ciber

Lan-lab

6/19/2024

Structure learning

Load packages and data

```
library(Seurat) # If processing single cell data, Seurat is necessary.
library(foreach)
library(tidyverse)
library(bnlearn) # Core package
library(Matrix) # Core package
library(space) # Core package
library(infotheo)

# install.packages("ciber.tar.gz", repos = NULL, type = "source")
library(ciber)

data("HMMA")
head(HMMA)
```

```
##                     HSC  MPPa     MPPb     sCMP   MEP      GMP      MkP   pCFU-E
## 0610005C13Rik 4.400000 4.420 4.170000 4.180000 4.250 4.370000 4.890000 4.320000
## 0610006L08Rik 3.810000 3.950 4.030000 3.960000 4.390 4.460000 4.850000 4.540000
## 0610007C21Rik 8.950000 6.340 7.650000 6.550000 8.040 8.630000 7.030000 7.390000
## 0610007L01Rik 7.693333 8.320 8.423333 8.446667 9.300 9.876667 7.453333 8.023333
## 0610007P08Rik 5.430000 5.742 6.088000 5.870000 6.166 5.790000 5.496000 5.682000
## 0610007P14Rik 7.930000 7.700 7.760000 7.160000 8.820 8.820000 7.530000 7.540000
##                  CLP      BLP  DN1
## 0610005C13Rik 4.15 4.510000 4.19
## 0610006L08Rik 4.38 4.330000 4.28
## 0610007C21Rik 9.03 8.740000 8.68
## 0610007L01Rik 8.35 8.213333 7.69
## 0610007P08Rik 5.89 5.896000 5.63
## 0610007P14Rik 8.47 8.280000 7.85
```

```
dim(HMMA)
```

