NFP:Network fingerprint - a knowledge-based characterization of biomedical networks

Yang Cao, Fei Li, Haochen He

November 18, 2016

Contents

1 Introduction

Recent network fingerprint method based on GO knowledge and propagation clustering mentioned in [3] provide a novel representation of network differentiation known as biological spectra, or Network Fingerprint. This method was used to describe the relationship between multiple disease networks and its related pathways, and to visually compare and parse different diseases by generating a fingerprint overlay. Thus, we achieve the function of complex network comparison based on network fingerprint on the open scientific computing platform *R*R, and present *NFP*, an *Bioconductor*Bioconductor package for fingerprint-based network analyzing and comparison of systems. Driven by the research needs of customers, *NFP* provides a unified interface to three similarity clustering algorithms. In addition, *NFP* can also provide multiscale statistical analysis and visualization, access to the specific attributes of different disease networks.

This manual is a brief introduction to structure, functions and usage of *NFP* package. The NFP package provides a set of functions to support knowledge-based network fingerprint (NFP) framework. A biomedical network is characterized as a spectrum-like vector called "network fingerprint", which contains similarities to basic reference networks. This framework provides a more intuitive way to decipher molecular networks, especially for large-scale network comparisons and clustering analyses.

The three main features of *NFP*:

- Basic reference networks generation.
- Network comparison, which encompasses network merging, annotation and similarity scoring.
- Network standardization.

1.1 Installation

NFP requires these packages: *magrittr*, *igraph*, *plyr*, *ggplot2*, *apcluster*, *dplyr*, *stringr*, *graph* and *KEGGgraph*. To install **NFP**, please note especially two depencies of **NFP**, **graph** and **KEGGgraph** are only available from [Bioconductor](www.bioconductor.org). Appanrantly, function 'install.packages' can not insall Biocondutor packages. There

is a function 'biocLite', a wrapper around 'install.packages' provided by Bioconductor, can be used to install both CRAN and Bioconductor packages simply. More details on 'biocLite' is available from https://www.bioconductor.org/install/. Thus, users can install NFP install the latest released version using 'biocLite' directly:

```
## install release version of NFP
source("http://bioconductor.org/biocLite.R")
biocLite("NFP")
```

or install the Bioconductor dependencies package first:

```
## install release version of NFP
source("http://bioconductor.org/biocLite.R")
biocLite(c("graph","KEGGgraph"))
install.packages("NFP")
```

It also allows users to install the latest development version from github, which requires **devtools** package has been installed on your system (or can be installed using 'install.packages("devtools")'). Note that devtools sometimes needs some extra non-R software on your system – more specifically, an Rtools download for Windows or Xcode for OS X. There's more information about devtools [here](https://github.com/hadley/devtools).

```
## install NFP from github, require biocondutor dependencies package pre-installed
if (!require(devtools))
  install.packages("devtools")
devtools::install_github("yiluheihei/NFP")
```

After installation, the *NFP* is ready to load into the current workspace by the following codes to the current workspace by typing or pasting the following codes:

Moreover, gene similarity data used in our *NFP* package is stored in a external data repository *NFPdata* https://github.com/yiluheihei/datarepo for the large size (about 16 MB). More details on how to construct External Data Repositories using the Additional_repositories field see The Coatless Professor blog post http://thecoatlessprofessor.com/programming/r-data-packages-in-external-data-repositories-using-the-additional_repositories-field/. Thus, users must install the *NFPdata* before the networkfinger print analyis as following code.

```
if (!require("NFPdata")) {
    install_data_package()
}
```

2 Analysis Pipeline: from Basic Reference Network Generation to Network Networkfingerprint Visualization

We will demonstrate go through an analysis pipeline to illustrate some of the main functions in *NFP*. This pipeline consists of several steps:

1. Basic Reference Network Generation: prepare the well-known biomedical netowks as the NFP framework reference networks. Several pathway databases have been developed for biological network research, e.g. KEGG, Reactome (www.reactome.org). All of this pathway databases is well-stuied and can be used as the basic reference networks of NFP.
2. Network fingerprint calculation: The similarity between two biomedical networks is calculated based on the following intition: grouping the nodes in the merged network into strongly inter-connected communities with high functional similarity score between intra-community nodes in different networks The functional similarity was measured based on GO [5]. And we employed affinity propagation (AP) [2] clustering algorithm to detect the aligned functional modules between the two networks to be compared.
3. Network fingerprint Visualization: Show the network fingerprint along all the reference networks. We could observe the differences of different biological networks' fingerprint intuitively from visualization.

