TargetRepositioningProstateCancer

DiscoNet R package (vignette)

Introduction

DiscoNet is an *R* package containing an ensemble of functions to automatically extract node features from complex biological network that can later be used by machine learning algorithms for prediction, classification, etc.

The main categories of features that can be extracted from a set of nodes (*start_nodes*) using ***DiscoNet*** are:

- **Propagation-based features**: simple one-to-one distances between *start_nodes* and other nodes in the network.
  The distances can be directed, inversely directed, or undirected. (`extract_by_shp`, `extract_by_shp_inv`, `extract_by_shp_ind`, `extract_by_rwr`,`extract_by_rwr_inv`, `extract_by_rwr_ind`).
- **Topological metrics and similarities**: common metrics such as degree, betweenness, or centrality are extracted, as well as node similarity based on node neighborhoods (`extract_topolo`).
- **Module-based**: Will identify clusters from the network and compute the distance between each *start_nodes* these clusters, also works with a user-wn list of pre-calculated clusters (`extract_cluster`).
- **Signature-based**: Based on a user-own list of nodes (signature), different distances between *start_nodes* and these nodes can be calculated (`extract_by_sig`).

This vignette shows a script example on how to use **DiscoNet** to extract features from the A3LNetwork to do therapeutic target repositioning on prostate cancer (supervised classification).

```
# You'll need these four libraries to run this vignette, they can be be easily installed like so:
# if (!require("BiocManager", quietly = TRUE))
#   install.packages("BiocManager")
#
# BiocManager::install("devtools")
# BiocManager::install("tidyverse")
# BiocManager::install("igraph")
# devtools::install_github("https://github.com/MilanPicard/DiscoNet")

library(DiscoNet)
library(tidyverse)
#> ── Attaching core tidyverse packages ──────────────────────── tidyverse 2.0.0 ──
#> ✔ dplyr     1.1.4     ✔ readr     2.1.5
#> ✔ forcats   1.0.0     ✔ stringr   1.5.1
#> ✔ ggplot2   3.5.1     ✔ tibble    3.2.1
#> ✔ lubridate 1.9.3     ✔ tidyr     1.3.1
#> ✔ purrr     1.0.2
#> ── Conflicts ────────────────────────────────────────── tidyverse_conflicts() ──
#> ✖ dplyr::filter() masks stats::filter()
#> ✖ dplyr::lag()    masks stats::lag()
#> ℹ Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become
```

```
library(igraph)
#>
#> Attaching package: 'igraph'
#>
#> The following objects are masked from 'package:lubridate':
#>
#>     %--%, union
#>
#> The following objects are masked from 'package:dplyr':
#>
#>     as_data_frame, groups, union
#>
#> The following objects are masked from 'package:purrr':
#>
#>     compose, simplify
#>
#> The following object is masked from 'package:tidyr':
#>
#>     crossing
#>
#> The following object is masked from 'package:tibble':
#>
#>     as_data_frame
#>
#> The following objects are masked from 'package:stats':
#>
#>     decompose, spectrum
#>
#> The following object is masked from 'package:base':
#>
#>     union
```

```
user_verbose = FALSE
```

Feature extraction

Disconet needs at least a biological network from which to extract features.
If no *start_nodes* are given, features will be extracted for every node in the network.
A three layer network called A3LNetwork (protein+gene+go) of medium size is preloaded.
It is comprised of:

- a PPI network (from stringdb)
- a gene layer connected in a directed manner to the PPI layer
- a GO layer connected to the gene layer.

How the network was built can be accessed in *data-raw*/DATASET.R
Nodes `types` are described in their name ('Prot_', 'Gene_', 'GO_'), as or node attribute.

```
A3LNetwork = DiscoNet::A3LNetwork
summary(A3LNetwork)
#> IGRAPH 0620eb7 DN-B 7538 53919 --
#> + attr: name (v/c), type (v/c)
```

```
table(V(A3LNetwork)$type)
#>
#>    gene      go protein
#>    3757    1816    1865
```

This vignette is a perform a simple target repositioning against prostate cancer
We will extract features for every protein in the network, as any of them could interesting. Let's calculate some basic features.

```
# Get all proteins beginning with Prot_
all_proteins = vnames(A3LNetwork, pattern = "Prot_")

# Calculate topological features for the 1 865 proteins in the network
TopologicalMetrics = extract_topolo(Graph = A3LNetwork,
                                    start_nodes = all_proteins,
                                    verbose = user_verbose)
dim(TopologicalMetrics)
#> [1] 1865 7549
```

```
# Calculate for the 1 865 proteins their random walk distance to every other nodes
# this will take a bit of time, especially without nCores = 1 (no parallel computing)
# Faster distance calculation include the shortest path ones.
# shp_dist = extract_by_shp(Graph = A3LNetwork, start_nodes = all_proteins)
rwr_dist = extract_by_rwr(Graph = A3LNetwork,
                          start_nodes = all_proteins,
                          nCores = 1,
                          verbose = user_verbose)
dim(rwr_dist)
#> [1] 1865 7539
```

```
# Additional distance measures can be obtained with extract_by_rwr_inv or extract_by_rwr_ind
```

