Epistatic Nested Effects Models
Inferring mixed epistatis from indirect measurements of knockout screens.

Madeline Diekmann & Martin Pirkl

November 22, 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")

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], rand

##         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 =

##             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$single]) {
##                sifMatrix <- rbind(sifMatrix, c("S", "1", k))#, c("S", "-1", k)) # bnem can set a pri
##                 for (l in LETTERS[1:random$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,
##                         CNOlist=CNOlist,
##                         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(CNOlist, t(simulateStatesRecursive(CNOlist, 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))
```

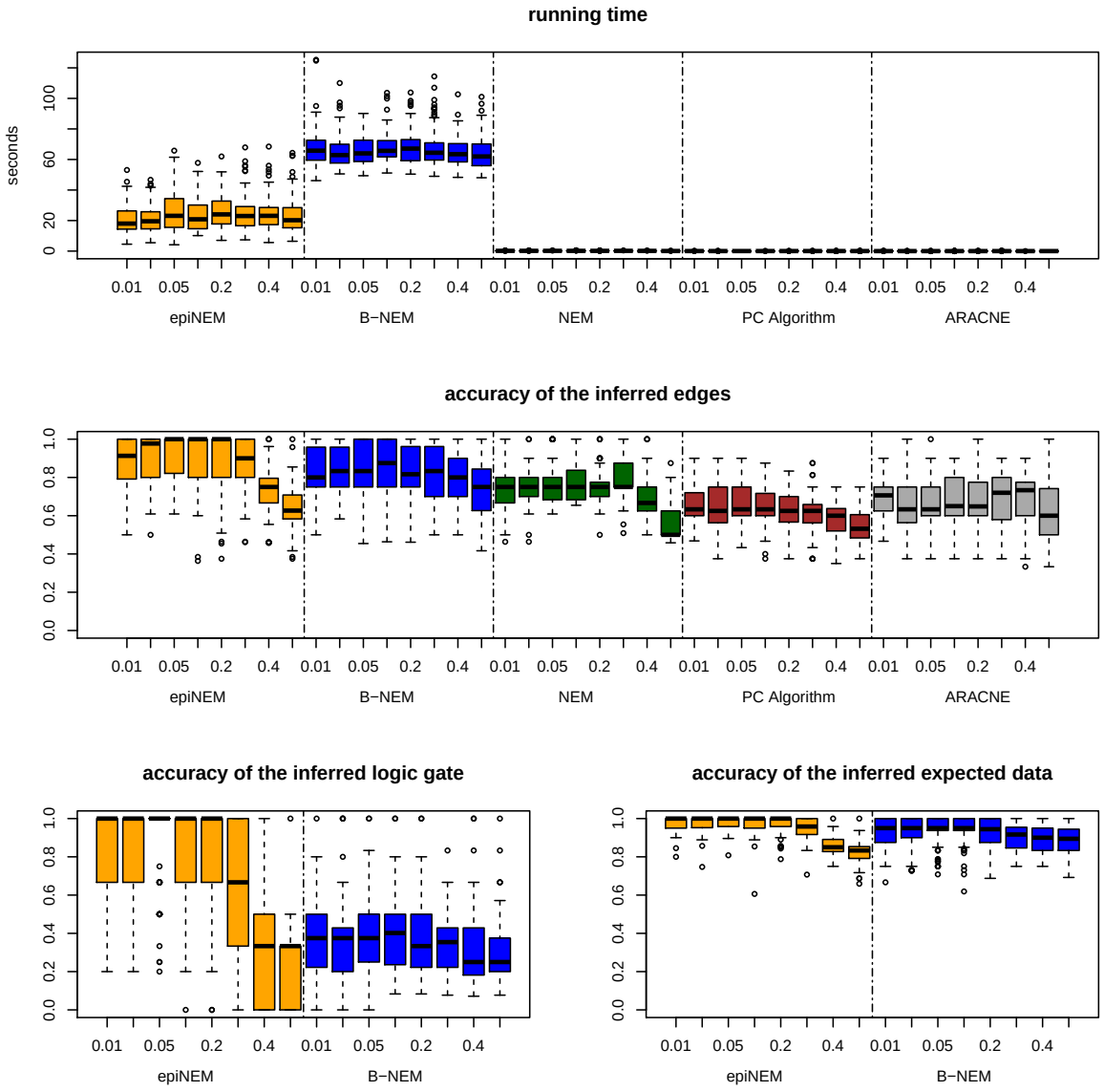
```
##             if (length(grep("S", rownames(gtn2))) > 0) {
##                 gtn2 <- gtn2[-grep("S", rownames(gtn2)), -grep("S", colnames(gtn2))]
##             }
##             gtn2 <- gtn2[order(rownames(gtn2)), order(colnames(gtn2))]
##             res <- abs(dnf2adj(bga$graph))
##             if (length(grep("S", rownames(res))) > 0) {
##                 res <- as.matrix(res[-grep("S", rownames(res)), -grep("S", colnames(res))])
##             }
##             if (dim(res)[1] == 1) {
##                 colnames(res) <- rownames(res) <- gsub(".*=", "", bga$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, nr
##                 colnames(res2)[(ncol(res)+1):ncol(res2)] <- colnames(gtn2)[which(!(colnames(gtn2) %i
##                 rownames(res2)[(nrow(res)+1):nrow(res2)] <- rownames(gtn2)[which(!(rownames(gtn2) %i
##                 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$origModel
##             gtnadj <- gtnadj[order(apply(gtnadj, 1, sum), decreasing = T), order(apply(gtnadj, 2, su
##             gtnadj[lower.tri(gtnadj)] <- gtnadj[upper.tri(gtnadj)]
##             gtnadj <- gtnadj[order(rownames(gtnadj)), order(colnames(gtnadj))]
##             eadj <- topology$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(re
##             }
```