2.1 Generating well-studied basic reference networks

The basic idea of calculating the network fingerprint is to have the biomedical networks map to well-studied basic networks. KEGG PATHWAY is a collection of manually drawn pathway maps representing our knowledge on the molecular interaction and reaction networks. Since its first introduction in 1995, KEGG PATHWAY has been widely used as a reference knowledge base for understanding biological pathways and functions of cellular processes. The knowledge from KEGG has proven of great value by numerous work in a wide range of fields [1]. So by default, we take KEGG pathways as basic reference networks in *NFP* by default.

Function `load_KEGG_refnet` can be used to retrieve the KEGG pathway maps with KEGG API http://www.kegg.jp/kegg/rest/keggapi.h In KEGG pathways, only the pathways of the map are manual drawing, and to different organisms, the map reference helps the automatic generation of organism-specific pathway for each organism. The `organism` (e.g. `organsim ="hsa"`) parameter indicate the organism name of KEGG pathway maps.

```
## donot run, retrive pathway maps from KEGG database may take several minutes,
## we have pre-stored this data in our package
## kegg_refnet <- load_KEGG_refnet(organism = 'hsa')
data(kegg_refnet)

# show the kegg reference networks
show(kegg_refnet)
```

```
## Basic networks of organism hsa
##
## 134 basic networks; classification into 4 groups:
##                                 group_name net_num
## group1       Genetic Information Processing      22
## group2 Environmental Information Processing      28
## group3                   Cellular Processes      15
## group4                   Organismal Systems      69
##                                                 net_name
## group1             RNA polymerase - Homo sapiens (human), ...
## group2           ABC transporters - Homo sapiens (human), ...
## group3               Endocytosis - Homo sapiens (human), ...
## group4 Hematopoietic cell lineage - Homo sapiens (human), ...
```

We defined a new S4 class *NFPrefnet* to store the *NFP* reference networks. *NFP* also provides five kinds of methods for this S4 class:

1. `net`: Exact the basic reference networks of *NFPRefnet*.

2. `group`: Obtain the group information, group names, number and the size of each group,e.g. KEGG pathway database contains seven group pathway maps.

3.`subnet`: Extract or replace parts of the *NFP* basic reference networks.

4.`show`: Display of *NFPRefnet*.

5.`name`: Extract the names of reference networks.

```
## group information of kegg reference networks
refnet_group <- group(kegg_refnet)
show(refnet_group)
```

```
## $name
## [1] "Genetic Information Processing"
## [2] "Environmental Information Processing"
## [3] "Cellular Processes"
## [4] "Organismal Systems"
```

```
##
## $num
## [1] 4
##
## $size
##       Genetic Information Processing Environmental Information Processing
##                                   22                                   28
##                   Cellular Processes                  Organismal Systems
##                                   15                                   69
```

```
## select goup 1 and 2, and subset this two groups
selected_group <- refnet_group$name[c(1,2)]
NFPnet <- subnet(kegg_refnet,selected_group)
NFPnet
```

```
## Basic networks of organism hsa
##
## 50 basic networks; classification into 2 groups:
##                                  group_name net_num
## group1       Genetic Information Processing      22
## group2 Environmental Information Processing      28
##                                          net_name
## group1   RNA polymerase - Homo sapiens (human), ...
## group2 ABC transporters - Homo sapiens (human), ...
```

Detailed instructions for this five methods refer to package function help.

Obviously users can also customize a NFPRefnet as a reference for computing network fingerprint. Users can refer to the documents of *NFPRefnet* about the composition details of this class. *graphite* [6] allow users to build *graphNEL* object from several pathway databases.

```
## Reactome human pathway maps
require(graphite)
human_pathway <- pathways("hsapiens", "kegg")
## just choose first two pathway maps for testing
p <- human_pathway[1:2]
show(p)
```

```
## KEGG pathways for hsapiens
## 2 entries, retrieved on 03-05-2016
```

```
g <- lapply(p, pathwayGraph)
show(g)
```

```
## $`Acute myeloid leukemia`
## A graphNEL graph with directed edges
## Number of Nodes = 57
## Number of Edges = 152
##
## $`Adherens junction`
## A graphNEL graph with directed edges
## Number of Nodes = 71
## Number of Edges = 190
```

Then users can create their own customized *NFPRefnet* object as following:

```
## here, just take the above two reactome pathway maps as NFP basic reference
## networks as example
```

```
g_names <- names(human_pathway)[1:2]
## only one group and two reference networks
customized_refnet <- new("NFPRefnet",network = list(g), name = list(g_names),
  group = "test group", organism ='hsa')

## methods of NFPRefnet
show(customized_refnet)
```

```
## Basic networks of organism hsa
##
## 2 basic networks; classification into 1 groups:
##        group_name net_num                    net_name
## group1 test group       2 Acute myeloid leukemia, ...
```

```
group(customized_refnet)
```

```
## $name
## [1] "test group"
##
## $num
## [1] 1
##
## $size
## [1] 2
```

```
subnet(customized_refnet, 'test group', 1)
```

```
## Basic networks of organism hsa
##
## 1 basic networks; classification into 1 groups:
##        group_name net_num                    net_name
## group1 test group       1 Acute myeloid leukemia, ...
```

2.2 Network fingerprint calculation

NFP algorithm consists of three steps: merging network, nodes clustering and similarity scoring.

