How to use the cisPath Package

Likun Wang

July 27, 2013

Institute of Systems Biomedicine, Peking University Health Science Center.

wanglk@hsc.pku.edu.cn

Contents

1 Introduction

This package is used to manage, visualize and edit protein–protein interaction (PPI) networks. Results can be shown visually and managed with different browsers across different platforms. The network is displayed as a `force-directed graph` (`http://bl.ocks.org/4062045`) with JavaScript library `D3` (`www.d3js.org`). Both mouse and touch screen users can view and edit the network graph easily. The HTML file follows `HTML 4.01 Strict` and `CSS version 3` standards to maintain consistency across different browsers. Chrome, Firefox, Safari, and IE9 will all properly display the PPI view. Please contact us if these paths do not display correctly.

2 Data

As an example, we have generated PPI data for several species from the PINA database (Cowley and et al., 2012; Wu and et al., 2009), the iRefIndex database(Razick and et al., 2008; Aranda and et al., 2011), and the STRING database (Szklarczyk and et al., 2011; Franceschini and et al., 2013). Users can download these files from `http://www.isb.pku.edu.cn/cisPath/`. If you make use of these files, please cite PINA, iRefIndex, or STRING accordingly. Users can edit the PPI data downloaded from these two databases, or combine them with their private data to construct more comprehensive PPI networks. In this introduction, we select only a small portion of the available PPI data as an example. An ID mapping file is also provided in this package, which was generated according to data from the UniProt database (UniProt Consortium and others, 2012). The following examples (section 4 below) will show how these files may be used.

3 Getting started

To load the *cisPath* package, type `library(cisPath)`. Five methods have been included in this package to format the files downloaded from the PINA, iRefIndex, and STRING databases. These methods are `getMappingFile`, `formatSTRINGPPI`, `formatPINAPPI`, `formatiRefIndex`, and `combinePPI`. Four other methods have been also provided that are used to manage, visualize, and edit PPI networks. These methods are `cisPath`, `addProteinNames`, `networkView`, and `easyEditor`.

4 Format PPI data

4.1 Format PPI data downloaded from the STRING database

The method `formatSTRINGPPI` is used to format the PPI file which is downloaded from the STRING database. Before formatting the PPI data, users should generate the identifier mapping file with the method `getMappingFile`. The input file for `getMappingFile` is downloaded from the UniProt (`http://www.uniprot.org/`) database. Please find the URLs for different species by typing `?getMappingFile`. The output file contains identifier mapping information which is necessary for the method `formatSTRINGPPI`. Each line contains both the Ensembl Genomes Protein identifier and the Swiss-Prot accession number for a given protein.

```
> library(cisPath)
> sprotFile <- system.file("extdata", "uniprot_sprot_human10.dat",
+                                              package="cisPath")
> mappingFile <- file.path(tempdir(), "mappingFile.txt")
> getMappingFile(sprotFile, output=mappingFile)

Processed lines: 1000
Processed lines: 2000
Processed lines: 3000
Processed lines: 4000
```

```
Processed lines: 4580
[1] "/tmp/Rtmp7siXTo/mappingFile.txt"
```

The input file for `formatSTRINGPPI` is downloaded from the STRING database (`http://string-db.org/`). The URL for this file is `http://string-db.org/newstring_download/protein.links.v9.05.txt.gz`.

```
>   STRINGPPI <- file.path(tempdir(), "STRINGPPI.txt")
>   fileFromSTRING <- system.file("extdata", "protein.links.txt",
+                                              package="cisPath")
>   formatSTRINGPPI(fileFromSTRING, mappingFile, "9606", output=STRINGPPI, 700)

Processing ID mapping file...

Processed lines: 10
Formatting PPI data...

Processed lines: 36
[1] "/tmp/Rtmp7siXTo/STRINGPPI.txt"
```

Each line of the output file contains Swiss-Prot accession numbers and gene names for two interacting proteins. An edge value has been estimated for each link between two interacting proteins. This value is defined as `max(1,log(1000-STRING_SCORE,100))`. This may be treated as the "cost" of determining the shortest paths between two proteins. Advanced users can edit the file and change the edge values for each edge.