```
##              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 = gaussCItest, ## 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)

colvec <- c(rep("orange", length(noiselvls)), rep("blue", length(noiselvls)), rep("darkgreen", length(no

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.fram

colnames(timeframe) <- rep(noiselvls, 5) # c(paste(rep("epi", length(noiselvls)), noiselvls, sep = "_"),

boxplot(timeframe, col = colvec, main = "running time", ylab = "seconds")

abline(v=(1:(length(do)-1)*length(noiselvls) + 0.5), col = "black", lty = 6)

axis(1, c(3, 11, 19, 28, 36)+1, c("epiNEM", "B-NEM", "NEM", "PC Algorithm", "ARACNE"), tick = F, pos = -

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

colnames(accframe2) <- rep(noiselvls, 5) # c(paste(rep("E", length(noiselvls)), noiselvls, sep = "_"), f

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

```r
abline(v=(1:(length(do)-1)*length(noiselvls) + 0.5), col = "black", lty = 6)
axis(1, c(3, 11, 19, 28, 36)+1, c("epiNEM", "B-NEM", "NEM", "PC Algorithm", "ARACNE"), tick = F, pos = -
## logical nems:
colvec2 <- c(rep("orange", length(noiselvls)), rep("blue", length(noiselvls)))
logicsframe <- as.data.frame(cbind(data.frame(epiNEM = logics[1,,]), data.frame(BNEM = logics[2,,])))
colnames(logicsframe) <- rep(noiselvls, 2) # c(paste(rep("E", length(noiselvls)), noiselvls, sep = "_"),
boxplot(logicsframe, col = colvec2, main = "accuracy of the inferred logic gate", ylim = c(0,1))
abline(v=length(noiselvls)+0.5, col = "black", lty = 6)
axis(1, c(3, 11, 19, 28, 36)+1, c("epiNEM", "B-NEM", "NEM", "PC Algorithm", "ARACNE"), tick = F, pos = -
accframe <- as.data.frame(cbind(data.frame(epiNEM = acc[1,,]), data.frame(BNEM = acc[2,,])))
colnames(accframe) <- rep(noiselvls, 2) # c(paste(rep("E", length(noiselvls)), noiselvls, sep = "_"), pa
boxplot(accframe, col = colvec2, main = "accuracy of the inferred expected data", ylim = c(0,1)) # what
abline(v=length(noiselvls)+0.5, col = "black", lty = 6)
axis(1, c(3, 11, 19, 28, 36)+1, c("epiNEM", "B-NEM", "NEM", "PC Algorithm", "ARACNE"), tick = F, pos = -
```



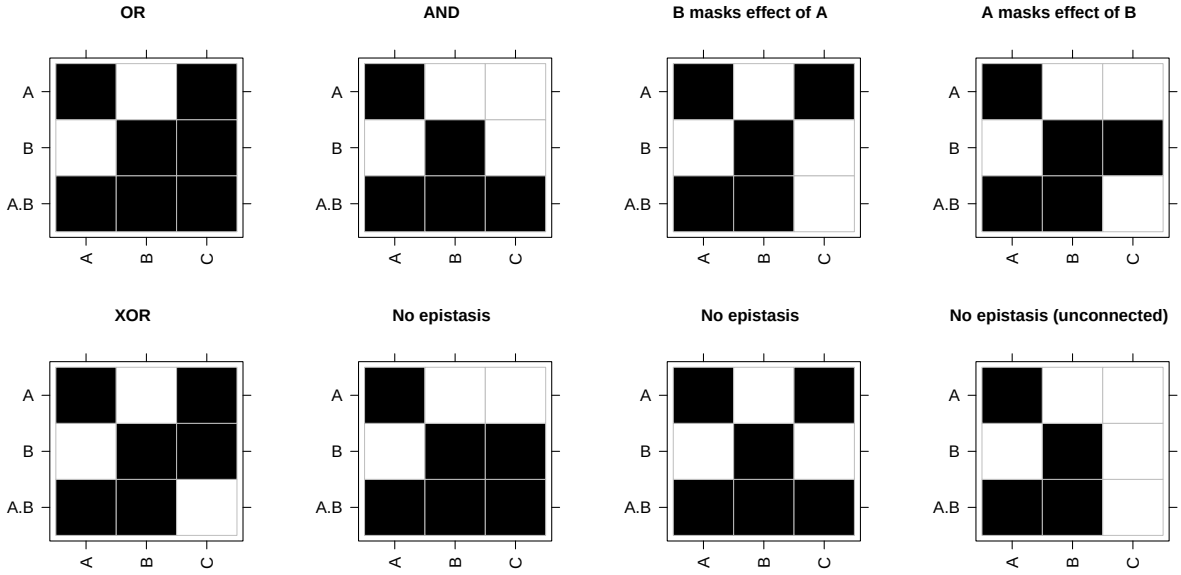
Yeast knockout screens

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

```
heatmapOP(matrix(c(1,-1,1,-1,1,1, 1, -1, 1), 3, 3, dimnames = list(c("A", "B", "A.B"), LETTERS[1:3])), C
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 <- read.delim("http://www.holstegelab.nl/publications/sv/signaling_redundancy/downloads/DataS1.txt"

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 <- log2(1.7)

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), ]

genelist <- toupper(c('hsl1', 'cla4', 'gin4', 'swe1', 'hsl1.cla4'))
```

```r
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) ==
##     for (j in singles) {
##         print(j)
##         if (j %in% doubles.singles) { next() }

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

##         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
```

