduction(graph2adj(nemres$graph))
##      time[3, i, j] <- difftime(Sys.time(), start, units = "secs")
##      print(time[3, i, j])

##      tp <- sum(eadj == 1 & nadj == 1)
##      tn <- sum(eadj == 0 & nadj == 0)
##      fp <- sum(eadj == 0 & nadj == 1)
##      fn <- sum(eadj == 1 & nadj == 0)
##      sens2[3, i, j] <- tp/(tp+fn)
##      spec2[3, i, j] <- tn/(tn+fp)

```

```

##          print(sens2[3, i, j])
##          print(spec2[3, i, j])

##      }

##      if ("p" %in% do) {
##          print("PCalg")

##          start <- Sys.time()
##          pc.fit <- pc(suffStat = list(C = cor(reddata2), n = nrow(reddata2)),
##              indepTest = gaussCIttest, ## indep.test: partial correlations
##              alpha=0.05, labels = colnames(reddata2), verbose = F)
##          pcadj <- graph2adj(pc.fit@graph)
##          time[4, i, j] <- difftime(Sys.time(), start, units = "secs")
##          print(time[4, i, j])

##          tp <- sum(gtnadj == 1 & pcadj == 1)
##          tn <- sum(gtnadj == 0 & pcadj == 0)
##          fp <- sum(gtnadj == 0 & pcadj == 1)
##          fn <- sum(gtnadj == 1 & pcadj == 0)
##          sens2[4, i, j] <- tp/(tp+fn)
##          spec2[4, i, j] <- tn/(tn+fp)
##          print(sens2[4, i, j])
##          print(spec2[4, i, j])

##      }

##      if ("a" %in% do) {
##          print("Aracne")

##          start <- Sys.time()
##          ares <- build.mim(reddata2)
##          ares <- aracne(ares)
##          ares <- disc(ares, 0)
##          ares <- ares[order(rownames(ares)), order(colnames(ares))]
##          nas <- which(is.na(ares) == T)
##          ares[nas] <- 0
##          diag(ares) <- 0
##          time[5, i, j] <- difftime(Sys.time(), start, units = "secs")
##          print(time[5, i, j])

##          tp <- sum(gtnadj == 1 & ares == 1)
##          tn <- sum(gtnadj == 0 & ares == 0)
##          fp <- sum(gtnadj == 0 & ares == 1)
##          fn <- sum(gtnadj == 1 & ares == 0)
##          sens2[5, i, j] <- tp/(tp+fn)
##          spec2[5, i, j] <- tn/(tn+fp)
##          print(sens2[5, i, j])
##          print(spec2[5, i, j])

##      }

##  }

```



```
## }
```

```
data(sim)

acc <- (sens + spec)/2

acc2 <- (sens2 + spec2)/2

m <- rbind(c(1,1), c(2,2), c(3,4))

layout(m)

timeframe <- as.data.frame(cbind(data.frame(epiNEM = time[1,,]), data.frame(BNEM = time[2,,]), data.frame(BNEM = time[3,,])))

colnames(timeframe) <- c(paste(rep("epi", length(noiselvls)), noiselvls, sep = "_"), paste(rep("B", length(noiselvls)), noiselvls, sep = "_"))

boxplot(timeframe, col = "pink", main = "running time", ylab = "seconds", log = "y")

abline(v=(1:(length(do)-1)*length(noiselvls) + 0.5), col = "black", lty = 3)

accframe2 <- as.data.frame(cbind(data.frame(epiNEM = acc2[1,,]), data.frame(BNEM = acc2[2,,]), data.frame(BNEM = acc2[3,,])))

colnames(accframe2) <- c(paste(rep("E", length(noiselvls)), noiselvls, sep = "_"), paste(rep("B", length(noiselvls)), noiselvls, sep = "_"))

boxplot(accframe2, col = "green", main = "accuracy of the inferred edges", ylim = c(0,1))

abline(v=(1:(length(do)-1)*length(noiselvls) + 0.5), col = "black", lty = 3)

logicsframe <- as.data.frame(cbind(data.frame(epiNEM = logics[1,,]), data.frame(BNEM = logics[2,,]), data.frame(BNEM = logics[3,,])))

colnames(logicsframe) <- c(paste(rep("E", length(noiselvls)), noiselvls, sep = "_"), paste(rep("B", length(noiselvls)), noiselvls, sep = "_"))

boxplot(logicsframe, col = "blue", main = "accuracy of the inferred logic gate", ylim = c(0,1))