4.2 Format PPI data downloaded from the PINA database

The input file is downloaded from the PINA database (`http://cbg.garvan.unsw.edu.au/pina/`). Access `http://cbg.garvan.unsw.edu.au/pina/interactome.stat.do` to download PPI files with the `MITAB format` for different species.

```
>   input <- system.file("extdata", "Homo_sapiens_PINA100.txt", package="cisPath")
>   PINAPPI <- file.path(tempdir(), "PINAPPI.txt")
>   formatPINAPPI(input, output=PINAPPI)

Correct MITAB format input

Processed 100 lines
Processed 100 lines
[1] "/tmp/Rtmp7siXTo/PINAPPI.txt"
```

Each line of the output file contains Swiss-Prot accession numbers and gene names for two interacting proteins. The edge value for each link between two interacting proteins is assigned as `1`. This may be treated as the "cost" of determining the shortest paths between two proteins. Advanced users can edit the file and change the edge values for each edge.

4.3 Format PPI data downloaded from the iRefIndex database

The input file is downloaded from the iRefIndex database (`http://irefindex.uio.no/wiki/iRefIndex`). Access `ftp://ftp.no.embnet.org/irefindex/data/archive/release_10.0/psi_mitab/MITAB2.6/` to download PPI files with the `MITAB2.6 format` for different species.

```
>   input <- system.file("extdata", "9606.mitab.100.txt", package="cisPath")
>   iRefIndex <- file.path(tempdir(), "iRefIndex.txt")
>   formatiRefIndex(input, output=iRefIndex)

Correct iRefIndex MITAB format

Processed 100 lines
Processed 101 lines
[1] "/tmp/Rtmp7siXTo/iRefIndex.txt"
```

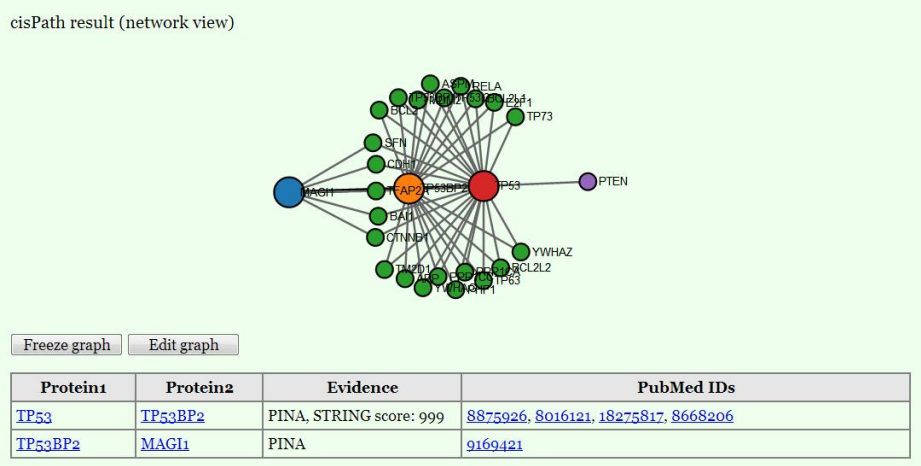
Each line of the output file contains Swiss-Prot accession numbers and gene names for two interacting proteins. The edge value for each link between two interacting proteins is assigned as `1`. This may be treated as the "cost" of determining the shortest paths between two proteins. Advanced users can edit the file and change the edge values for each edge.

4.4 Combine PPI data from different databases

The method `combinePPI` is used to combine the PPIs that were generated from different databases.

```
>   output <- file.path(tempdir(), "allPPI.txt")
>   combinePPI(c(PINAPPI, iRefIndex, STRINGPPI), output)

Output: /tmp/Rtmp7siXTo/allPPI.txt
Maximum edge value: -1.000
Processing file /tmp/Rtmp7siXTo/PINAPPI.txt

Processed 100 lines
Processed 101 lines
Processing file /tmp/Rtmp7siXTo/iRefIndex.txt

Processed 188 lines
Processing file /tmp/Rtmp7siXTo/STRINGPPI.txt

Processed 200 lines
Processed 220 lines
[1] "/tmp/Rtmp7siXTo/allPPI.txt"
```

We have processed the PPI data from the PINA, iRefIndex, and STRING databases. Users can access these files from our website `http://www.isb.pku.edu.cn/cispath/`.