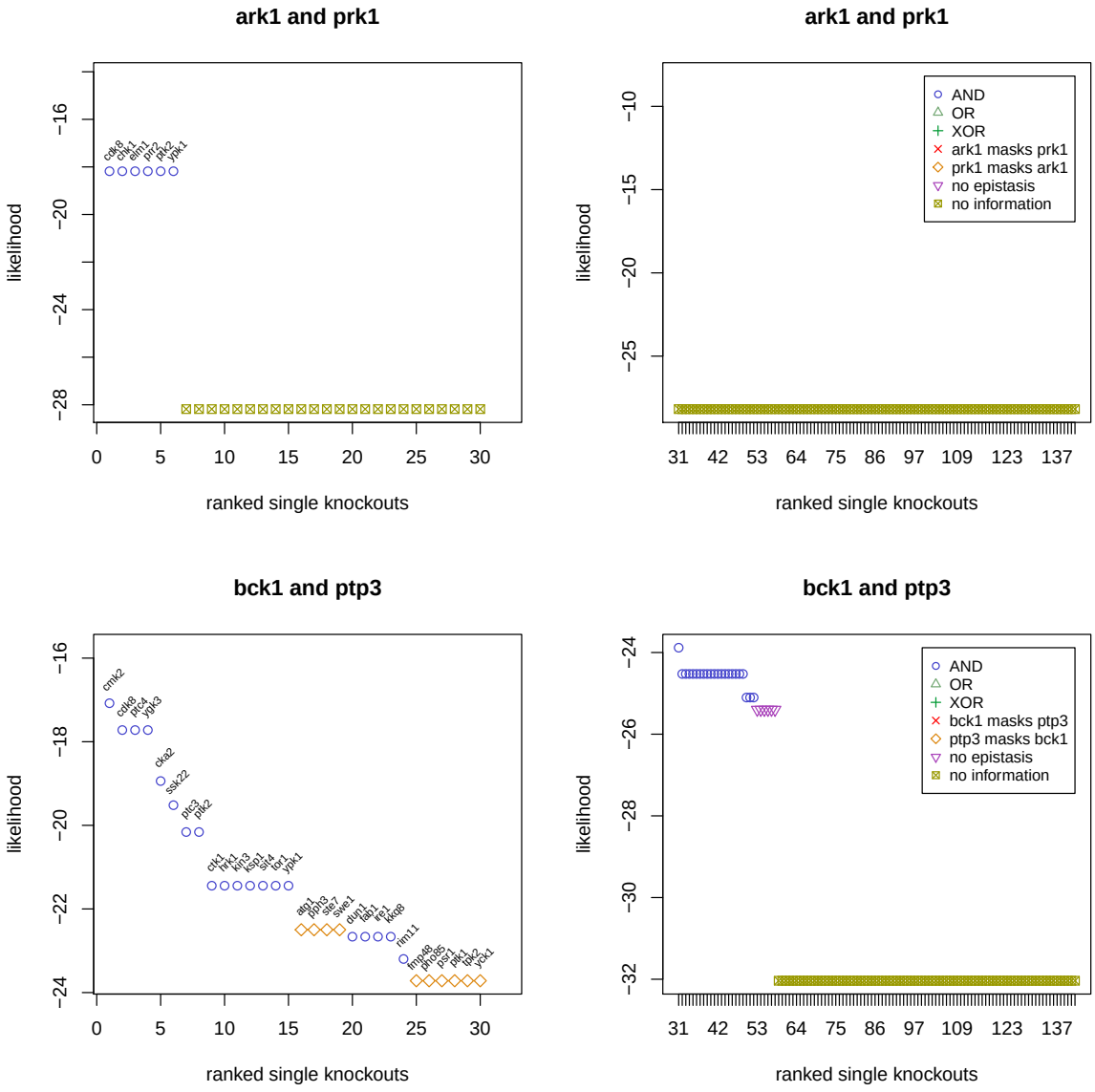
```
##                 if ("OR" %in% tmp) {
##                     if (sum(epires$origModel[, j]) != 2) {
##                         tmp <- "NOEPI"
##                     } else {
##                         if (all(tmp %in% "OR")) {
##                             tmp <- "OR"
##                      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

##         }

##     }

## }
```

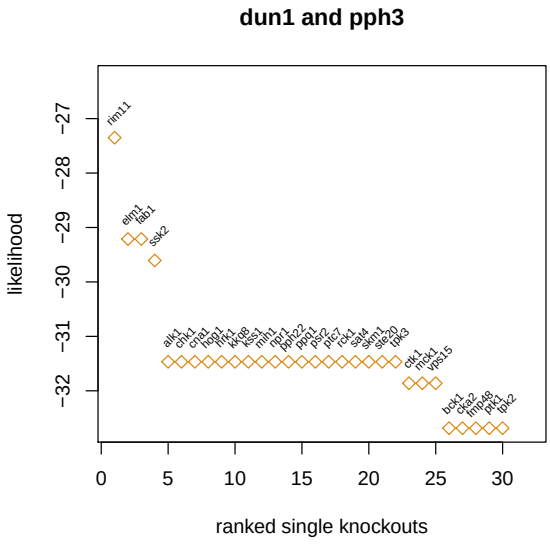
Plot results.