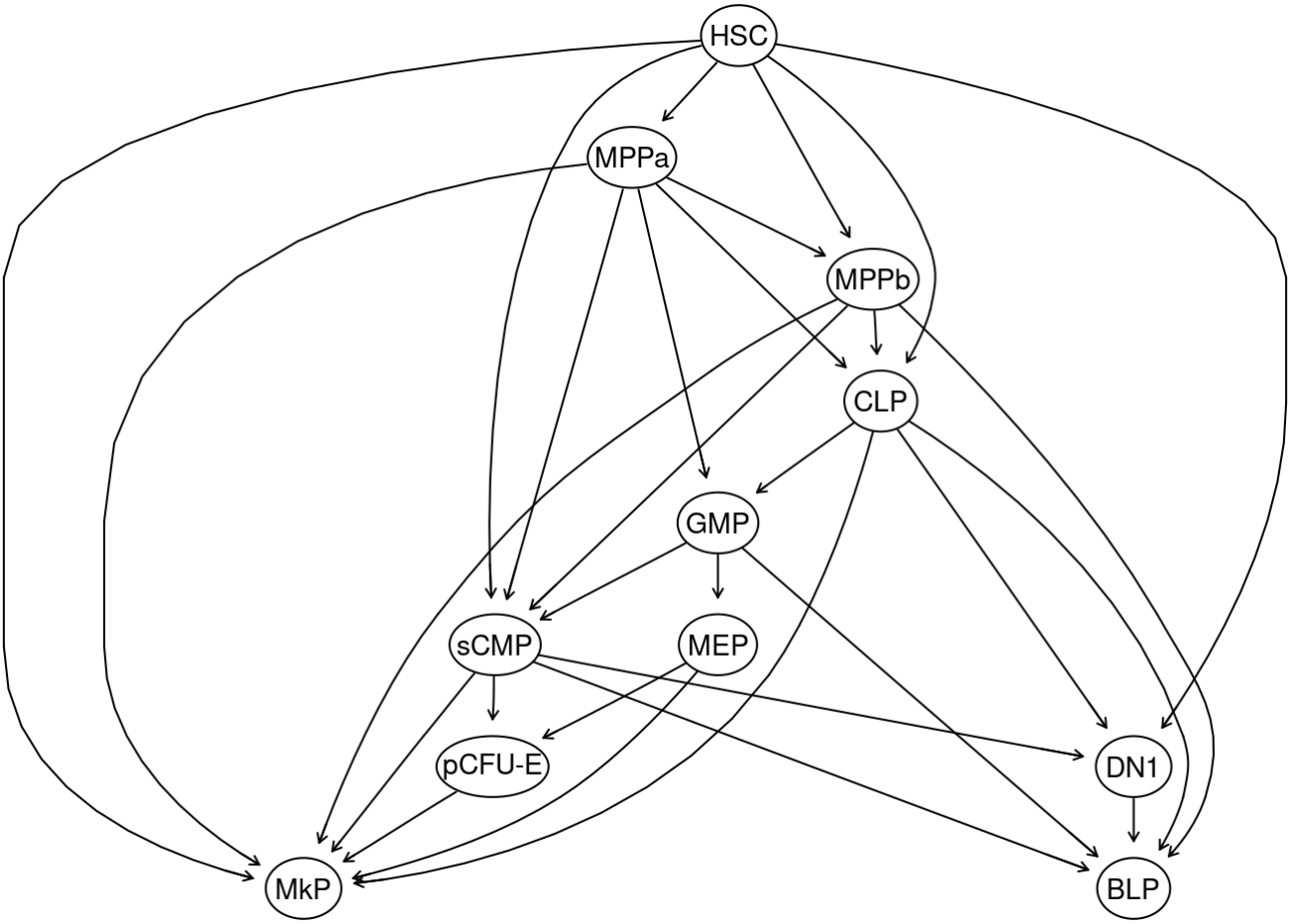
```
## [1] 21678    11
```

First, find variable genes with mutual information.

```
ctypes <- colnames(HMMA)
# Binning expression values
HMMA_disc <- infotheo::discretize(HMMA) %>% as.matrix()
# Calculate the mutual information
doParallel::registerDoParallel(cores = 5)
HMMA_glist <- foreach(i = 1:nrow(HMMA), .combine = c) %dopar% {
  # Higher mi, more variable the gene is
  infotheo::mutinformation(HMMA_disc[i, ], ctypes)
}
names(HMMA_glist) <- rownames(HMMA)
HMMA_glist <- sort(HMMA_glist, decreasing = T)
```

The BN structure is learned with a set of parameters with `BNLearning`.

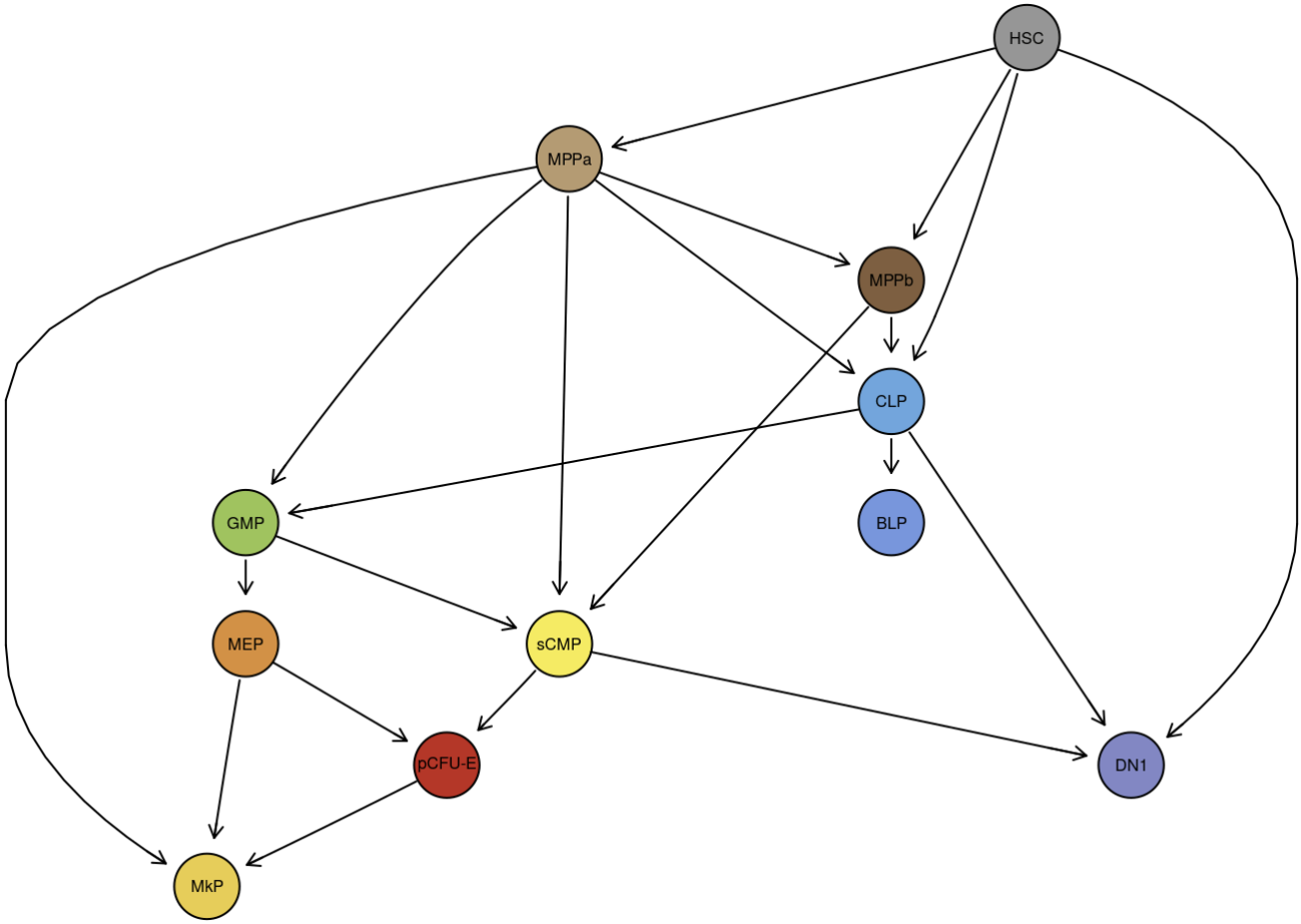
```
dat <- HMMA[names(HMMA_glist)[1:3000], ]
struc_smpls <- BNLearning(dat,
  params = seq(from = 0.08, to = 0.23, length.out = 10),
  root = "HSC", mode = "bulk", ncores = 5, dagMethod = "hc"
)
net_struc <- combineDAGs(struc_smpls, Emin = 20, Emax = 30)
net_struc <- rmCyc(net_struc)
e <- df2bn(net_struc, ctypes, plot = TRUE)
# We take a glimpse of the output DAG
bnlearn::graphviz.plot(e, shape = "ellipse")
```



In the following step, we make use of gene variability information to prune our graph with `trimDAG`.

`trimDAG` automatically plots the modified structure for you. It can be turned off by manually setting `plot = FALSE`.

```
outputDAG <- trimDAG(HMMA, e, 2, 3, .9, plot = FALSE)
# We added colors manually
```



Calculate Effect Matrix

After getting a desired network structure, we can use functions below to calculate the diffBN result. 1. `PerturbResult` : use this function to calculate the BN coefficients after gene perturbation. 2. `EffectMatrix` : extract gene contribution to each edge in the structure. 3. `diffScore` : calculate gene contribution to a set of edges. 4. `dsRank` : give ranked gene list according to previous scores.

```
perturbRes <- PerturbResult(outputDAG, HMMA, mode = "bulk")
effMat <- EffectMatrix(perturbRes, mode = "mean")

edgeSet <- setEdges(fromSet = ctypes, toSet = ctypes) # Calculate for all edges
effectScore <- diffScore(effMat, edgeSet)
geneList <- dsRank(effectScore)
head(geneList)
```

```
## [1] "Gm10883 or Gm1420 or Gm7202 or Igk-C or Igk-J1 or Igk-V28"
## [2] "Car1"
## [3] "Pf4"
## [4] "Mpo"
## [5] "Gm10482"
## [6] "Aldh1a1"
```