Protein1	Protein2	Evidence	PubMed IDs
TP53	TP53BP2	PINA, STRING score: 999	8875926, 8016121, 18275817, 8668206
TP53BP2	MAGI1	PINA	9169421

Figure 1: networkView result

5 Examples

5.1 Example of the networkView method

This method is used to visualize the input proteins in the PPI network and display the evidence that supports the specific interactions among these proteins. If the PPI information downloaded from PINA and/or STRING is used, the PubMed IDs for the corresponding publications and/or STRING scores will also be shown. If the PubMed ID is provided, users can access the publication directly using the link. Occasionally, two given proteins cannot interact directly but their interaction can be mediated by scaffold proteins. This tool therefore also displays other proteins that can interact with at least two of the main proteins. In addition, using this method, users can choose the visual style (color and model size) in which the proteins will be displayed in the network.

```
>   infoFile <- system.file("extdata", "PPI_Info.txt", package="cisPath")
>   outputDir <- file.path(tempdir(), "networkView")
>   networkView(infoFile, c("MAGI1","TP53BP2","TP53", "PTEN"), outputDir,
+                                    FALSE, c(1,1,1,0), displayMore=TRUE)

Please wait patiently!
Processing input file...
input file: /tmp/Rtmp077v7B/Rinst4f132d9eeed4/cisPath/extdata/PPI_Info.txt
protein file: /tmp/Rtmp7siXTo/networkView/proteins.txt
output directory: /tmp/Rtmp7siXTo/networkView
Done.
[1] "/tmp/Rtmp7siXTo/networkView/graphView.html"
```

Figure 1 is the screenshot of the HTML file which will open automatically. In this example, the selected proteins are TP53, TP53BP2, MAGI1, and PTEN. The first three proteins are assigned as the main nodes and PTEN is treated as a leaf node. The three main

nodes are displayed as bigger circles filled in with default colors. The links between these main nodes are highlighted with bold lines. The leaf node (PTEN) will be displayed as a smaller circle in another default color. All node colors can be customized by the parameter `nodeColors` in this method. The other proteins which can interact with at least two given main proteins are presented as leaf nodes. The color of these leaf nodes can be customized by the parameter `leafColor`. The evidence which supports the functional interactions between these main proteins is shown in the table below the graph.

```
>   infoFile <- system.file("extdata", "PPI_Info.txt", package="cisPath")
>   outputDir <- file.path(tempdir(), "networkView2")
>   inputFile <- system.file("extdata", "networkView.txt", package="cisPath")
>   rt <- read.table(inputFile, sep=",", comment.char="", header=TRUE)
>   proteins <- as.vector(rt[,1])
>   sizes <- as.vector(rt[,2])
>   colors <- as.vector(rt[,3])
>   networkView(infoFile, proteins, outputDir, FALSE, sizes, colors, FALSE)

Please wait patiently!
Processing input file...
input file: /tmp/Rtmp077v7B/Rinst4f132d9eeed4/cisPath/extdata/PPI_Info.txt
protein file: /tmp/Rtmp7siXTo/networkView2/proteins.txt
output directory: /tmp/Rtmp7siXTo/networkView2
Done.
[1] "/tmp/Rtmp7siXTo/networkView2/graphView.html"
```

5.2 Example of the cisPath method

This method is used to identify and visualize the shortest functional paths between two given proteins in the PPI network. The file containing PPI information with edge cost for each two interacting proteins is used as input for this function. To identify the paths utilizing the shortest number of steps instead of reflecting minimal cost, the parameter `byStep` should be set as `TRUE`. In this situation, all the edge cost will be assigned as `1`. Setting this parameter as `TRUE` allows users to view more of the possible paths between two proteins.

Please see the file `PPI_Info.txt` in this package for the input file format. Taking the protein `TP53` as an example, input of the following codes in R will result in output of all the shortest paths to the target proteins `MAGI1` and `GH1`.

```
>   infoFile <- system.file("extdata", "PPI_Info.txt", package="cisPath")
>   outputDir <- file.path(tempdir(), "TP53_example")
>   results <- cisPath(infoFile, "TP53", outputDir, c("MAGI1", "GH1"),
+                                                        byStep=TRUE)

Please wait patiently!
Processing input file...
input file: /tmp/Rtmp077v7B/Rinst4f132d9eeed4/cisPath/extdata/PPI_Info.txt
source protein: TP53
```