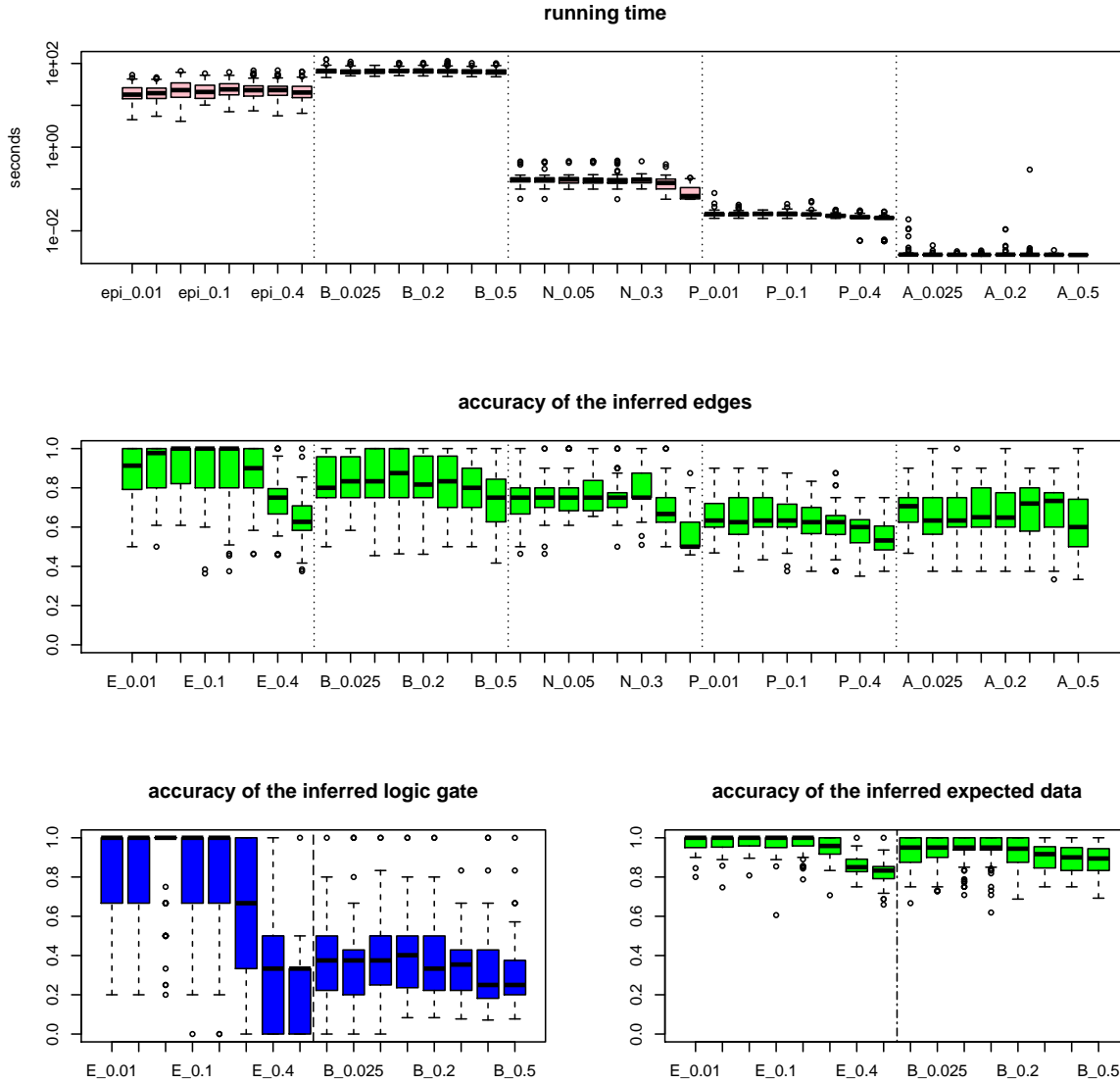
abline(v=length(noiselvls)+0.5, col = "black", lty = 5)

accframe <- as.data.frame(cbind(data.frame(epiNEM = acc[1,,]), data.frame(BNEM = acc[2,,]), data.frame(BNEM = acc[3,,])))

colnames(accframe) <- c(paste(rep("E", length(noiselvls)), noiselvls, sep = "_"), paste(rep("B", length(noiselvls)), noiselvls, sep = "_"))

boxplot(accframe, col = "green", main = "accuracy of the inferred expected data", ylim = c(0,1))

abline(v=length(noiselvls)+0.5, col = "black", lty = 6)
```



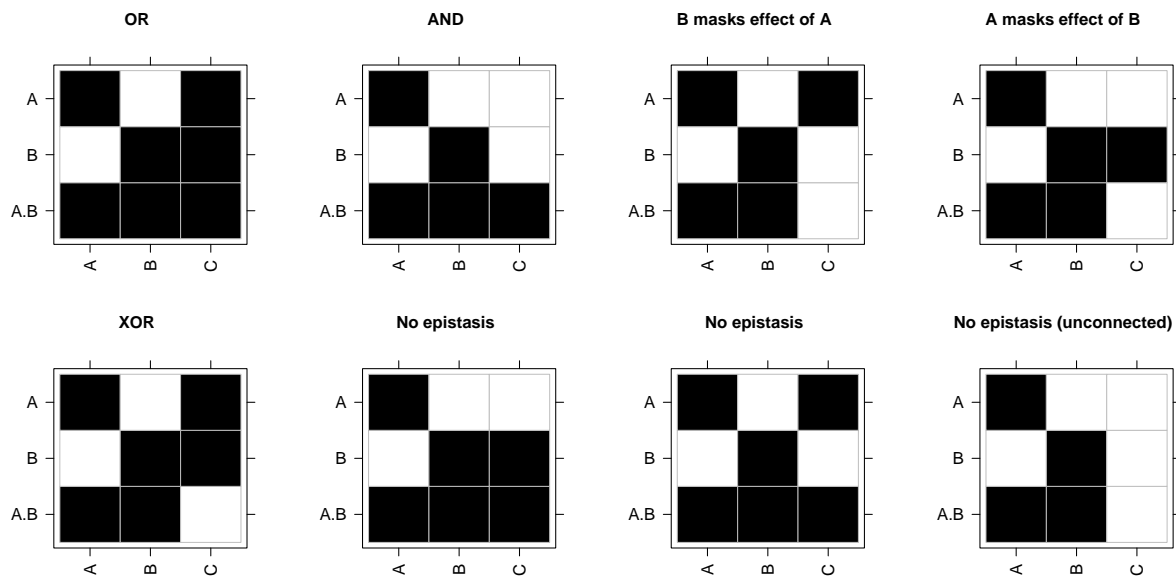
Yeast knockout screens

In this section we analyse previously published yeast knockout screens. The screens consist of gene expression data derived from double and single knockout mutants. We use epiNEM on each double mutant combined with each single mutant.

The results of the knockout screens have been annotated according to the following legend:

```
options(warn=-1)
heatmapOP(matrix(c(1,-1,1,-1,1,1, 1, 1, 1), 3, 3, dimnames = list(c("A", "B", "A.B"), LETTERS[1:3])), Co
heatmapOP(matrix(c(1,-1,1,-1,1,1, -1, -1, 1), 3, 3, dimnames = list(c("A", "B", "A.B"), LETTERS[1:3])), Co
heatmapOP(matrix(c(1,-1,1,-1,1,1, 1, -1, -1), 3, 3, dimnames = list(c("A", "B", "A.B"), LETTERS[1:3])), Co
heatmapOP(matrix(c(1,-1,1,-1,1,1, -1, 1, -1), 3, 3, dimnames = list(c("A", "B", "A.B"), LETTERS[1:3])), Co
heatmapOP(matrix(c(1,-1,1,-1,1,1, 1, 1, -1), 3, 3, dimnames = list(c("A", "B", "A.B"), LETTERS[1:3])), Co
heatmapOP(matrix(c(1,-1,1,-1,1,1, -1, 1, 1), 3, 3, dimnames = list(c("A", "B", "A.B"), LETTERS[1:3])), Co
```

```
heatmapOP(matrix(c(1,-1,1,-1,1,1, 1, -1, 1), 3, 3, dimnames = list(c("A", "B", "A.B"), LETTERS[1:3])), (
heatmapOP(matrix(c(1,-1,1,-1,1,1, -1, -1, -1), 3, 3, dimnames = list(c("A", "B", "A.B"), LETTERS[1:3])), (
options(warn=0)
```



Wageningen et al., 2010

```
data(wageningen)

dataM <- data[-(1:2), (1+(1:(324/2))*2)]

dataP <- data[-(1:2), (2+(1:(324/2))*2)]

dataM <- dataM[-1, ]

dataP <- dataP[-1, ]

dataM <- apply(dataM, c(1,2), as.numeric)

dataP <- apply(dataP, c(1,2), as.numeric)

dataBin <- dataM

sig <- 0.05

cutoff <- 0.7

dataBin[which(dataP < sig & dataP > 0 & abs(dataM) >= cutoff)] <- 1

dataBin[which(dataP >= sig | dataP == 0 | abs(dataM) < cutoff)] <- 0 # why do you throw away p-values w

dataBin <- dataBin[-which(apply(dataBin, 1, max) == 0), ]

genelist <- toupper(c('hs11', 'cla4', 'gin4', 'swe1', 'hs11.cla4'))
```

```

read_in_genes <- function(genes){
  return(unlist(lapply(genes, function(x) {paste(x, '.del.vs..wt.1', sep='')})))
}

single <- read_in_genes(genelist)

colnames(dataBin) <- gsub(".del.vs..wt", "", colnames(dataBin))

colnames(dataBin) <- gsub(".del", "", colnames(dataBin))

doubles <- colnames(dataBin)[grep("\\.", colnames(dataBin))]

doubles <- sort(doubles[-grep("vs", doubles)])

doubles.genes <- unique(unlist(strsplit(doubles, "\\.")))

singles <- colnames(dataBin)[-grep("\\.", colnames(dataBin))]

singles <- unique(sort(singles))

llmat <- logicmat <- matrix(0, length(singles), length(doubles))

rownames(llmat) <- rownames(logicmat) <- singles

colnames(llmat) <- colnames(logicmat) <- doubles

globalgenes <- which(apply(dataBin, 1, max) == 1)

## for (i in doubles[set]) {
##   if (which(doubles %in% i) == 8) { next() }
##   print(i)
##   doubles.singles <- unlist(strsplit(i, "\\."))
##   egenes <- which(apply(dataBin[, which(colnames(dataBin) %in% c(i, doubles.singles))], 1, max) == 1)
##   for (j in singles) {
##     print(j)
##     if (j %in% doubles.singles) { next() }

##     dataTmp <- dataBin[, grep(paste(paste("^", c(i, j, doubles.singles), "$", sep = ""), collapses

##     if (path %in% "fixed_set") {
##       dataTmp <- dataTmp[egenes, ]
##     }
##     if (path %in% "global") {
##       dataTmp <- dataTmp[globalgenes, ]
##     }
##     if (path %in% "") {
##       dataTmp <- dataTmp[which(apply(dataTmp, 1, max) == 1), ]
##     }

##     i1 <- which(singles %in% j)
##     i2 <- which(doubles %in% i)

##     if (!(is.null(dim(dataTmp)))) {

```

```

##             if (any(dataTmp[, j] != 0)) {
##
##             epires <- epiNEM(dataTmp, method = "exhaustive")
##
##             tmp <- epires$logics
##             if ("OR" %in% tmp) {
##                 if (sum(epires$origModel[, j]) != 2) {
##                     tmp <- "NOEPI"
##                 } else {
##                     if (all(tmp %in% "OR")) {
##                         tmp <- "OR"
##                     } else {
##                         tmp <- tmp[which(!(tmp %in% "OR"))]
##                     }
##                 }
##             }
##
##             logicmat[i1, i2] <- tmp
##             llmat[i1, i2] <- epires$score
##
##         } else {
##
##             logicmat[i1, i2] <- "UNCON"
##             llmat[i1, i2] <- -Inf
##
##         }
##
##     } else {
##
##         logicmat[i1, i2] <- "UNCON"
##         llmat[i1, i2] <- -Inf
##
##     }
##
## }
## }

```

Plot results.