We can also extract module-based features

```
# Based on precalculated clusters done on A3LNetwork
Protein_clusters = DiscoNet::data_prostate_cancer$Clusters

# They can also be calculated using the following code
# # Create PPI subgraph
# A3LNetwork_prot =igraph::induced_subgraph(A3LNetwork, vids = all_proteins)
# # Calculate clusters, optionnaly, you don't have to calculate clusters if you have precalculate
# A3LNetwork_prot_clust = extract_cluster(Graph = A3LNetwork_prot, start_nodes = all_proteins, o
# # Only take clusters of node size higher than 10
# A3LNetwork_prot_clust = A3LNetwork_prot_clust[nbrs(A3LNetwork_prot_clust) > 10] # Results in 55


# Proximity of every  protein to each cluster is calculated based on the distance matrix that we
# If no clusters are given, it will calculate its own based on the whole network using cluster_wa
Dist_cluster = extract_cluster(A3LNetwork,
                               all_proteins,
                               dist_matrix = rwr_dist,
                               Cluster_list = Protein_clusters,
                               verbose = user_verbose)
dim(Dist_cluster)
#> [1] 1865  197
```

Signature-based features necessitates a list of genes (or proteins) to be used as a signature.
The user should provide that signature.
A preloaded signature is included within ***DiscoNet***

```
data_prostate_cancer = DiscoNet::data_prostate_cancer

# Clusters should be converted to the same ID than the gene layer ID
Gene_clusters = data_prostate_cancer$Clusters_gene
# Remove genes not present in the network, or else an error will be returned
Gene_clusters = lapply(Gene_clusters, function(x) intersect(x, vnames(A3LNetwork)))

# Get preloaded gene signatures for prostate cancer, it contains a list of signatures and also kn

# Remove genes not present in the network (it is rather a small network), resulting in 25 mutated
Signature_for_prostate_cancer = lapply(data_prostate_cancer$Signatures,
                                       function(x) intersect(x, vnames(A3LNetwork)))

# Extract signature-based features for 1 865 proteins
Dist_signature = extract_by_sig(A3LNetwork,
                                all_proteins,
                                Signature_for_prostate_cancer,
                                Gene_clusters,
                                rwr_dist,
                                verbose = user_verbose)
dim(Dist_signature)
#> [1] 1865   58
```

Feature selection

Once some features are extracted, it is often necessary to reduce their dimensionality before applying machine learning algorithm.
DiscoNet make accessible two supervised variable selection method:

- a quick but robust selection based on information gain with bootstraping
- a longer but more powerful selection based on Adaptive LASSO.

```
# merge all feature datasets together.
Full_features = purrr::reduce(.x = list(TopologicalMetrics, rwr_dist, Dist_cluster, Dist_signatur
                              merge, by = c('Target'), all = T)

Full_features %>% dim
#> [1]  1865 15250
```

```
# That is a lot of features, so variable selection will be done in order to keep only the most in
# To run information gain variable selection, a class must be given.
# In this vignette, the class will be based on whether or not a protein is a known therapeutic ta
# data_prostate_cancer$Targets contains 13 known targets. The other proteins are therefore assume
class = Full_features$Target %in% data_prostate_cancer$Targets

# Create a training set and a test set
# training set contains all positive observations and a random selection of 200 negative observat
set.seed(123)
index = c(which(class),sample(which(!class), 300))

train.x = Full_features[index, -1]
train.y = class[index]

test.x = Full_features[-index, -1]
test.y = class[-index]

# Run information gain on different bootstraps
infgain_full = InformationGain_Bootstrap(df = train.x, class = train.y, nbr_of_boot = 3, seed = 1
# Take the best 100 features (with the highest information gain)
best_features = infgain_full$feature[1:100]
```

There! The 100 best features extracted from the network based on this problem are retrieved.

Machine learning classification

Optionnaly, machine learning can be done to predict new targets for prostate cancer.

```
# # Using your favorite machine learning algorithms, you can now use these features to predict ne
# # Example:
# # if (!require("BiocManager", quietly = TRUE))
# #     install.packages("BiocManager")
# #
# # if (!require("kknn", quietly = TRUE))
# #     BiocManager::install("kknn")
# #
# library("kknn")
# #
# # Using the k-nearest neighbors algorithm to train of these features on the training set.
# set.seed(123)
# KNN_mod = kknn::train.kknn(Y~., data = data.frame(Y = as.factor(train.y), train.x[, best_featur
# conf_mat = table(preds = KNN_mod$fitted.values[[1]], actuals = KNN_mod$data$Y)
# print(paste0("Accuracy: ", round(sum(diag(conf_mat))/sum(conf_mat), 2)))
# print(paste0("Sensibility: ", round(conf_mat[1, 1]/sum(conf_mat[1, ]), 2)))
# print(paste0("Sensitivity: ", round(conf_mat[2, 2]/sum(conf_mat[2, ]), 2)))
#
# # Predict new potential targets on the bigger test set.
# PotentialNewTarget = Full_features$Target[-index][predict(KNN_mod, test.x[, best_features]) ==
```