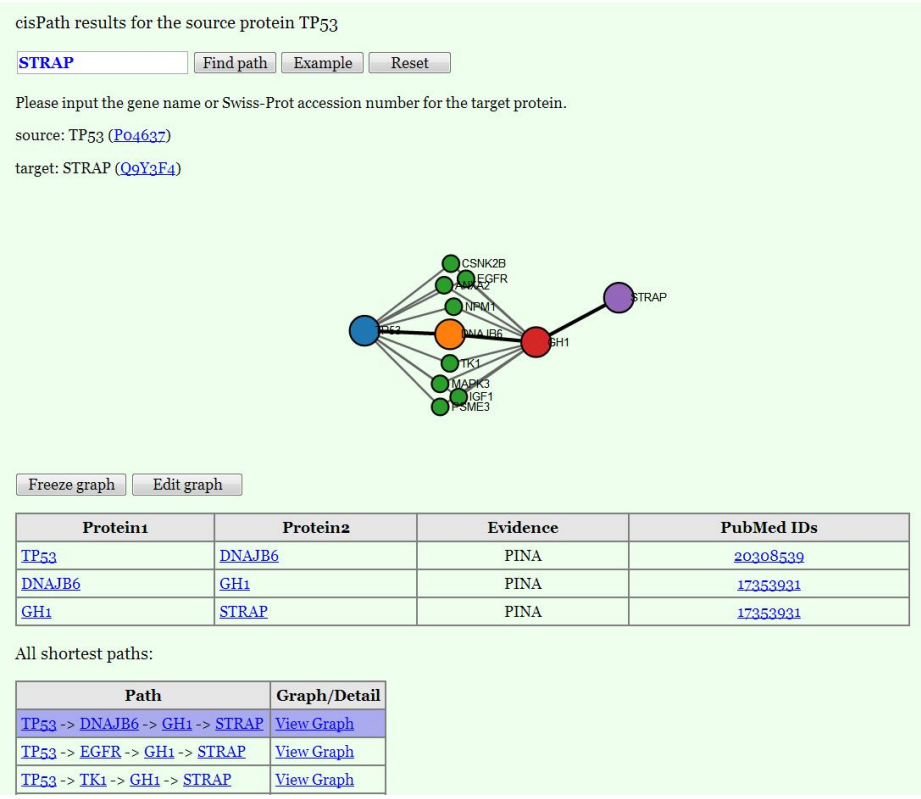
```
output directory: /tmp/Rtmp7siXTo/TP53_example
TP53: valid gene name
Swiss-Prot number: P04637
Searching ...
Number of proteins:1157

Number of processed proteins:0
Number of processed proteins:951

Output: 0
Output: 3
Done.
```

```
> results["MAGI1"]
```

```
$MAGI1
$MAGI1[[1]]
[1] "TP53:P04637"  "BAI1:O14514"  "MAGI1:Q96QZ7"

$MAGI1[[2]]
[1] "TP53:P04637"   "TFAP2A:P05549" "MAGI1:Q96QZ7"

$MAGI1[[3]]
[1] "TP53:P04637"  "CDH1:P12830"  "MAGI1:Q96QZ7"

$MAGI1[[4]]
[1] "TP53:P04637"  "SFN:P31947"   "MAGI1:Q96QZ7"

$MAGI1[[5]]
[1] "TP53:P04637"   "CTNNB1:P35222" "MAGI1:Q96QZ7"

$MAGI1[[6]]
[1] "TP53:P04637"    "TP53BP2:Q13625" "MAGI1:Q96QZ7"
```

```
> results["GH1"]
```

```
$GH1
$GH1[[1]]
[1] "TP53:P04637"   "DNAJB6:O75190" "GH1:P01241"

$GH1[[2]]
[1] "TP53:P04637" "EGFR:P00533" "GH1:P01241"

$GH1[[3]]
[1] "TP53:P04637" "TK1:P04183"  "GH1:P01241"
```



Protein1	Protein2	Evidence	PubMed IDs
TP53	DNAJB6	PINA	20308539
DNAJB6	GH1	PINA	17353931
GH1	STRAP	PINA	17353931

Path	Graph/Detail
TP53 -> DNAJB6 -> GH1 -> STRAP	View Graph
TP53 -> EGFR -> GH1 -> STRAP	View Graph
TP53 -> TK1 -> GH1 -> STRAP	View Graph