Network merging. The two networks to be compared are first merged into one. Given two networks *G1* and *G2*, the merged network *Gm* is constructed by connecting each node between the *G1* and *G2* network. Two nodes corresponding to the same protein in the merged network are replaced by a single node that inherited all the interactions from the two individual nodes in the subsequent process.

Clustering in merged network. Grouping the nodes in the merged network into strongly inter-connected communities with high functional similarity score between intra-community nodes in different networks. We employed affinity propagation (AP) clustering algorithm to detect the aligned functional modules between the two networks to be compared. The nodes are grouped on the cluster based on nearest neighbor analysis.

Similarity scoring. The calculation of similarity score is processed in two steps: First, local similarity for each cluster and network similarity among cluster. Second, standardization: the original similarity score depends on the topological properties of query network to some extent. There is implicit bias of network fingerprint, because the outliers could be greatly distorted the relevant pattern presented in the network fingerprint. In order to eliminate the possible topological weight differences, the similarity calculation process of each node are standardized processing and the final network fingerprint facing to users is totally standardized. The standardization process is based on the random distribution of similarity scores. To the number of nodes, the number of edges and node degree, these three topological properties of random network for standardized estimate are consistent with the original network.

To not affect the results of standardization and improve the efficiency of fingerprint calculation, we set limit on the

permutation time (the default is 100) of background network randomization in the standardization process. Users can also adjust randomization time of background network according to their own demands for the precision of network fingerprint.

NFP provides the `calc_sim_score` function for calculating the similarity score between two networks. As the similarity score is subjected to the size of the network, we use the maslov's method [4] to randomize a network while preserving the degree distribution. Then`nperm` parameter is added to `calc_sim_score` refers to the permutation times (the default is 100) of random network while calculating the similarity score mentioned above. We define a S4 class *NFP* in our package to store the calculation results of network fingerprints.

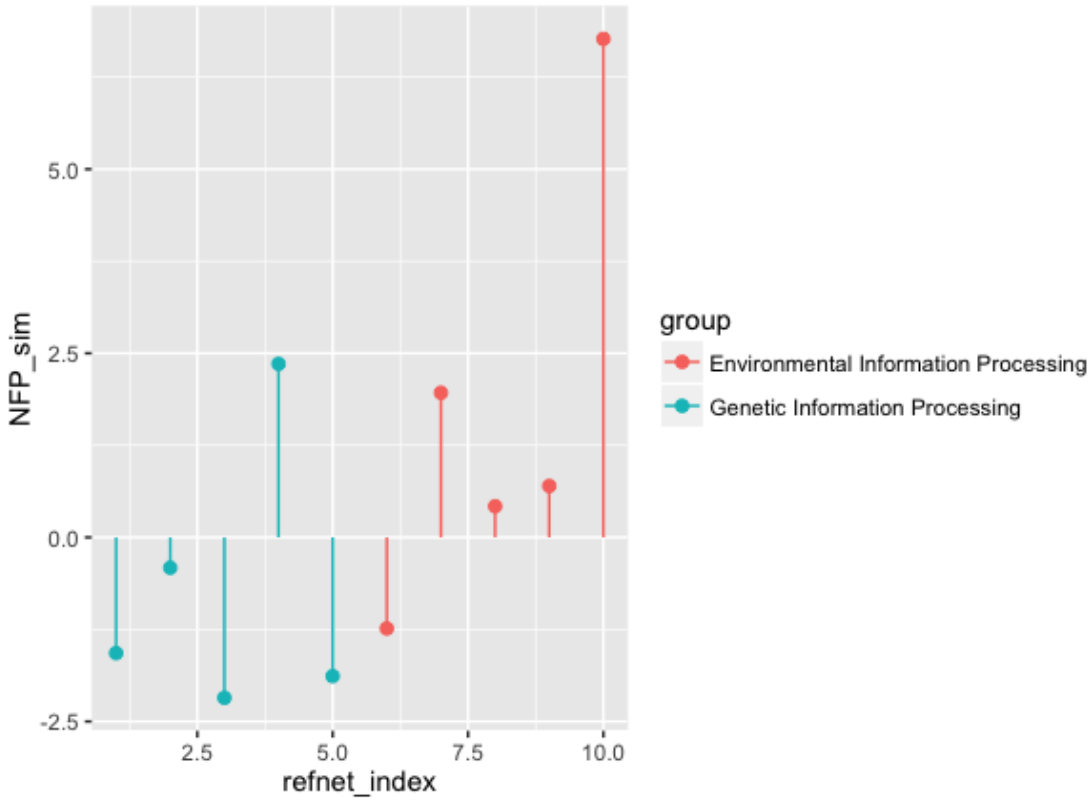
Simply, we choose the two pathway maps g as a query network, and a subset networks of `kegg_refnet` as the reference networks. Then the NFP can be calculated as following:

```
## set g as the query network
query_net <- g
## a subset of kegg_refnet, select the head five networks of group 1, 2
group_names <- group(kegg_refnet)$name
sample_NFPRefnet <- subnet(kegg_refnet, group_names[1:2],list(1:5,1:5))

## In order to save calculating time, we take nperm = 10
NFP_score <- lapply(query_net, calc_sim_score,NFPnet = sample_NFPRefnet,
  nperm = 10)

## methods of NFP class
show(NFP_score[[1]])
randomized_score  <- perm_score(NFP_score[[1]])
cluster <- cluster_info(NFP_score[[1]])
```

