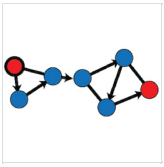

Dec 11, 2020

Precompute Node Rankings

Lillian R Thistlethwaite[1]

[1]Baylor College of Medicine

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

Lillian Thistlethwaite
Baylor College of Medicine

ABSTRACT

This protocol describes how probability diffusion algorithm informs the path a network walker takes in a disease-specific network as described in Thistlethwaite et al. (2020). Briefly, using a probability diffusion walker, a network walker decides which nodes to step into, until a specified node subset has been found, or a given number of "missteps" occur.

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.bkecktaw

PROTOCOL CITATION

Lillian R Thistlethwaite 2020. Precompute Node Rankings. **protocols.io** https://dx.doi.org/10.17504/protocols.io.bkecktaw

KEYWORDS

network walker, network search algorithm, greedy search, probability diffusion algorithm, information propagation

CREATED

Aug 27, 2020

LAST MODIFIED

Dec 11, 2020

PROTOCOL INTEGER ID

41124

GUIDELINES

This computational workflow is best performed using a computing cluster with a job scheduler.

MATERIALS TEXT

MATERIALS

NONE **Contributed by**

users Catalog #N/A

This is a computational workflow. The only materials required are a computer, R version 4.0+ installed, and some required R packages.

SAFETY WARNINGS

None

DISCLAIMER:

> This computational workflow is best performed using a computing cluster and a job scheduler.

ABSTRACT

This protocol describes how probability diffusion algorithm informs the path a network walker takes in a disease-specific network as described in Thistlethwaite et al. (2020). Briefly, using a probability diffusion walker, a network walker decides which nodes to step into, until a specified node subset has been found, or a given number of "missteps" occur.

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.

wrapper_ranks.r 15m

1 15m

This Protocol should be completed on a computing cluster, ideally. Running node ranks in serial takes a much longer amount of time, depending on the size of the network and the number of starting nodes you want to precompute node ranks for. If you launch a job for each starting node you want node ranks for and then collate those node ranks into one list object, you can process ~500 starting node node rankings in less than 15 minutes. We implement the computation of node ranks using this strategy with the following set-up:

```
# FILE HIERARCHY:
#       graphs/ folder with all bg_[model]_[type]_fold[fold].RData files in it.
#       wrapper_ranks.r
#       getRanks.pbs
#       getRanksN.r
# DIRECTIONS: Run wrapper_ranks.r on your computing cluster with for each pruned disease network. Set type="ind"
#           since we want the node ranks for the pruned disease-specific network (which uses latent embedding and
#           network pruning). type="noPruning" is used for the disease+reference network (no network pruning) and
#           type="noLatent" is used for a disease-only network (no latent embedding or network pruning).
Rscript wrapper_ranks.r [model] [fold] [type]
Rscript wrapper_ranks.r "CIT" 1 "ind"
Rscript wrapper_ranks.r "CIT" 2 "ind"
Rscript wrapper_ranks.r "CIT" 3 "ind"
Rscript wrapper_ranks.r "CIT" 4 "ind"
Rscript wrapper_ranks.r "CIT" 5 "ind"
Rscript wrapper_ranks.r "CIT" 6 "ind"
Rscript wrapper_ranks.r "CIT" 7 "ind"
Rscript wrapper_ranks.r "CIT" 8 "ind"
Rscript wrapper_ranks.r "CIT" 9 "ind"
# etc. for "MSUD", "MMA", "PA" and "PKU", too.
```

wrapper_ranks.r

```
args = commandArgs(trailingOnly=TRUE)
diag = args[1]
fold = as.numeric(args[2])
type = args[3]

require(igraph)
output_dir = sprintf("./diag%s%d", diag, fold)
str = sprintf("mkdir %s", output_dir)
system(str)
```


Citation: Lillian R Thistlethwaite (12/11/2020). Precompute Node Rankings. https://dx.doi.org/10.17504/protocols.io.bkecktaw

```
require(R.utils)
require(igraph)
if (type=="noLatent") {
  ig =
loadToEnv(sprintf("../graphs/noLatent_foldNets/bg_%s_noLatent_fold%d.RData",
tolower(diag), fold))[["ig"]]
} else if (type=="ind") {
  ig = loadToEnv(sprintf("../graphs/ind_foldNets/bg_%s_ind_fold%d.RData",
tolower(diag), fold))[["ig_pruned"]]
} else {
  ig = loadToEnv(sprintf("../graphs/ind_foldNets/bg_%s_ind_fold%d.RData",
tolower(diag), fold))[["ig"]]
}
totalN = length(V(ig)$name)
print(sprintf("Total N = %d", totalN))

for (n in 1:totalN) {
  str = sprintf("qsub -v diag=%s,fold=%d,n=%d getRanks.pbs", diag, fold, n)
  system(str, wait=FALSE)
  system("sleep 0.2")
}

ready=FALSE
last_sum = 0
while (!ready) {
  f = list.files(sprintf("./diag%s%d", diag, fold), pattern=".RData")
  print(sprintf("#permutation files ready = %d", length(f)))
  if (length(f)==totalN) {
    ready=TRUE
  } else {
    system("sleep 20")
    system("rm *.pbs.*")
    curr_sum = length(f)
    if (curr_sum > last_sum) {
      last_sum = curr_sum
    }
  }
}
system("rm *.pbs.*")

# collate permutations into one R object
all_nodes = V(ig)$name
permutationByStartNode = list()
for (n in 1:length(all_nodes)) {
  load(sprintf("./diag%s%d/%d.RData", diag, fold, n))
  permutationByStartNode[[n]] = toupper(current_node_set)
}
names(permutationByStartNode) = all_nodes
save.image(file=sprintf("%sRanks_miller/%s%d-ranks.RData", type, toupper(diag),
fold))
print("collate complete...")
str = sprintf("rm -r ./diag%s%d", diag, fold)
```


Citation: Lillian R Thistlethwaite (12/11/2020). Precompute Node Rankings. https://dx.doi.org/10.17504/protocols.io.bkecktaw

system(str)

This is the wrapper_ranks.r R code, which uses a PBS job launcher qsub protocol to send an Rscript command to the head node of a computing cluster. If you use a different job launcher protocol, you can substitute the sprintf() command pasted below for the appropriate job launcher syntax that works for your computing cluster set-up:

```
str = sprintf("qsub -v diag=%s,fold=%d,n=%d,type=%s getRanks.pbs", diag, fold, n, type)
```

getRanks.* 1m

2

1m

getRanks.pbs

```
# Request 1 processors on 1 node
#PBS -l nodes=1:ppn=1
#Request x number of hours of walltime
#PBS -l walltime=1:00:00
#Request that regular output and terminal output go to the same file
#PBS -j oe
#PBS -m abe
# Load R version 4.0
module load R/4.0
# Go to wherever your files are on your cluster
cd metabolomics/singleNodeRanks
diag=${diag}
fold=${fold}
n=${n}
type=${type}
Rscript getRanksN.r $diag $fold $n $type > ranks:$diag-$fold-$n-$type.out
rm ranks:$diag-$fold-$n-$type.out
```

This is the PBS launcher script that gets called by wrapper_ranks.r. If your computing cluster uses a different job launcher protocol, you'll use a different file type for this step.

Citation: Lillian R Thistlethwaite (12/11/2020). Precompute Node Rankings. https://dx.doi.org/10.17504/protocols.io.bkecktaw

getRanksN.r

```
require(igraph)
args = commandArgs(trailingOnly=TRUE)
diag = args[1]
fold = as.numeric(args[2])
n = as.numeric(args[3])
type = args[4]

print(diag)
print(fold)
print(n)
print(type)

require(CTD)
require(R.utils)
if (type=="noLatent") {
  ig =
loadToEnv(sprintf("../graphs/noLatent_foldNets/bg_%s_noLatent_fold%d.RData",
tolower(diag), fold))[["ig"]]
} else if (type=="ind") {
  ig = loadToEnv(sprintf("../graphs/ind_foldNets/bg_%s_ind_fold%d.RData",
tolower(diag), fold))[["ig_pruned"]]
} else {
  ig = loadToEnv(sprintf("../graphs/ind_foldNets/bg_%s_ind_fold%d.RData",
tolower(diag), fold))[["ig"]]
}
print(ig)
V(ig)$name = tolower(V(ig)$name)
G = vector(mode="list", length=length(V(ig)$name))
names(G) = V(ig)$name
adj_mat = as.matrix(get.adjacency(ig, attr="weight"))
p1=0.9
thresholdDiff=0.01
all_nodes = tolower(V(ig)$name)
print(head(all_nodes))
current_node_set = singleNode.getNodeRanksN(n, G, p1, thresholdDiff, adj_mat)
new_dir =  sprintf("./diag%s%d", diag, fold)
if (!dir.exists(new_dir)) {
  str = sprintf("mkdir %s", new_dir)
  system2(str)
}
save(current_node_set, file=sprintf("./diag%s%d/%d.RData", diag, fold, n))
```

This is the R code for the getRanksN.r that gets called by getRanks.pbs.

Citation: Lillian R Thistlethwaite (12/11/2020). Precompute Node Rankings. https://dx.doi.org/10.17504/protocols.io.bkecktaw