```

data(wageningen_res)

llmat0 <- wageningen$ll

logicmat0 <- wageningen$logic

paperdoubles <- c(4, 9, 17)

for (i in 1:length(doubles)) {

    if (!(doubles[i] %in% c("ark1.prk1", "prk1.ark1", "ptp2.ptp3", "ptp3.ptp2", "bck1.ptp3", "ptp3.bck1")))

    if (i %in% 8) { next() }
}

```

```

logicvec <- logicmat0[, i]

llvec <- llmat0[, i]

logicvec <- logicvec[order(llvec, decreasing = T)]

llvec <- llvec[order(llvec, decreasing = T)]

parents <- unlist(strsplit(doubles[i], "\\\\"))

pchvec <- numeric(length(llvec))

pchvec[which(logicvec %in% "AND")] <- 1
pchvec[which(logicvec %in% "OR")] <- 2
pchvec[which(logicvec %in% "XOR")] <- 3
pchvec[grepl(paste("^", parents[1], sep = ""), logicvec)] <- 4
pchvec[grepl(paste("^", parents[2], sep = ""), logicvec)] <- 5
pchvec[which(logicvec %in% "NOEPI")] <- 6
pchvec[which(logicvec %in% "UNCON")] <- 7

logicvec <- logicvec[-which(logicvec %in% "0")]
pchvec <- pchvec[-which(pchvec == 0)]
llvec <- llvec[-which(llvec == 0)]

colvec <- pchvec

llvec[which(is.infinite(llvec) == T)] <- Inf

llvec[which(is.infinite(llvec) == T)] <- min(llvec) - 100

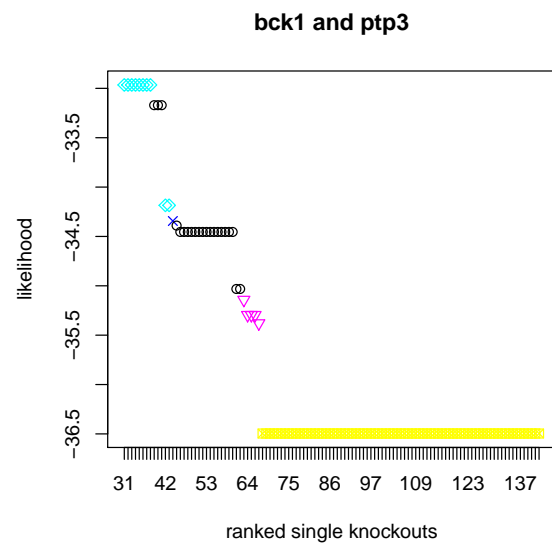
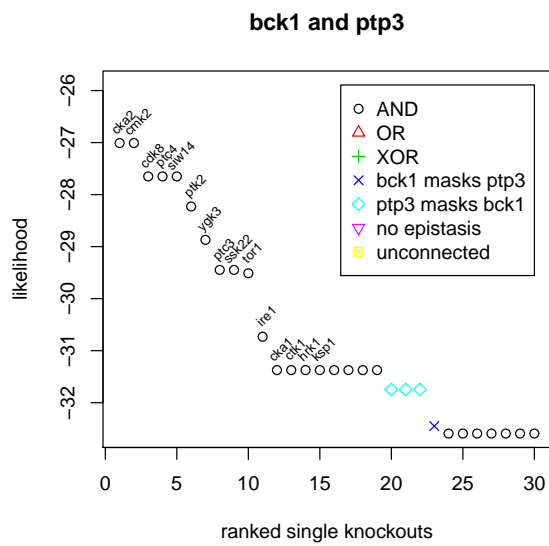
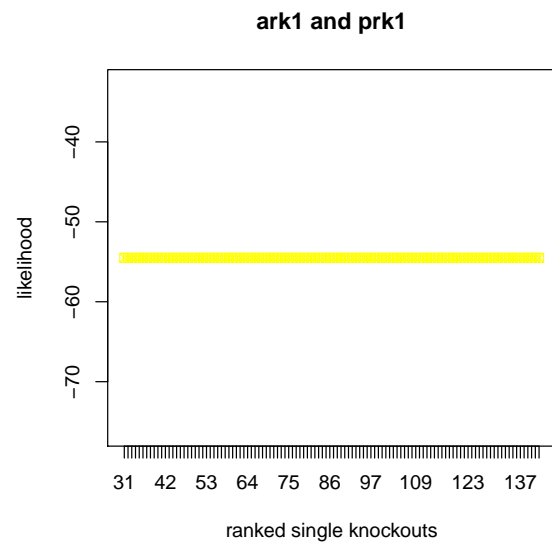
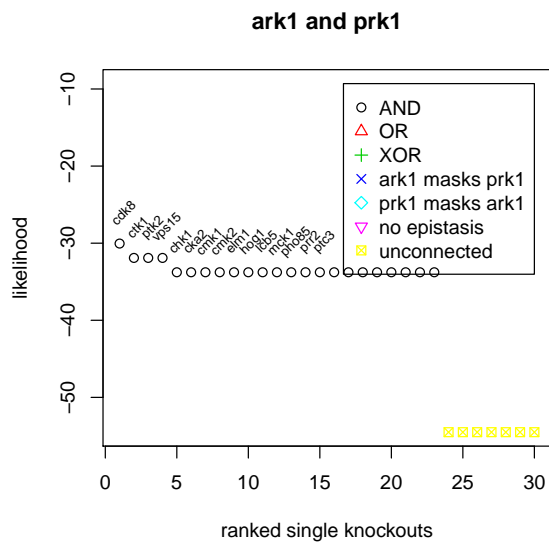
if (all(is.infinite(llvec) == T)) { llvec[1:length(llvec)] <- -1000 }

