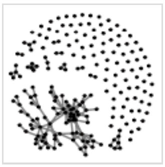

Learn Partial Correlation Disease-Specific Networks V.2

Lillian R Thistlethwaite[1]

[1]Baylor College of Medicine

Version 2

Dec 11, 2020

1 Works for me dx.doi.org/10.17504/protocols.io.bk7xkzpn

Lillian Thistlethwaite
Baylor College of Medicine

ABSTRACT

This protocol describes how to construct disease-specific network structures as described in Thistlethwaite et al. (2020).

Thistlethwaite L.R., Petrosyan V., Li X., Miller M.J., Elsea S.H., Milosavljevic A. (2020). CTD: an information-theoretic method to interpret multivariate perturbations in the context of graphical models with applications in metabolomics and transcriptomics. In review.

DOI

dx.doi.org/10.17504/protocols.io.bk7xkzpn

PROTOCOL CITATION

Lillian R Thistlethwaite 2020. Learn Partial Correlation Disease-Specific Networks. **protocols.io**
https://dx.doi.org/10.17504/protocols.io.bk7xkzpn
Version created by Lillian Thistlethwaite

KEYWORDS

network learning, gaussian graphical models, partial correlation, graphical lasso

CREATED

Sep 09, 2020

LAST MODIFIED

Dec 11, 2020

PROTOCOL INTEGER ID

41943

GUIDELINES

This protocol relies on the R package huge and CTD, which imports igraph.

MATERIALS TEXT

MATERIALS

NONE **Contributed by users Catalog #N/A**

This is a computational workflow. A computer is required, with R version 4.0+ installed.

SAFETY WARNINGS

None

ABSTRACT

This protocol describes how to construct disease-specific network structures as described in Thistlethwaite et al. (2020).

Thistlethwaite L.R., Petrosyan V., Li X., Miller M.J., Elsea S.H., Milosavljevic A. (2020). CTD: an information-theoretic method to interpret multivariate perturbations in the context of graphical models with applications in metabolomics and transcriptomics. In review.

Prepare dataset 1s

1 1s

Load the CTD R package

```
require(CTD)
```

Load the Miller et al 2015 dataset

```
data(Miller2015)
```

Remove metabolite annotation columns from dataset

```
data_mx.og = as.matrix(Miller2015[,grep("IEM_", colnames(Miller2015))])
```

One sample per column, one metabolite per row.

Create diagnosis-patient mappings

```
cohorts = list()
diags = data_mx.og[1,]
cohorts$mcc = names(diags[which(diags=="3-methylcrotonyl CoA
carboxylase")])
cohorts$arg = names(diags[which(diags=="Argininemia")])
cohorts$cit = names(diags[which(diags=="Citrullinemia")])
cohorts$cob = names(diags[which(diags=="Cobalamin biosynthesis")])
cohorts$ga = names(diags[which(diags=="Glutaric Aciduria")])
cohorts$gamt = names(diags[which(diags=="Guanidinoacetate
methyltransferase")])
cohorts$msud = names(diags[which(diags=="Maple syrup urine disease")])
cohorts$mma = names(diags[which(diags=="Methylmalonic aciduria")])
cohorts$otc = names(diags[which(diags=="Ornithine transcarbamoylase")])
cohorts$pa = names(diags[which(diags=="Propionic aciduria")])
cohorts$pku = names(diags[which(diags=="Phenylketonuria")])
cohorts$tmhle = names(diags[which(diags=="Trimethyllysine hydroxylase
epsilon")])
cohorts$ref = names(diags[which(diags=="No biochemical genetic diagnosis")])
```

Create a list object that maps patient identifiers to their respective diagnostic class.

Remove diagnosis row and x-compounds from data_mx.og

```
data_mx.og = data_mx.og[-c(1, grep("x -", rownames(data_mx.og))),]
```

Convert data_mx.og to numeric matrix.

```
data_mx.og = apply(data_mx.og, c(1,2), as.numeric)
```

All elements should be numeric, not character.

Subset data and store reference sample data separately

```
refs = Miller2015[-c(1, grep("x -",
rownames(Miller2015))),which(colnames(Miller2015) %in% cohorts$ref)]
ref_fill = as.numeric(Miller2015$`Times identifed in all 200 samples`[-c(1, grep("x
-", rownames(Miller2015)))])/200
refs2 = refs[which(ref_fill>0.8),]
```

Reference data is used to construct "surrogate profiles" for reference samples.

Only return metabolite data in reference samples associated with a fill rate >80%. Fill rate is the percentage of samples with a z-scored value for a given metabolite.

2 Learn disease-specific network folds for three different network learning paradigms for 5 disease states (citrullinemia, maple syrup urine disease, methylmalonic aciduria, propionic aciduria, phenylketonuria): 5h

i) latent embedding + network pruning ("ind")
ii) latent embedding + no network pruning ("noPruning")
iii) no latent embedding or network pruning ("noLatent")

"ind" networks will be the ig_pruned R objects saved in .RData files in a folder called ind_foldNets.
"noLatent networks will be the ig R object saved in .RData files in a folder called noLatent_foldNets.
"noPruning" networks will be the ig R object in the ind_foldNets .RData files.

Each disease-specific network takes an average of 3-5 minutes to learn. We learn a total of CIT(9)*2+MSUD(18)*2+MMA(9)*2+PA(9)*2+PKU(8)*2 = 106 network folds, 53 per network learning paradigm ("ind" versus "noLatent"). In total, all networks were learned within 5-9 hours.

Learn disease-specific network structures for rare disease datasets

```
require(huge)
for (type in c("ind", "noLatent")) {
  for (model in c("cit", "msud", "mma", "pa", "pku")) {
    for (fold in 1:length(cohorts[[model]])) {
      print(sprintf("Learning graphs for diag %s, fold %d...", model, fold))
      diag_pts = cohorts[[model]][-fold]
      print(diag_pts)
      fill.rate = 1-apply(data_mx.og[,which(colnames(data_mx.og) %in% diag_pts)], 1,
sum(is.na(i))/length(i))
      diag_data = data_mx.og[intersect(which(ref_fill>0.8), which(fill.rate>0.80)),
which(colnames(data_mx.og) %in% diag_pts)]
      diag_data = diag_data[which(rownames(diag_data) %in% rownames(refs2)),]
      if (type=="noLatent") {
        print("Disease only, no pruning, no latent variable embedded / differential net
        diag_data = data.surrogateProfiles(data = diag_data, std = 1, ref_data = NULL)
      } else {
        print("Individual samples as training data. Latent variable embedding and netw
        diag_data = data.surrogateProfiles(data = diag_data, std = 1, ref_data = refs2
      }
      print(dim(diag_data))

      # Disease Network: GLASSO approach
      inv_covmatt = huge(t(diag_data), method="glasso")
      inv_covmatt_select = huge.select(inv_covmatt, criterion = "stars")
      inv_covmat =
as.matrix(inv_covmatt_select$icov[[which(inv_covmatt_select$lambda==inv_covmat
      diag(inv_covmat) = 0;
      colnames(inv_covmat) = rownames(diag_data)
      ig = graph.adjacency(as.matrix(inv_covmat), mode="undirected", weighted=TR
add.colnames='name')
      V(ig)$name = rownames(diag_data)
      print(ig)

      if (type=="ind") {
        # Reference Network: GLASSO approach
        # Reference Network: GLASSO approach
        ref_data = data.surrogateProfiles(data = refs2, std = 1, ref_data = refs2)
        ref_data = ref_data[,-which(duplicated(colnames(ref_data)))]
        print(dim(ref_data))
        inv_covmatt = huge(t(ref_data), method="glasso", lambda = inv_covmatt_sele
        inv_covmat = as.matrix(inv_covmatt$icov[[1]])
        diag(inv_covmat) = 0;
        colnames(inv_covmat) = rownames(ref_data)
        ig_ref = graph.adjacency(as.matrix(inv_covmat), mode="undirected", weighte
add.colnames='name')
        V(ig_ref)$name = rownames(ref_data)
        print(ig_ref)

        ig_pruned = graph.naivePruning(ig, ig_ref)
        print(ig_pruned)
        save(ig, ig_ref, ig_pruned, file=sprintf("bg_%s_%s_fold%d.RData", model, type,
        rm(ig, ig_pruned)
      } else {
        save(ig, file=sprintf("bg_%s_%s_fold%d.RData", model, type, fold))
        rm(ig)
      }
    }
  }
}
```

Two solutions are provided to learn network structures on very underranked (large feature space with small number of examples) data (as is the case for rare disease):

1. Add surrogate profiles to fill in the rank of the data matrix.
2. Use the glasso algorithm