Figure 2: cisPath result

```
$GH1[[4]]
[1] "TP53:P04637" "IGF1:P05019" "GH1:P01241"

$GH1[[5]]
[1] "TP53:P04637" "NPM1:P06748" "GH1:P01241"

$GH1[[6]]
[1] "TP53:P04637"  "ANXA2:P07355" "GH1:P01241"

$GH1[[7]]
[1] "TP53:P04637"  "MAPK3:P27361" "GH1:P01241"

$GH1[[8]]
[1] "TP53:P04637"  "PSME3:P61289" "GH1:P01241"

$GH1[[9]]
[1] "TP53:P04637"   "CSNK2B:P67870" "GH1:P01241"
```

The HTML file will open automatically. Figure 2 is a screenshot of this file. Users can query the path to the target protein by input of the gene name or Swiss-Prot accession

number. All of the shortest paths will be shown in the table at the bottom.

Upon input, a randomly selected shortest path will be shown graphically. This shortest path will be highlighted by a bold black line. The other proteins which can interact with at least two proteins which lie on the shortest path are also displayed as leaf nodes. The colors of the main nodes and leaf nodes in this graph can be customized by the parameters `nodeColors` and `leafColor` respectively in the method `cisPath`. Users can change the view of this graph by dragging a node. When the `Release graph` button is clicked, the nodes will find their appropriate positions automatically. The evidence for the interactions between the proteins along the shortest path is shown in the table below the graph. By clicking the `View Graph` link in the table, users can select the path to be shown graphically.

In order to avoid inputting invalid target protein names, the unique identifier Swiss-Prot accession numbers may alternatively be used as input. The Swiss-Prot accession numbers can be sought from the UniProt (`http://www.uniprot.org/`) database.

```
>   infoFile <- system.file("extdata", "PPI_Info.txt", package="cisPath")
>   outputDir <- file.path(tempdir(), "TP53_example")
>   results <- cisPath(infoFile, "TP53", outputDir,
+                      targetProteins=c("Q96QZ7", "P01241"), swissProtID=TRUE)

Please wait patiently!
Processing input file...
input file: /tmp/Rtmp077v7B/Rinst4f132d9eeed4/cisPath/extdata/PPI_Info.txt
source protein: TP53
output directory: /tmp/Rtmp7siXTo/TP53_example
TP53: valid gene name
Swiss-Prot number: P04637
Searching ...
Number of proteins:1157

Number of processed proteins:0
Number of processed proteins:935

Output: 0
Output: 3
Done.

>   results["Q96QZ7"]

$Q96QZ7
$Q96QZ7[[1]]
[1] "TP53:P04637"  "BAI1:O14514"  "MAGI1:Q96QZ7"

$Q96QZ7[[2]]
[1] "TP53:P04637"   "TFAP2A:P05549" "MAGI1:Q96QZ7"

$Q96QZ7[[3]]
```

```
[1] "TP53:P04637"  "CDH1:P12830"  "MAGI1:Q96QZ7"

$Q96QZ7[[4]]
[1] "TP53:P04637"  "SFN:P31947"   "MAGI1:Q96QZ7"

$Q96QZ7[[5]]
[1] "TP53:P04637"    "TP53BP2:Q13625" "MAGI1:Q96QZ7"
```

```
> results["P01241"]
```

```
$P01241
$P01241[[1]]
[1] "TP53:P04637"   "DNAJB6:O75190" "GH1:P01241"

$P01241[[2]]
[1] "TP53:P04637" "NPM1:P06748" "GH1:P01241"

$P01241[[3]]
[1] "TP53:P04637"  "MAPK3:P27361" "GH1:P01241"

$P01241[[4]]
[1] "TP53:P04637"  "PSME3:P61289" "GH1:P01241"

$P01241[[5]]
[1] "TP53:P04637"   "CSNK2B:P67870" "GH1:P01241"
```

This method may be run without input of target proteins. If target proteins are not provided, this method will identify the shortest paths from the source protein to all the other relevant proteins. Users can thus query the results with a browser upon finding an interesting "target" protein without launching R.

```
> infoFile <- system.file("extdata", "PPI_Info.txt", package="cisPath")
> outputDir <- file.path(tempdir(), "TP53_example")
> results <- cisPath(infoFile, "TP53", outputDir)
> results["GH1"]
> results["P01241"]
```

A protein often has several names, and some of these names have perhaps not been included in the input file `PPI_Info.txt`. We therefore suggest users take a look on the output file `targetIDs.txt` to check whether the input target protein names are valid. The parameter `name2IDFile` allows users to add ID mapping information. In this way, users can search for the shortest paths using the protein names with which they are familiar.

```
> infoFile <- system.file("extdata", "PPI_Info.txt", package="cisPath")
> outputDir <- file.path(tempdir(), "TP53_example")
> name2protFile <- system.file("extdata", "name2prot.txt", package="cisPath")
> results <- cisPath(infoFile, "P04637", outputDir, name2IDFile=name2protFile)
```

5.3 Example of the addProteinNames method

This method allows users to add more ID mapping information even after the shortest paths have been identified. For example, the following codes will add all ID mapping information included in the file `name2prot.txt`.

```
>   name2protFile <- system.file("extdata", "name2prot.txt", package="cisPath")
>   addProteinNames(name2protFile, outputDir)
```

```
[1] "/tmp/Rtmp7siXTo/TP53_example"
```

5.4 Example of downloading PPI data from our website

The following codes will first download the PPI data that was generated for this example, and then will identify the shortest paths from `TP53` to all other proteins. With combined PPI data from PINA, iRefIndex, and STRING, the output from a single source protein is about `1.5G` in size. Although the output in such cases is large, we strongly suggest users give this method of selecting a source but not a target a try.

```
>   outputDir <- "/home/user/TP53"
>   # Create the output directory
>   dir.create(outputDir, showWarnings = FALSE, recursive = TRUE)
>   # infoFile: site where the PPI data file will be saved.
>   infoFile <- file.path(outputDir, "Homo_sapiens_PPI.txt")
>   # Download PPI data
>   url <- "http://www.isb.pku.edu.cn/cispath/data/Homo_sapiens_PPI.txt"
>   download.file(url, infoFile)
>   results <- cisPath(infoFile, "TP53", outputDir, byStep=TRUE)
>   url<-"http://www.isb.pku.edu.cn/cispath/data/Homo_sapiens_PPI.txt"
>   url<-"http://www.isb.pku.edu.cn/cispath/data/Caenorhabditis_elegans_PPI.txt"
>   url<-"http://www.isb.pku.edu.cn/cispath/data/Drosophila_melanogaster_PPI.txt"
>   url<-"http://www.isb.pku.edu.cn/cispath/data/Mus_musculus_PPI.txt"
>   url<-"http://www.isb.pku.edu.cn/cispath/data/Rattus_norvegicus_PPI.txt"
>   url<-"http://www.isb.pku.edu.cn/cispath/data/Saccharomyces_cerevisiae_PPI.txt"
```

5.5 Example of the easyEditor method

The output HTML file of this method is a editor for network graphs. Users can draw and edit their own network with this editor.

```
>   outputDir <- file.path(tempdir(), "easyEditor")
>   easyEditor(outputDir)
```

```
Please open the output HTML file using a browser!
[1] "/tmp/Rtmp7siXTo/easyEditor/easyEditor.html"
```

References

Aranda, B. and et al. (2011). Psicquic and psiscore: accessing and scoring molecular interactions. *Nat Methods*, 8:D528–D529.

Cowley, M. and et al. (2012). PINA v2.0: mining interactome modules. *Nucleic Acids Res*, 40:D862–865.

Franceschini, A. and et al. (2013). String v9.1: protein-protein interaction networks, with increased coverage and integration. *Nucleic Acids Res*, 41:D808–D815.

Razick, S. and et al. (2008). irefindex: A consolidated protein interaction database with provenance. *BMC Bioinformatics*, 9:D405.

Szklarczyk, D. and et al. (2011). The string database in 2011: functional interaction networks of proteins, globally integrated and scored. *Nucleic Acids Res*, 39:D561–D568.

UniProt Consortium and others (2012). Reorganizing the protein space at the universal protein resource (uniprot). *Nucleic Acids Res*, 40:D71–D75.

Wu, J. and et al. (2009). Integrated network analysis platform for protein-protein interactions. *Nature methods*, 6:75–77.