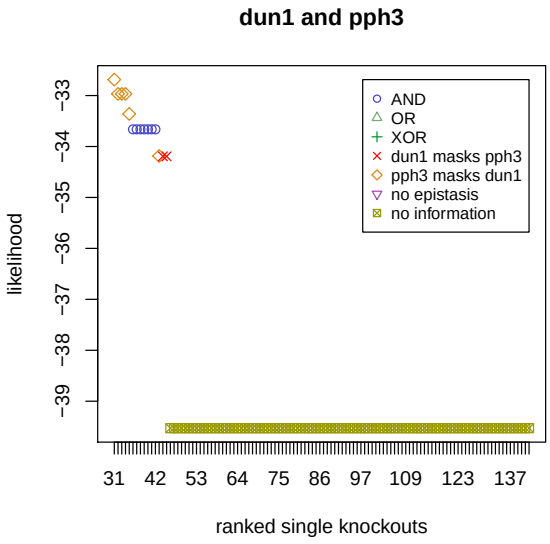
```
palette(c("#4444cc", "#77aa77", "#009933", "#ff0000", "#dd8811", "#aa44bb", "#999900"))

data(wageningen_res)

llmat0 <- wageningen$ll

logicmat0 <- wageningen$logic

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)]
```

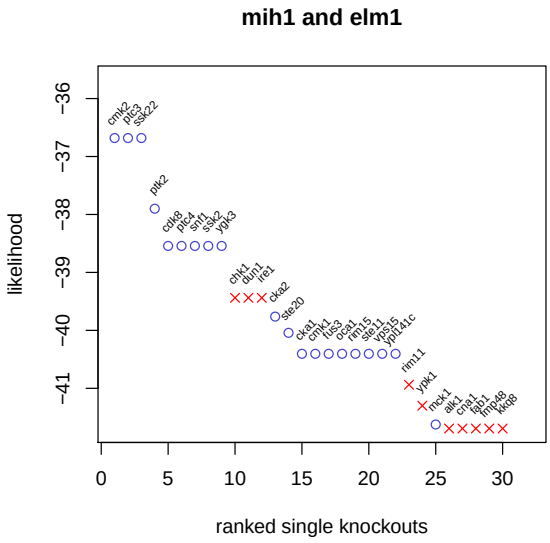
```
    llvec <- llvec[order(llvec, decreasing = T)]

    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[grep(paste("^", parents[1], sep = ""), logicvec)] <- 4
    pchvec[grep(paste("^", parents[2], sep = ""), logicvec)] <- 5
    pchvec[which(logicvec %in% "NOEPI")] <- 6
    pchvec[which(logicvec %in% c("NOINFO", "NOINF"))] <- 7

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

    colvec <- pchvec

    if (all(is.infinite(llvec) == T)) {

        llvec[1:length(llvec)] <- -1000

        margin <- 100

        donames <- 30

    } else {

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

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

        margin <- abs(max(llvec[1:30], na.rm = T) - min(llvec[1:30], na.rm = T))

        offset <- 0.075

        if (margin == 0) { margin <- 10; offset <- 0.0375 }

        donames <- 30 - sum(is.na(llvec[1:30]) == T)

        if (any(is.na(llvec[1:30]) == T)) { margin2 <- margin*2 } else { margin2 <- margin }

        llvec[which(is.na(llvec) == T)] <- min(llvec, na.rm = T) - margin

        margin <- margin2

    }

    if (all(llvec[-(1:30)] - min(llvec[-(1:30)]) == 0)) {

        p2max <- max(llvec[-(1:30)]) + margin
```

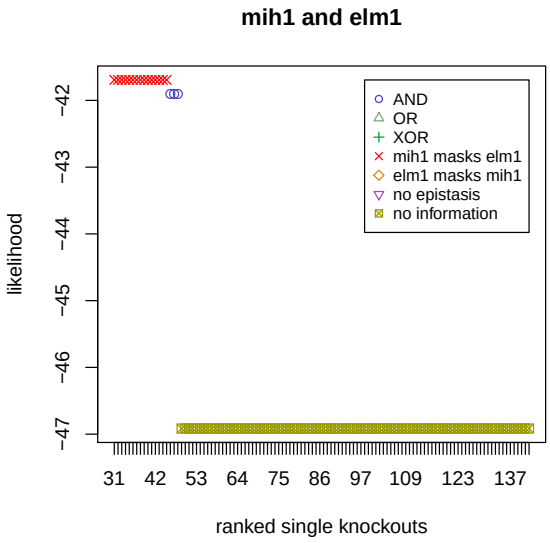
```
    } else {

        p2max <- max(llvec[-(1:30)])

    }
    par = par(mfrow=c(1,2))
    plot = plot(llvec[1:30], pch = pchvec[1:30], col = colvec[1:30], ylab = "likelihood", xlab = "ranked
    text = text((1:30)+0.5, llvec[1:30]+(margin*offset), labels = c(names(llvec)[1:donames], rep("", 30
    plot2 = plot(llvec[-(1:30)], pch = pchvec[-(1:30)], col = colvec[-(1:30)], ylab = "likelihood", xlab
    legend = legend(length(llvec[-(1:30)]), p2max,
                    legend = c("AND", "OR", "XOR", paste(parents[1], " masks ", parents[2], sep = ""), p
    axis = axis(1, at = 1:length(llvec[-(1:30)]), labels = 31:length(llvec))

}
```