2.3 Network fingerprint visualization

NFP provides the `plot_NFP_NFP` function to visualize the network fingerprint of a single query network.

```
plot_NFP(NFP_score[[1]])
```



Session Information

The version number of R and packages loaded for generating the vignette were:

```
## R version 3.3.2 (2016-10-31)
## Platform: x86_64-apple-darwin13.4.0 (64-bit)
## Running under: macOS Sierra 10.12.1
##
## locale:
## [1] C/zh_CN.UTF-8/zh_CN.UTF-8/C/zh_CN.UTF-8/zh_CN.UTF-8
##
## attached base packages:
## [1] parallel  stats     graphics  grDevices utils     datasets  methods
## [8] base
##
## other attached packages:
## [1] graphite_1.18.0     NFP_0.99.1          graph_1.50.0
## [4] BiocGenerics_0.18.0 knitr_1.15
##
## loaded via a namespace (and not attached):
##  [1] Rcpp_0.12.7          highr_0.6            plyr_1.8.3
##  [4] tools_3.3.2          evaluate_0.10        RSQLite_1.0.0
##  [7] tibble_1.2-12        gtable_0.2.0         lattice_0.20-34
## [10] Matrix_1.2-7.1       KEGGgraph_1.30.0     igraph_1.0.1
## [13] DBI_0.5-1            apcluster_1.4.3      dplyr_0.5.0.9000
## [16] stringr_1.1.0.9000   S4Vectors_0.10.1     IRanges_2.6.1
```

```
## [19] rappdirs_0.3.1       stats4_3.3.2         grid_3.3.2
## [22] Biobase_2.32.0       R6_2.2.0             AnnotationDbi_1.34.3
## [25] XML_3.98-1.4         ggplot2_2.2.0        tidyr_0.6.0.9000
## [28] magrittr_1.5         scales_0.4.1         assertthat_0.1
## [31] BiocStyle_2.0.3      colorspace_1.2-6     stringi_1.1.2
## [34] lazyeval_0.2.0.9000  munsell_0.4.3
```

Cleanup

This is a cleanup step for the vignette on Windows; typically not needed for users.

```
allCon <- showConnections()
socketCon <- as.integer(rownames(allCon)[allCon[, "class"] == "sockconn"])
sapply(socketCon, function(ii) close.connection(getConnection(ii)) )
```

```
## list()
```

References

[1] Kanehisa M, Araki M, Goto S, et al. KEGG for linking genomes to life and the environment[J]. Nucleic acids research, 2008, 36(suppl 1): D480-D484.

[2] Frey B J, Dueck D. Clustering by passing messages between data points[J]. science, 2007, 315(5814): 972-976.

[3] Cui X, He H, He F, et al. Network fingerprint: a knowledge-based characterization of biomedical networks[J]. Scientific reports, 2015, 5.

[4] Maslov S, Sneppen K. Specificity and stability in topology of protein networks[J]. Science, 2002, 296(5569): 910-913.

[5] Ashburner M, Ball C A, Blake J A, et al. Gene Ontology: tool for the unification of biology[J]. Nature genetics, 2000, 25(1): 25-29.

[6] Sales G, Calura E, Cavalieri D, et al. graphite-a Bioconductor package to convert pathway topology to gene network[J]. BMC bioinformatics, 2012, 13(1): 20.