margin <- abs(max(llvec[1:30]) - min(llvec[1:30]))*0.2

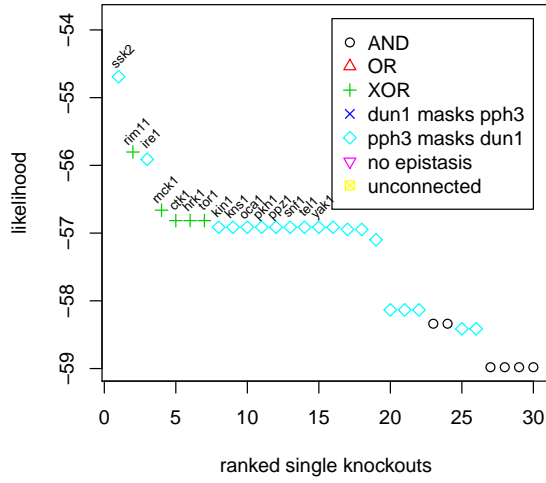
llvec[which(llvec == min(llvec))] <- min(llvec) + 100 - margin

par(mfrow=c(1,2))
plot(llvec[1:30], pch = pchvec[1:30], col = colvec[1:30], ylab = "likelihood", xlab = "ranked single
legend(30, max(llvec)+margin, legend = c("AND", "OR", "XOR", paste(parents[1], " masks ", parents[2],
text((1:30)+0.5, llvec[1:30]+(margin*0.1), labels = c(names(llvec)[1:15], rep("", 15)), cex = 0.7, p
plot(llvec[-(1:30)], pch = pchvec[-(1:30)], col = colvec[-(1:30)], ylab = "likelihood", xlab = "ranl
axis(1, at = 1:length(llvec[-(1:30)]), labels = 31:length(llvec))
}

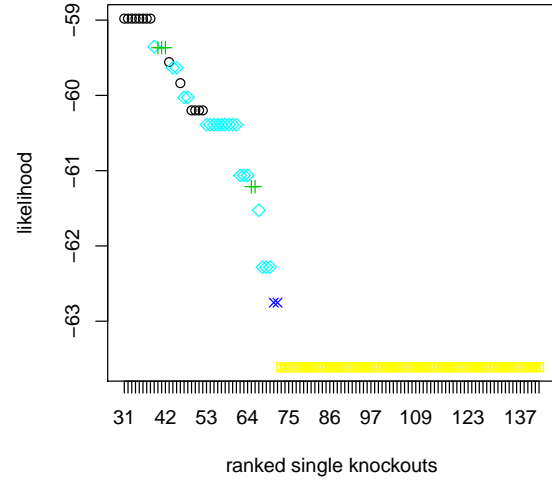
```



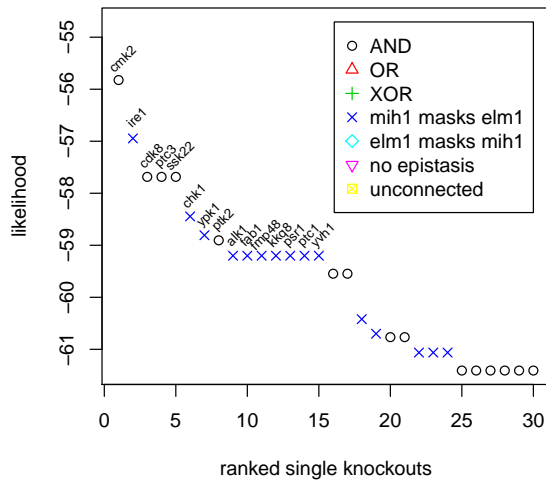
dun1 and pph3



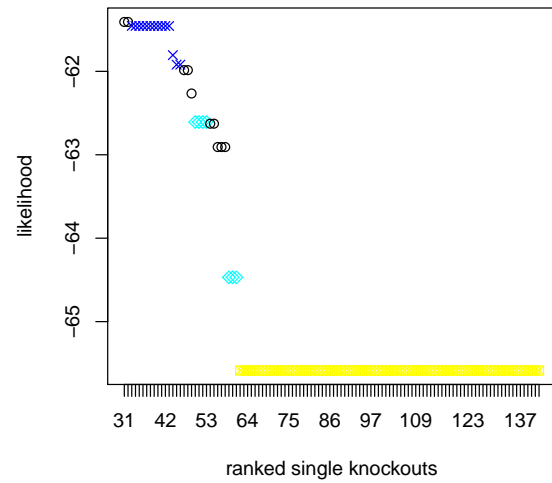
dun1 and pph3

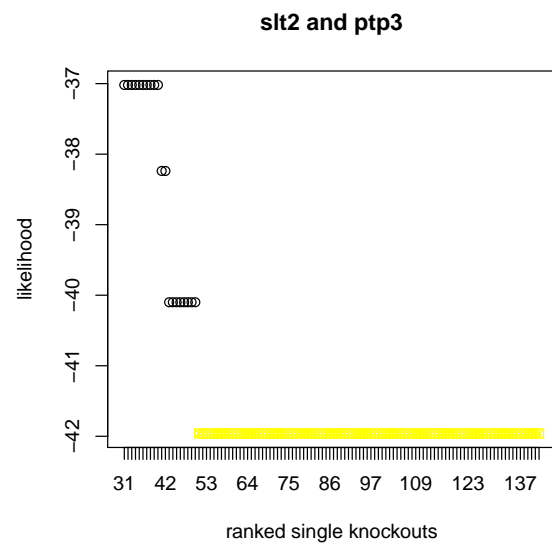
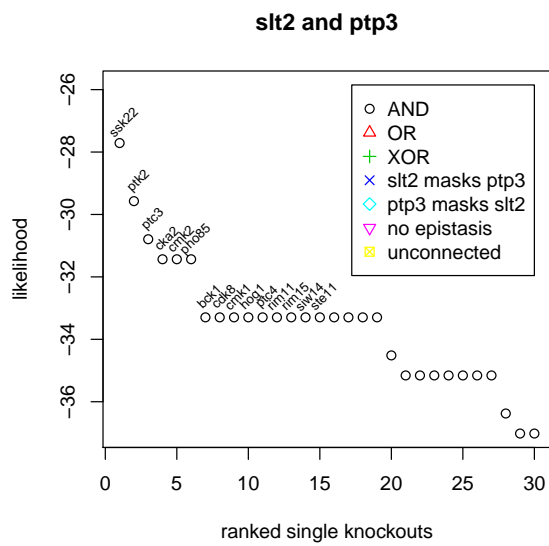
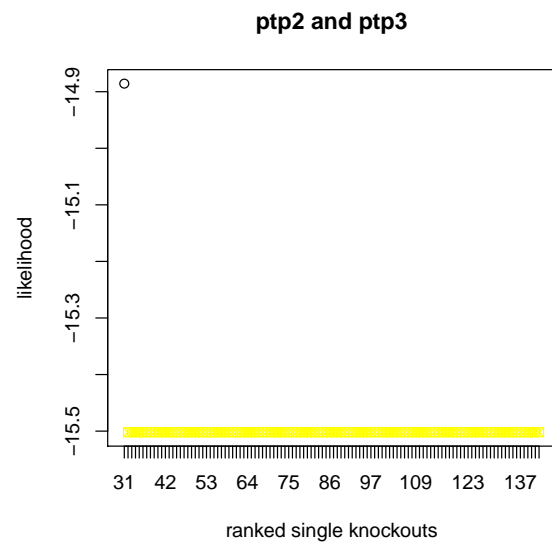
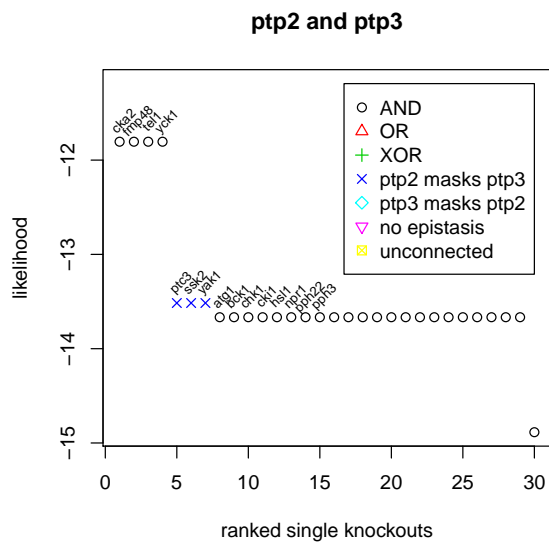


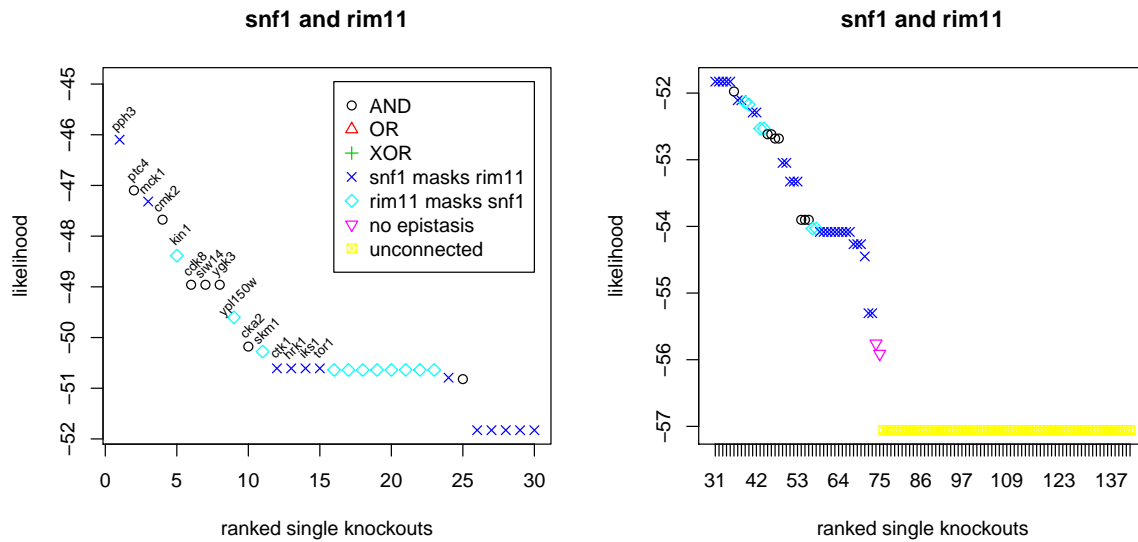
mih1 and elm1



mih1 and elm1







Sameith et al., 2015

```
data <- read.delim("http://www.holstegelab.nl/publications/GSTF_geneticinteractions/downloads/del_mutant")

data <- apply(data, c(1,2), as.character)

dataM <- data[-1, which(data[1, ] %in% "M")]

dataM <- apply(dataM, c(1,2), as.numeric)

dataP <- data[-1, which(data[1, ] %in% "p.value")]

dataP <- apply(dataP, c(1,2), as.numeric)

dataBin <- dataM

sig <- 0.05

cutoff <- 0.7

dataBin[which(dataP < sig & dataP > 0 & abs(dataM) >= cutoff)] <- 1

dataBin[which(dataP >= sig | dataP == 0 | abs(dataM) < cutoff)] <- 0

dataBin <- dataBin[-which(apply(dataBin, 1, max) == 0), ]

colnames(dataBin) <- gsub("\\.\\.\\.\\.\\.\\.", "\\.", colnames(dataBin))

## big screen:

doubles <- colnames(dataBin)[grep("\\.", colnames(dataBin))]

doubles.genes <- unique(unlist(strsplit(doubles, "\\.")))
```

```

singles <- colnames(dataBin)[-grep("\\.", colnames(dataBin))]

singles <- unique(sort(singles))

llmat <- logicmat <- matrix(0, length(singles), length(doubles))

rownames(llmat) <- rownames(logicmat) <- singles

colnames(llmat) <- colnames(logicmat) <- doubles

globalgenes <- which(apply(dataBin, 1, max) == 1)

## for (i in doubles[set]) {
##   print(i)
##   doubles.singles <- unlist(strsplit(i, "\\."))
##   egenes <- which(apply(dataBin[, which(colnames(dataBin) %in% c(i, doubles.singles))], 1, max) ==
##   for (j in singles) {
##     print(j)
##     if (j %in% doubles.singles) { next() }

##     dataTmp <- dataBin[, grep(paste(paste("^", c(i, j, doubles.singles), "$", sep = ""), collapse="|"))]

##     if (path %in% "fixed_set") {
##       dataTmp <- dataTmp[egenes, ]
##     }
##     if (path %in% "global") {
##       dataTmp <- dataTmp[globalgenes, ]
##     }
##     if (path %in% "") {
##       dataTmp <- dataTmp[which(apply(dataTmp, 1, max) == 1), ]
##     }

##     i1 <- which(singles %in% j)
##     i2 <- which(doubles %in% i)

##     if (!(is.null(dim(dataTmp)))) {

##       if (any(dataTmp[, j] != 0)) {

##         epires <- epiNEM(dataTmp, method = "exhaustive")

##         tmp <- epires$logics
##         if ("OR" %in% tmp) {
##           if (sum(epires$origModel[, j]) != 2) {
##             tmp <- "NOEPI"
##           } else {
##             if (all(tmp %in% "OR")) {
##               tmp <- "OR"
##             } else {
##               tmp <- tmp[which(!(tmp %in% "OR"))]
##             }
##           }
##         }
##       }
##     }

```

```
##             logicmat[i1, i2] <- tmp
##             llmat[i1, i2] <- epires$score

##         } else {

##             logicmat[i1, i2] <- "UNCON"
##             llmat[i1, i2] <- -Inf

##         }

##     } else {

##         logicmat[i1, i2] <- "UNCON"
##         llmat[i1, i2] <- -Inf

##     }

## }

## }
```

```
data(sameith_res)

llmat0 <- sameith$ll

logicmat0 <- sameith$logic

paperdoubles <- c(4, 9, 17)

for (i in 1:length(doubles)) {

  if (!(doubles[i] %in% c("ECM22.UPC2", "GLN3.GZF3"))) { next() }

  if (i %in% 8) { next() }

  logicvec <- logicmat0[, i]

  llvec <- llmat0[, i]

  logicvec <- logicvec[order(llvec, decreasing = T)]

  llvec <- llvec[order(llvec, decreasing = T)]

  parents <- unlist(strsplit(doubles[i], "\\."))

  pchvec <- numeric(length(llvec))

  pchvec[which(logicvec %in% "AND")] <- 1
  pchvec[which(logicvec %in% "OR")] <- 2
  pchvec[which(logicvec %in% "XOR")] <- 3
  pchvec[grepl(paste("^", parents[1], sep = ""), logicvec)] <- 4
  pchvec[grepl(paste("^", parents[2], sep = ""), logicvec)] <- 5
  pchvec[which(logicvec %in% "NOEPI")] <- 6

}
```

```

pchvec[which(logicvec %in% "UNCON")] <- 7

logicvec <- logicvec[-which(logicvec %in% "0")]
pchvec <- pchvec[-which(pchvec == 0)]
llvec <- llvec[-which(llvec == 0)]

colvec <- pchvec

llvec[which(is.infinite(llvec) == T)] <- Inf

llvec[which(is.infinite(llvec) == T)] <- min(llvec) - 100

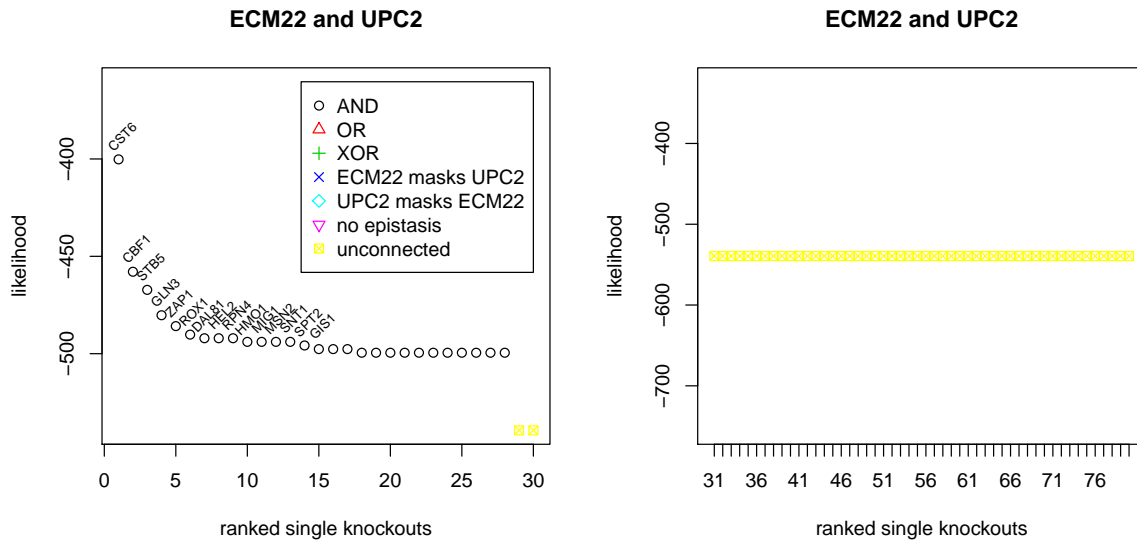
if (all(is.infinite(llvec) == T)) { llvec[1:length(llvec)] <- -1000 }

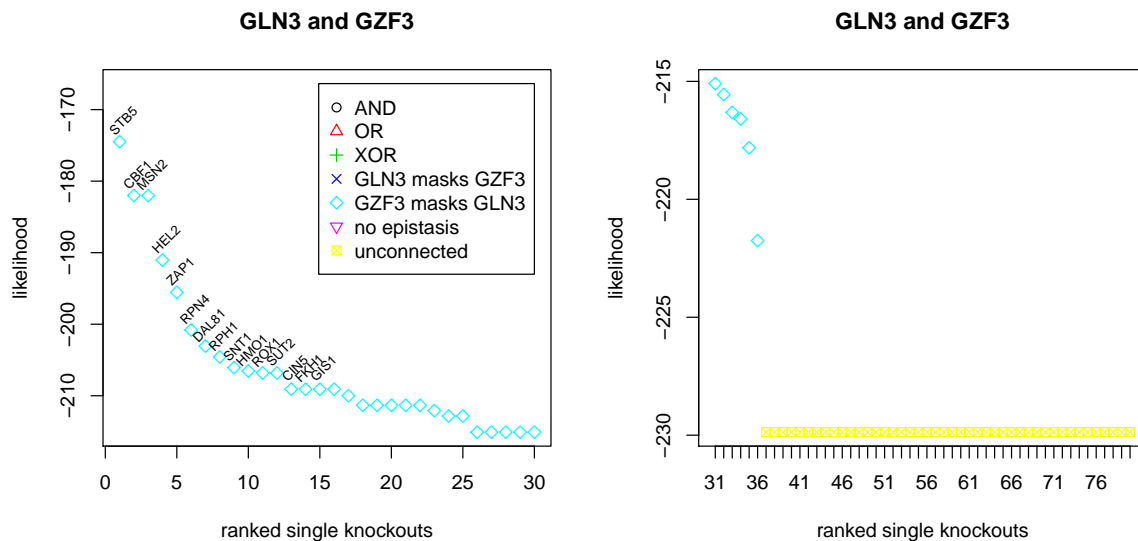
margin <- abs(max(llvec[1:30]) - min(llvec[1:30]))*0.2

llvec[which(llvec == min(llvec))] <- min(llvec) + 100 - margin

par(mfrow=c(1,2))
plot(llvec[1:30], pch = pchvec[1:30], col = colvec[1:30], ylab = "likelihood", xlab = "ranked single knockouts",
legend(30, max(llvec)+margin, legend = c("AND", "OR", "XOR", paste(parents[1], " masks ", parents[2])),
text((1:30)+0.5, llvec[1:30]+(margin*0.1), labels = c(names(llvec)[1:15], rep("", 15)), cex = 0.7, p
plot(llvec[-(1:30)], pch = pchvec[-(1:30)], col = colvec[-(1:30)], ylab = "likelihood", xlab = "ranked single knockouts",
axis(1, at = 1:length(llvec[-(1:30)]), labels = 31:length(llvec))
}

```





```
sessionInfo()

## R version 3.3.1 (2016-06-21)
## Platform: x86_64-apple-darwin13.4.0 (64-bit)
## Running under: OS X 10.11.5 (El Capitan)
##
## locale:
## [1] C/UTF-8/C/C/C/C
##
## attached base packages:
## [1] grid      parallel  stats      graphics  grDevices  utils      datasets  methods
## [9] base
##
## other attached packages:
## [1] pcalg_2.4-3          minet_3.30.0          bnem_1.0              latticeExtra_0.6-28
## [5] RColorBrewer_1.1-2   lattice_0.20-34       snowfall_1.84-6.1     snow_0.4-1
## [9] matrixStats_0.50.2   nem_2.46.0            CellNOptR_1.18.0      XML_3.98-1.4
## [13] Rgraphviz_2.16.0     RCurl_1.95-4.8        bitops_1.0-6          ggplot2_2.1.0
## [17] hash_2.2.6           RBGL_1.48.1           graph_1.50.0          BiocGenerics_0.18.0
## [21] epiNEM_1.0           knitr_1.14            igraph_1.0.1          gtools_3.5.0
## [25] e1071_1.6-7          BoolNet_2.1.1         devtools_1.12.0
##
## loaded via a namespace (and not attached):
## [1] statmod_1.4.26       colorspace_1.2-6      stats4_3.3.1          fastICA_1.2-0
## [5] gmp_0.5-12           withr_1.0.2           plyr_1.8.4            robustbase_0.92-6
## [9] stringr_1.1.0        munsell_0.4.3         gtable_0.2.0          bdsmatrix_1.3-2
## [13] memoise_1.0.0        evaluate_0.9          ggm_2.3              curl_2.1
## [17] class_7.3-14         highr_0.6             DEoptimR_1.0-6        Rcpp_0.12.7
## [21] corpcor_1.6.8        scales_0.4.0          formatR_1.4           limma_3.28.21
## [25] plotrix_3.6-3        abind_1.4-5           digest_0.6.10         stringi_1.1.1
## [29] clue_0.3-51          tools_3.3.1           magrittr_1.5          cluster_2.0.4
## [33] httr_1.2.1           R6_2.1.3             boot_1.3-18          sfsmisc_1.1-0
## [37] git2r_0.15.0
```

References:

