CIBER pipeline for single-cell data

Lan-lab

6/17/2024

Structure learning

```
library(Seurat) # Seurat is necessary for singel-cell data
library(foreach)
library(tidyverse)
library(bnlearn) # Core package
library(Matrix) # Core package
library(space) # Core package
library(ciber)

data("HMR")
head(HMR)
```

```
##          orig.ident nCount_RNA nFeature_RNA celltype percent.mt     S.Score
## HSPC_002       HSPC  1085.3333         8778    LTHSC          0 -0.12077291
## HSPC_004       HSPC  1221.0767        11352     LMPP          0 -0.16803616
## HSPC_006       HSPC  1431.2583        11891     LMPP          0 -0.01206173
## HSPC_008       HSPC  1514.3543        10355     LMPP          0 -0.16419568
## HSPC_011       HSPC  1516.5897        10977     LMPP          0 -0.19785059
## HSPC_014       HSPC  1470.6718         9533      MPP          0 -0.16047674
## HSPC_015       HSPC   800.7971         9825     LMPP          0 -0.10753757
## HSPC_017       HSPC  1386.0569        10124    LTHSC          0 -0.17785950
## HSPC_018       HSPC  1876.0635        11314      MPP          0 -0.18039289
## HSPC_020       HSPC  1715.7519        10017    LTHSC          0 -0.15426335
##            G2M.Score Phase old.ident
## HSPC_002 -0.09083001    G1      HSPC
## HSPC_004 -0.10391938    G1      HSPC
## HSPC_006 -0.11841703    G1      HSPC
## HSPC_008 -0.12593563    G1      HSPC
## HSPC_011 -0.15114819    G1      HSPC
## HSPC_014 -0.13218429    G1      HSPC
## HSPC_015 -0.08138777    G1      HSPC
## HSPC_017 -0.12033867    G1      HSPC
## HSPC_018 -0.17387886    G1      HSPC
## HSPC_020 -0.15628029    G1      HSPC
```

```
dim(HMR)
```

```
## [1] 40198   716
```

We use `Seurat::FindVariableFeatures` to identify the variable genes in our data set.

`gem2mem` is used to transform the single-cell data to gene expression matrix of cell types.

```
HMR <- FindVariableFeatures(HMR, selection.method = "vst", nfeatures = nrow(HMR))
glist <- VariableFeatures(HMR)

meta <- HMR$celltype
ctypes <- names(table(HMR$celltype))
blood_celltype <- gem2mem(data.frame(HMR@assays$RNA@data), meta, "mean")
```

The following process discusses a more detailed way to learn the network.

`BNLearning` function samples the data and gives a set of possible structures with different parameters. It is necessary to set the root node manually.

As the structure is learned with different sampling and parameters, not every DAG generated makes great sense. When combining the structures with `combineDAGsmpl`, `Emin` and `Emax` parameters are used to filter the DAGs. Only DAGs with number of edges between `Emin` and `Emax` contribute to the final structure. `Emin` is usually 1-2 times of node number, while `Emax` is usually 2-3 times of node number.

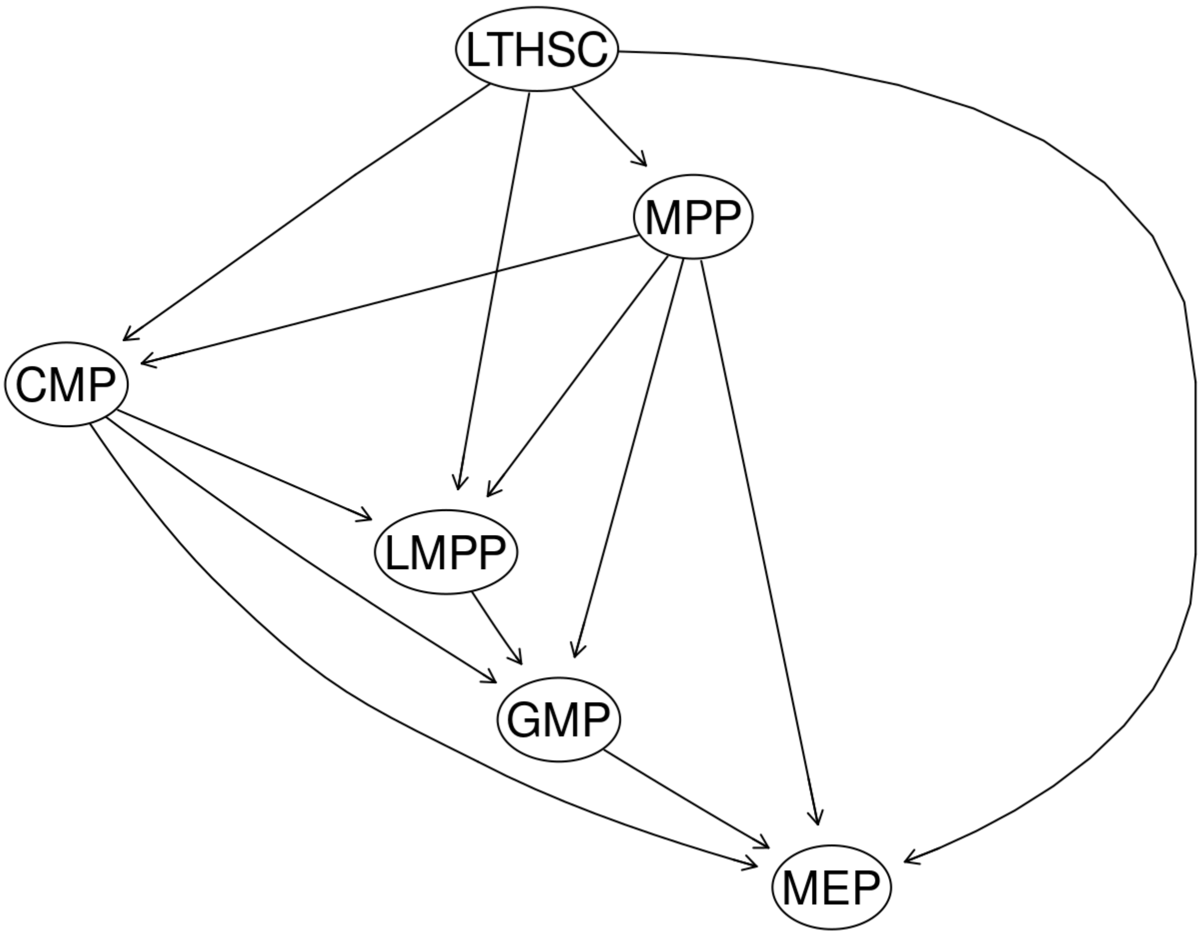
```
dag_smpl <- BNLearning(HMR[glist[1:5000]],
  frac = 0.2, N_smpl = 100, params = seq(0.05, 0.3, 0.05),
  root = "LTHSC", mode = "single_cell", ncores = 200,
  dagMethod = "hc", ugMethod = "cmi2ni"
)
net <- combineDAGsmpl(dag_smpl, Emin = 1, Emax = 7, ncores = 64)

# Let us take a glimpse of the structure now, the number indicates the occurence in p
revious step
net_mat <- net %>% df2mat()
net_mat
```

```
##       CMP GMP LMPP LTHSC MEP MPP
## CMP     0 290  111     0 278  16
## GMP     5   0    0     0   5   1
## LMPP   57  15    0     0   0 120
## LTHSC  30   0  118     0   4 296
## MEP     3   0    0     0   0   1
## MPP   260  12  176     0  11   0
```

The next step is to remove the cycles in the structure. Before we do so, we first trim off the very unlikely edges (i.e. the occurrence less than 2 times).

```
net <- (net_mat * (net_mat > 2)) %>%
  mat2df() %>%
  rmCyc()
e <- df2bn(net, ctypes, plot = TRUE)
```

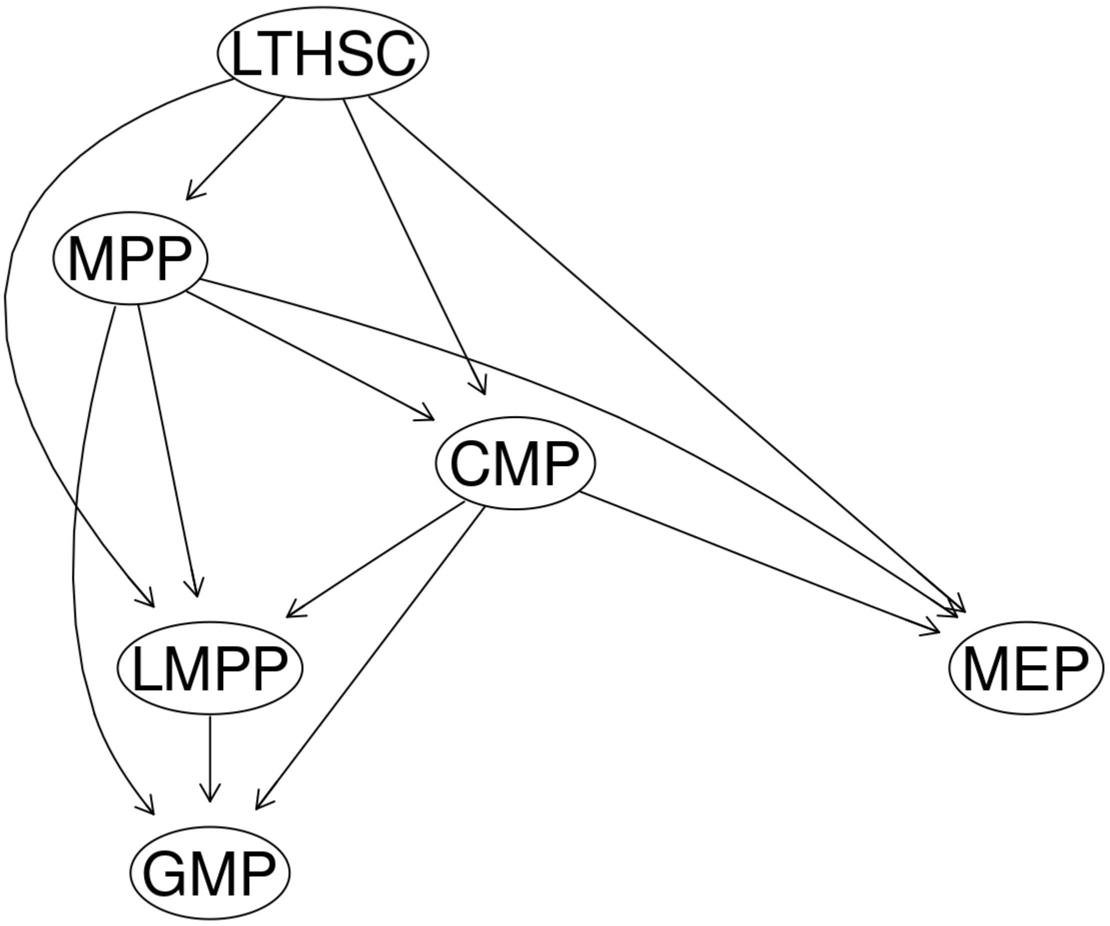


Our very first `e` is too complicated and unrealistic for a differentiation network. In the following step, we make use of gene variability information to prune our graph with `trimDAG`.

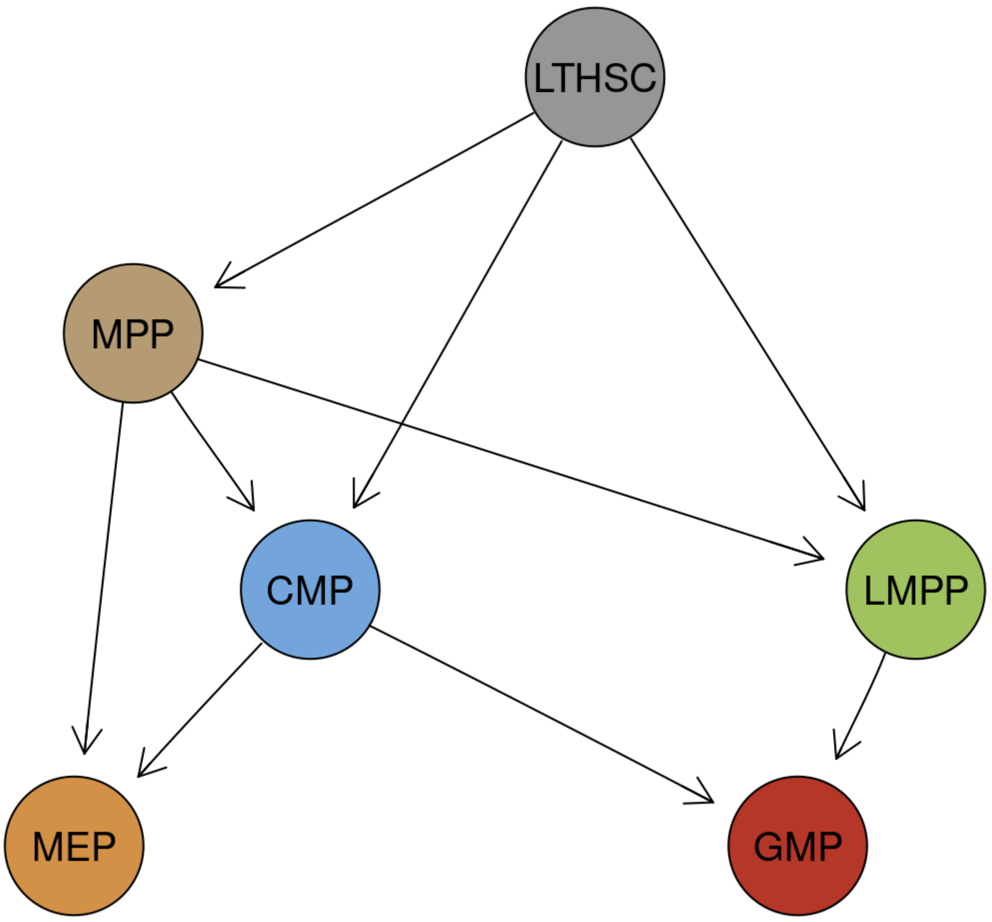
The parameters of `trimDAG` determines how many arcs can there be that point from or to a node. Usually, min_arc is 1 or 2, max_arc is 3 or 4. Besides, edges with too small coefficients are deleted after fitting as well. Threshold_value is normally from 0.8 to 1. Sometimes, you may need to use `trimDAG` for multiples times and on different gene sets to get a more ideal structure.

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

```
tmp <- blood_celltype
tmp2 <- trimDAG(tmp[glist[4000:8000], ], e, min_arc = 2, max_arc = 3, threshold_value
=  1)
```



```
tmp2 <- trimDAG(tmp[glist[1:4000], ], tmp2, min_arc = 2, max_arc = 2, threshold_value
=  1, plot = FALSE)
dag_struc <- tmp2
```



Gene effects on the structure

```r
# Generate indexes for bootstrapping
index <- bootstrap_index(meta, bootstrap_times = 20, ratio = 0.3)
perturbRes <- PerturbResult(dag_struc, HMR, meta, index, n_sample = 20,
                            mode = "single_cell", ncores = 20)
effMat <- EffectMatrix(perturbRes, mode = 'mean')

ctypes <- c("CMP", "GMP", "LMPP",  "LTHSC", "MEP", "MPP")
edgeSet <- setEdges(fromSet = ctypes, toSet = ctypes) # Calculate for all edges
effectScore <- diffScore(effMat, edgeSet)
geneList <- dsRank(effectScore)
head(geneList)
```

```
## [1] "Mpo"             "Ctsg"            "Apoe"            "Plac8"
## [5] "X..not.aligned" "Car1"
```