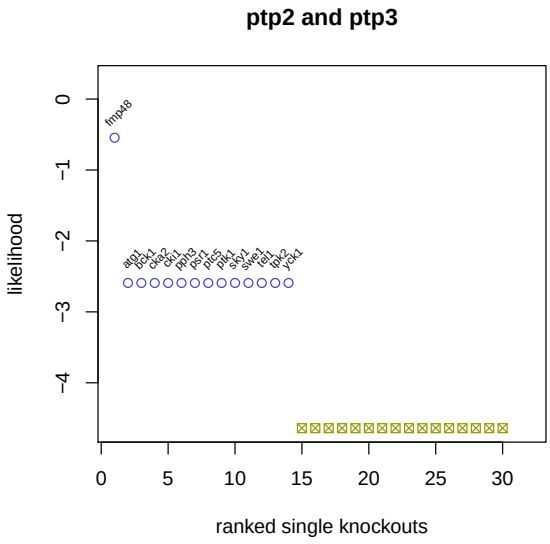


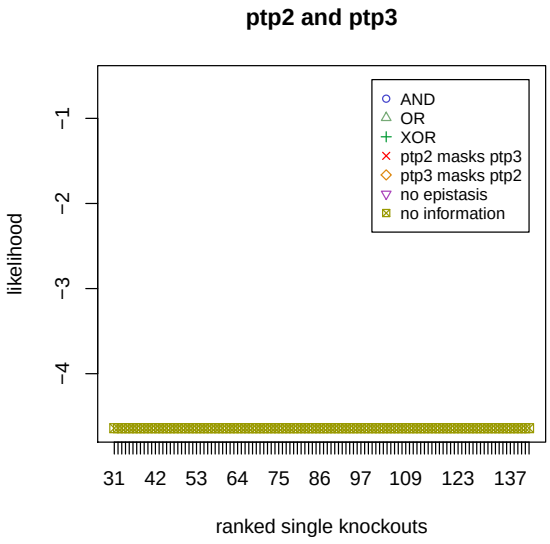

dun1 and pph3
likelihood
ranked single knockouts
rim11
elm1
tab1
ssk2
alk1
chk1
cna1
hog1
fin1
kkq8
ksp1
mih1
mpt1
pph22
ppz1
psr2
ptc7
rck1
sgv1
slm1
ste20
tpk3
cnk1
nck1
vps15
bck1
cka2
hmp48
pkc1
tpk2

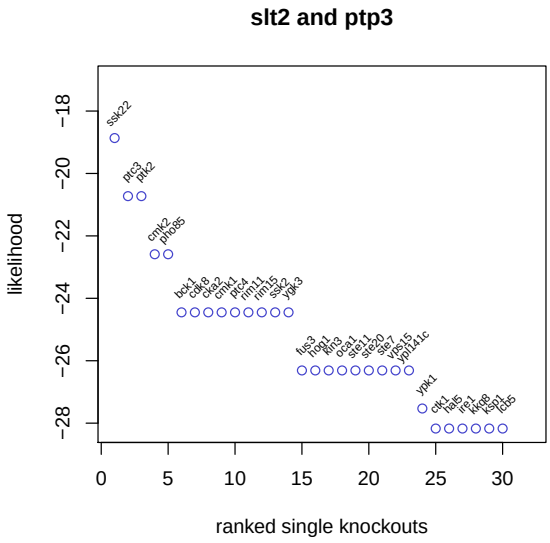

dun1 and pph3
likelihood
ranked single knockouts
AND
OR
XOR
dun1 masks pph3
pph3 masks dun1
no epistasis
no information

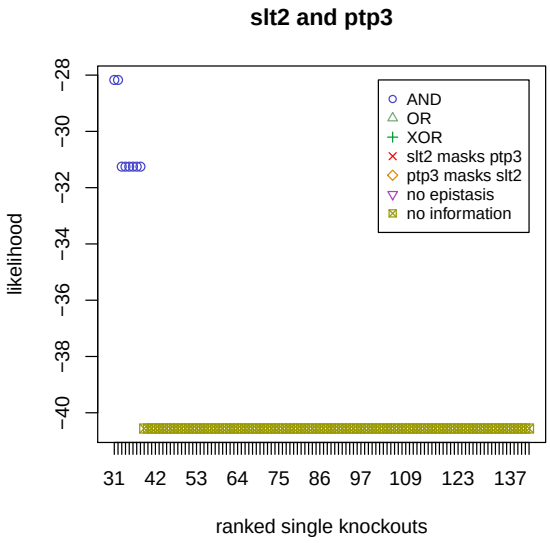

mih1 and elm1
likelihood
ranked single knockouts
cmk2
ptc3
ssk22
ptk2
cdc8
pte4
snf1
ssk2
ypk3
chk1
dun1
ire1
cka2
ste20
cke1
cmk1
fus3
ocr1
rim15
ste11
vps15
ypl141c
rim11
ypk1
nek1
alk1
cna1
tab1
hmp48
kkq8

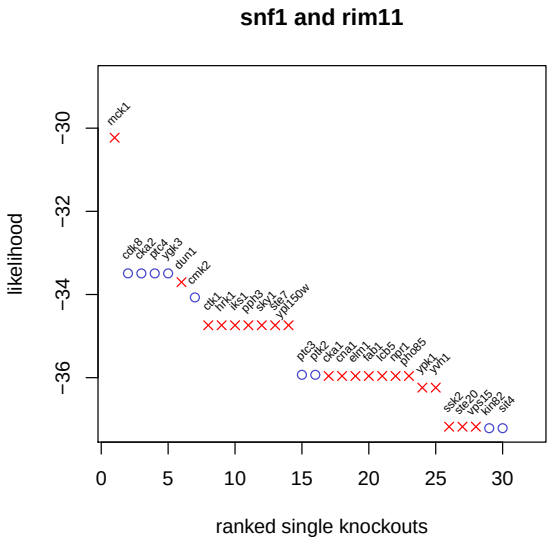

mih1 and elm1
likelihood
ranked single knockouts
AND
OR
XOR
mih1 masks elm1
elm1 masks mih1
no epistasis
no information

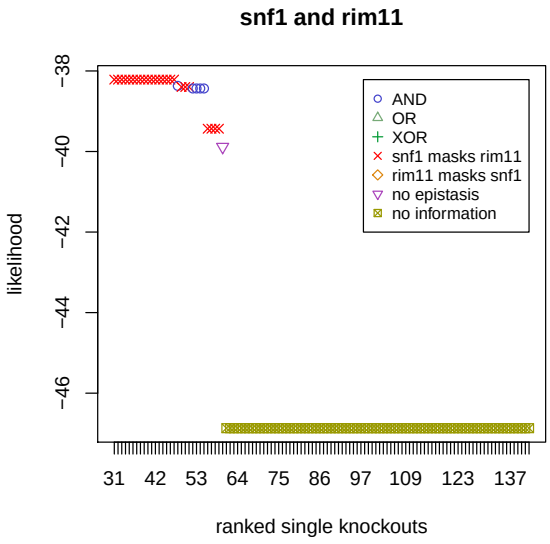

ptp2 and ptp3
likelihood
ranked single knockouts
fmp48
atg1 bck1 cka2 cka1 pp13 psr1 ptc5 ptk1 sky1 swe1 tel1 tpk2 yck1


ptp2 and ptp3
likelihood
ranked single knockouts
AND
OR
XOR
ptp2 masks ptp3
ptp3 masks ptp2
no epistasis
no information


slt2 and ptp3
likelihood
ranked single knockouts
ssk22
ptc3 ptk2
cmk2 pho85
bck1 cdc8 cka2 cmk1 ptc4 rim11 rim15 ssk2 ypk3
fus3 hog1 kin3 oca1 ste11 ste20 ste7 ybl015 ypl141c
ypk1
ckl1 hal5 ire1 kkq8 ksp1 tda5

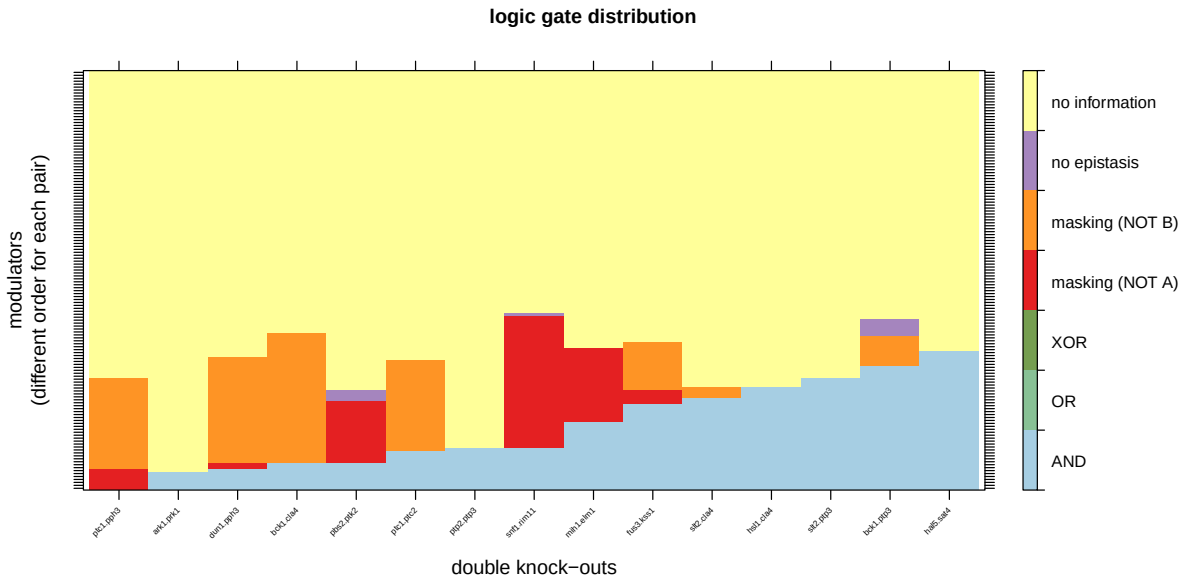

slt2 and ptp3
likelihood
ranked single knockouts
AND
OR
XOR
slt2 masks ptp3
ptp3 masks slt2
no epistasis
no information






```
distmat <- wageningen$logicmat

distmat[which(distmat %in% "AND")] <- 1
distmat[which(distmat %in% "OR")] <- 2
distmat[which(distmat %in% "XOR")] <- 3
distmat[which(distmat %in% "NOEPI")] <- 6
distmat[which(distmat %in% c("NOINFO", "NOINF"))] <- 7

for (i in 1:ncol(distmat)) {

    genes <- unlist(strsplit(colnames(distmat)[i], "\\."))

    distmat[which(distmat[, i] %in% paste(genes[1], " masks the effect of ", genes[2], sep = "")), i] <-


    distmat[which(distmat[, i] %in% paste(genes[2], " masks the effect of ", genes[1], sep = "")), i] <-

}

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

for (i in 1:ncol(distmat)) {
    distmat[, i] <- rev(sort(distmat[, i]))
}

rownames(distmat) <- 1:nrow(distmat)

distmat <- distmat[-which(apply(distmat, 1, sum) == 0), ]

distmat <- distmat[, -which(apply(distmat, 2, max) == 0 | apply(distmat, 2, min) == 7)]

library(bnem)

y <- distmat
```

```
distmat <- distmat[, order(apply(distmat, 2, function(x) { return(sum(x == 1)) }))]

y[which(y == 5)] <- 4

heatmapOP(distmat, Colv = F, Rowv = F, main = "logic gate distribution", sub = "", col = "Paired", breal
```



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))
```

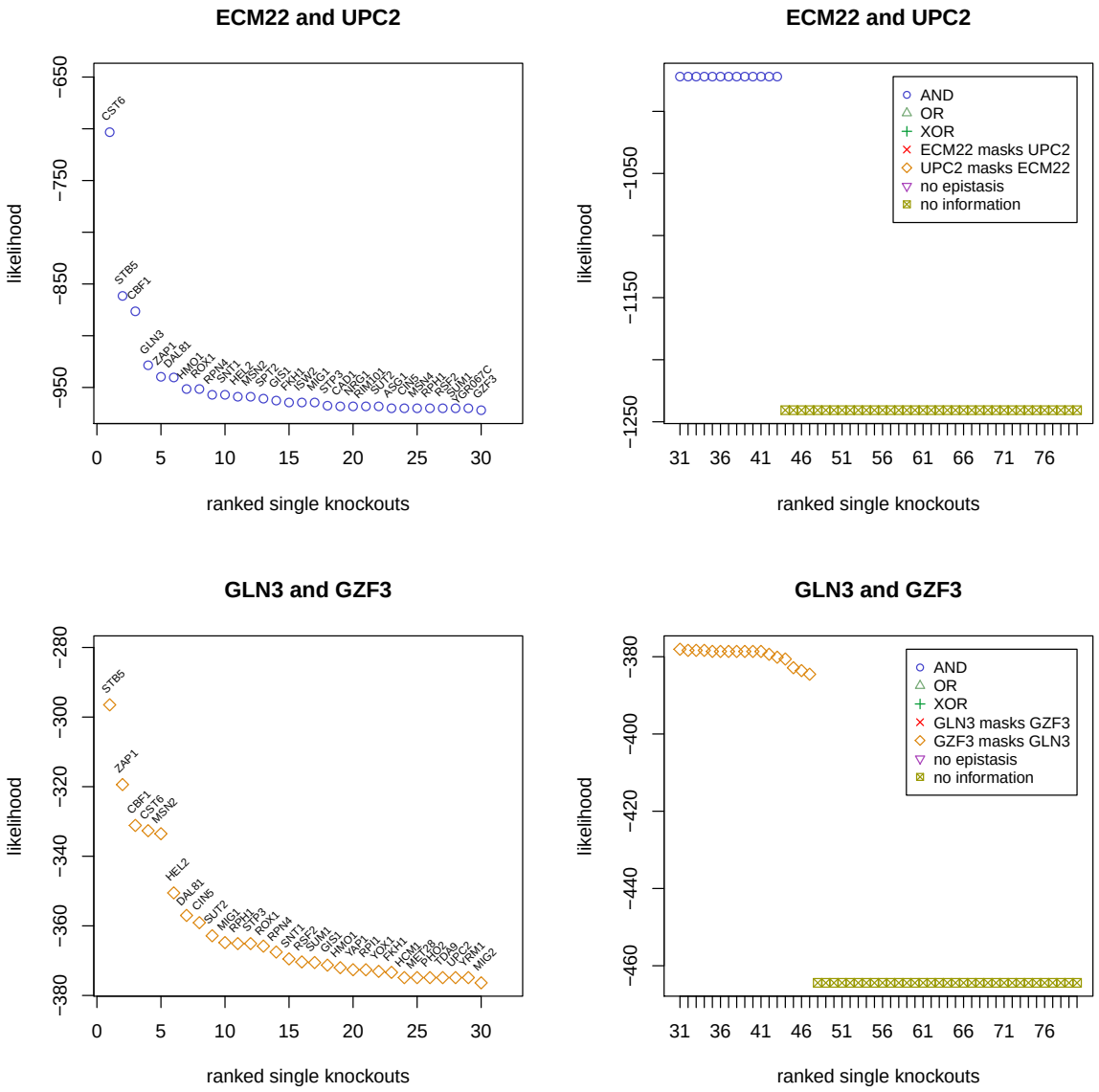
```
## big screen:

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 = ""), collaps

##         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() }

  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
```

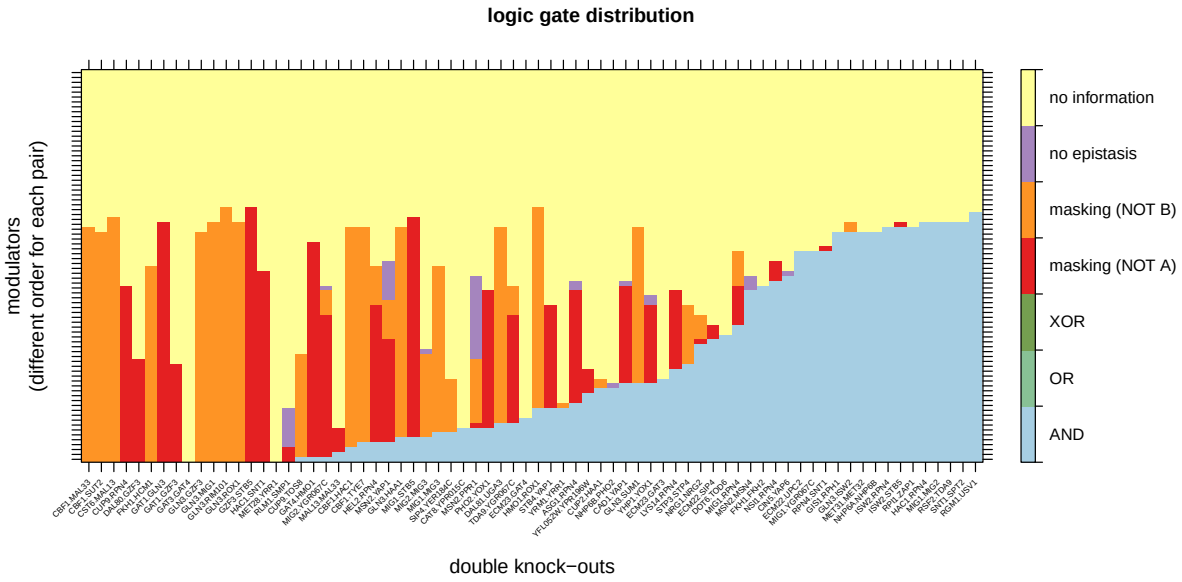
```
    pchvec[which(logicvec %in% "OR")] <- 2
    pchvec[which(logicvec %in% "XOR")] <- 3
    pchvec[grep(paste("^", parents[1], sep = ""), logicvec)] <- 4
    pchvec[grep(paste("^", parents[2], sep = ""), logicvec)] <- 5
    pchvec[which(logicvec %in% "NOEPI")] <- 6
    pchvec[which(logicvec %in% c("NOINFO", "NOINF"))] <- 7

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

    colvec <- pchvec

    if (all(is.infinite(llvec) == T)) {

        llvec[1:length(llvec)] <- -1000

        margin <- 100

        donames <- 30

    } else {

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

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

        margin <- abs(max(llvec[1:30], na.rm = T) - min(llvec[1:30], na.rm = T))

        if (margin == 0) { margin <- 10 }

        donames <- 30 - sum(is.na(llvec[1:30]) == T)

        if (any(is.na(llvec[1:30]) == T)) { margin2 <- margin*2 } else { margin2 <- margin }

        llvec[which(is.na(llvec) == T)] <- min(llvec, na.rm = T) - margin

        margin <- margin2

    }

    if (all(llvec[-(1:30)] - min(llvec[-(1:30)]) == 0)) {

        p2max <- max(llvec[-(1:30)]) + margin

    } else {

        p2max <- max(llvec[-(1:30)])

    }

    par = par(mfrow=c(1,2))
    plot = plot(llvec[1:30], pch = pchvec[1:30], col = colvec[1:30], ylab = "likelihood", xlab = "ranke
```

```
    text = text((1:30)+0.5, llvec[1:30]+(margin*0.075), labels = c(names(llvec)[1:donames], rep("", 30 -
    plot2 = plot(llvec[-(1:30)], pch = pchvec[-(1:30)], col = colvec[-(1:30)], ylab = "likelihood", xlab
    legend = legend(length(llvec[-(1:30)]), p2max,
                    legend = c("AND", "OR", "XOR", paste(parents[1], " masks ", parents[2], sep = ""), p
    axis = axis(1, at = 1:length(llvec[-(1:30)]), labels = 31:length(llvec))

}
```




```
distmat <- sameith$logic

distmat[which(distmat %in% "AND")] <- 1
distmat[which(distmat %in% "OR")] <- 2
distmat[which(distmat %in% "XOR")] <- 3
distmat[which(distmat %in% "NOEPI")] <- 6
distmat[which(distmat %in% c("NOINFO", "NOINF"))] <- 7
```

```
for (i in 1:ncol(distmat)) {

    genes <- unlist(strsplit(colnames(distmat)[i], "\\."))

    distmat[which(distmat[, i] %in% paste(genes[1], " masks the effect of ", genes[2], sep = "")), i] <-


    distmat[which(distmat[, i] %in% paste(genes[2], " masks the effect of ", genes[1], sep = "")), i] <-

}

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

for (i in 1:ncol(distmat)) {
    distmat[, i] <- rev(sort(distmat[, i]))
}

rownames(distmat) <- 1:nrow(distmat)

distmat <- distmat[-which(apply(distmat, 1, sum) == 0), ]

library(bnem)

y <- distmat

distmat <- distmat[, order(apply(distmat, 2, function(x) { return(sum(x == 1)) }))]

y[which(y == 5)] <- 4

heatmapOP(distmat, Colv = F, Rowv = F, main = "logic gate distribution", sub = "", col = "Paired", breal
```




```
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_3.32.0        bnem_0.99.0         latticeExtra_0.6-28
##  [5] RColorBrewer_1.1-2  lattice_0.20-34     snowfall_1.84-6.1   snow_0.4-2
##  [9] matrixStats_0.51.0  nem_2.48.0          CellNOptR_1.20.0    XML_3.98-1.5
## [13] Rgraphviz_2.18.0    RCurl_1.95-4.8      bitops_1.0-6        ggplot2_2.2.0
## [17] hash_2.2.6          RBGL_1.50.0         graph_1.52.0        BiocGenerics_0.20.0
## [21] epiNEM_0.99.0       igraph_1.0.1        gtools_3.5.0        e1071_1.6-7
## [25] BoolNet_2.1.1       knitr_1.15          devtools_1.12.0
##
## loaded via a namespace (and not attached):
##  [1] statmod_1.4.26       colorspace_1.3-0     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_0.10        ggm_2.3              BiocInstaller_1.24.0
## [17] class_7.3-14         highr_0.6            DEoptimR_1.0-6       Rcpp_0.12.8
## [21] corpcor_1.6.8        scales_0.4.1         limma_3.30.4         plotrix_3.6-3
## [25] abind_1.4-5          digest_0.6.10        stringi_1.1.2        clue_0.3-51
## [29] tools_3.3.1          magrittr_1.5         lazyeval_0.2.0       tibble_1.2
## [33] cluster_2.0.5        assertthat_0.1       boot_1.3-18          sfsmisc_1.1-0
```

References: