# STRINGdb Package Vignette

Andrea Franceschini

15 March 2015

## 1 INTRODUCTION

STRING (https://www.string-db.org) is a database of known and predicted protein-protein interactions. The interactions include direct (physical) and indirect (functional) associations. The database contains information from numerous sources, including experimental repositories, computational prediction methods and public text collections. Each interaction is associated with a combined confidence score that integrates the various evidences. We currently cover over 24 milions proteins from 5090 organisms.

As you will learn in this guide, the STRING database can be usefull to add meaning to list of genes (e.g. the best hits coming out from a screen or the most differentially expressed genes coming out from a Microarray/RNAseq experiment.)

We provide the STRINGdb R package in order to facilitate our users in accessing the STRING database from R. In this guide we explain, with examples, most of the package's features and functionalities.

In the STRINGdb R package we use the new ReferenceClasses of R (search for "ReferenceClasses" in the R documentation.). Besides we make use of the iGraph package (http://igraph.sourceforge.net) as a data structure to represent our protein-protein interaction network.

To begin, you should first know the NCBI taxonomy identifiers of the organism on which you have performed the experiment (e.g. 9606 for Human, 10090 for mouse). If you don't know that, you can search the NCBI Taxonomy (http://www.ncbi.nlm.nih.gov/taxonomy) or start looking at our species table (that you can also use to verify that your organism is represented in the STRING database).
Hence, if your species is not Human (i.e. our default species), you can find it and their taxonomy identifiers on STRING webpage under the 'organisms' section (https://string-db.org/cgi/input.pl?input_page_active_form=org
or download the full list in the download section of STRING website.

```
> library(STRINGdb)
> string_db <- STRINGdb$new( version="11.5", species=9606,
+                            score_threshold=200, network_type="functional", input_directory="")
```

```
WARNING: Only 'full' and 'physical' network types are valid. Setting to the network type to 'full' ST
```

As it has been shown in the above commands, you start instantiating the STRINGdb reference class. In the constructor of the class you can also define the STRING version to be used and a threshold for the combined scores of the interactions, such that any interaction below that threshold is not loaded in the object (by default the score threshold is set to 400).
You can also specify the network type "functional" for full functional STRING network or "physical" for physical subnetwork, which link only the proteins which share a physical complex.
Besides, if you specify a local directory to the parameter input-directory, the database files will be downloaded into this directory and most of the methods can be used off-line. Otherwise, the database files will be saved and cached in a temporary directory that will be cleaned automatically when the R session is closed.

For a better understanding of the package two other commands can be useful:

```
> STRINGdb$methods()              # To list all the methods available.

 [1] ".objectPackage"                        ".objectParent"
 [3] "add_diff_exp_color"                    "add_proteins_description"
 [5] "benchmark_ppi"                         "benchmark_ppi_pathway_view"
 [7] "callSuper"                             "copy"
 [9] "enrichment_heatmap"                    "export"
[11] "field"                                 "getClass"
[13] "getRefClass"                           "get_aliases"
[15] "get_annotations"                       "get_bioc_graph"
[17] "get_clusters"                          "get_enrichment"
[19] "get_graph"                             "get_homologs"
[21] "get_homologs_besthits"                 "get_homology_graph"
[23] "get_interactions"                      "get_link"
[25] "get_neighbors"                         "get_paralogs"
[27] "get_pathways_benchmarking_blackList"   "get_png"
[29] "get_ppi_enrichment"                    "get_ppi_enrichment_full"
[31] "get_proteins"                          "get_pubmed"
[33] "get_pubmed_interaction"                "get_subnetwork"
[35] "get_summary"                           "get_term_proteins"
[37] "import"                                "initFields"
[39] "initialize"                            "load"
[41] "load_all"                              "map"
[43] "mp"                                    "plot_network"
[45] "plot_ppi_enrichment"                   "post_payload"
[47] "ppi_enrichment"                        "remove_homologous_interactions"
[49] "set_background"                        "show"
[51] "show#envRefClass"                      "trace"
[53] "untrace"                               "usingMethods"


> STRINGdb$help("get_graph")      # To visualize their documentation.

Call:
$get_graph()
```

```
Description:
  Return an igraph object with the entire STRING network.
  We invite the user to use the functions of the iGraph package to conveniently
  search/analyze the network.

References:
  Csardi G, Nepusz T: The igraph software package for complex network research,
  InterJournal, Complex Systems 1695. 2006.
  http://igraph.sf.net

See Also:
  In order to simplify the most common tasks, we do also provide convenient functions
  that wrap some iGraph functions.
  get_interactions(string_ids)   # returns the interactions in between the input proteins
  get_neighbors(string_ids)      # Get the neighborhoods of a protein (or of a vector of proteins).
  get_subnetwork(string_ids)     # returns a subgraph from the given input proteins

Author(s):
   Andrea Franceschini
```

For all the methods that we are going to explain below, you can always use the help function in order to get additional information/parameters with respect to those explained in this guide.

As an example, we use the analyzed data of a microarray study taken from GEO (Gene Expression Omnibus, GSE9008). This study investigates the activity of Resveratrol, a natural phytoestrogen found in red wine and a variety of plants, in A549 lung cancer cells. Microarray gene expression profiling after 48 hours exposure to Revestarol has been performed and compared to a control composed by A549 lung cancer cells threated only with ethanol. This data is already analyzed for differential expression using the limma package: the genes are sorted by fdr corrected pvalues and the log fold change of the differential expression is also reported in the table.

```
> data(diff_exp_example1)
> head(diff_exp_example1)

     pvalue    logFC         gene
1 0.0001018 3.333461       VSTM2L
2 0.0001392 3.822383       TBC1D2
3 0.0001720 3.306056        LENG9
4 0.0001739 3.024605       TMEM27
5 0.0001990 3.854414 LOC100506014
6 0.0002393 3.082052       TSPAN1
```

As a first step, we map the gene names to the STRING database identifiers using the "map" method. In this particular example, we map from gene HUGO names, but our mapping function supports several other common identifiers (e.g. Entrez GeneID, ENSEMBL proteins, RefSeq transcripts ... etc.).

The map function adds an additional column with STRING identifiers to the dataframe that is passed as first parameter.

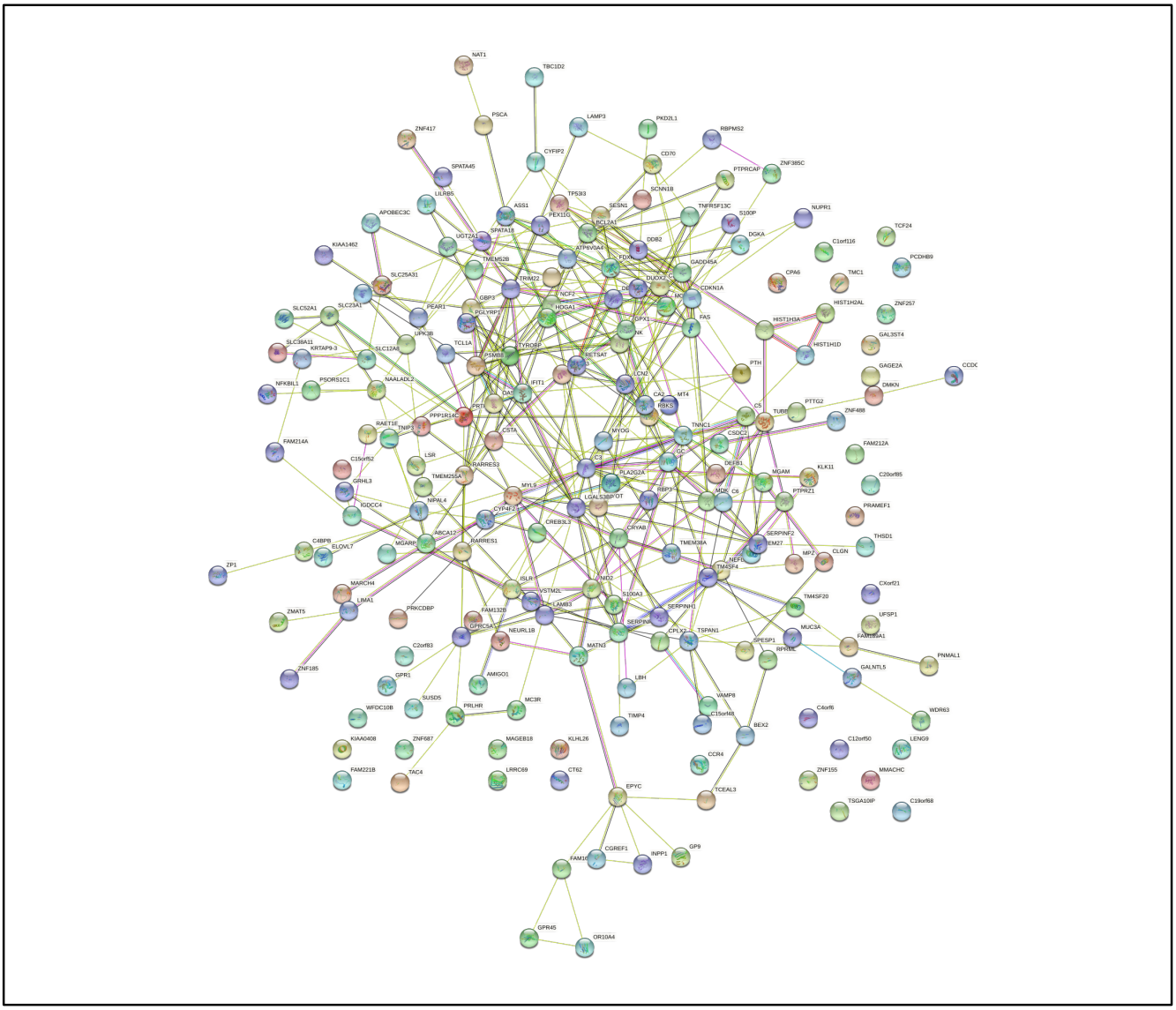
```
> example1_mapped <- string_db$map( diff_exp_example1, "gene", removeUnmappedRows = TRUE )

Warning:  we couldn't map to STRING 15% of your identifiers
```

As you may have noticed, the previous command prints a warning showing the number of genes that we failed to map. In this particular example, we cannot map all the probes of the microarray that refer to position of the chromosome that are not assigned to a real gene (i.e. all the LOC genes). If we remove all these LOC genes before the mapping we obtain a much lower percentage of unmapped genes (i.e. $< 6$ %).
If you set to FALSE the "removeUnmappedRows" parameter, than the rows which corresponds to unmapped genes are left and you can manually inspect them.
Finally, we extract the most significant 200 genes and we produce an image of the STRING network for those. The image shows clearly the genes and how they are possibly functionally related. On the top of the plot, we insert a pvalue that represents the probability that you can expect such an equal or greater number of interactions by chance.

```
> hits <- example1_mapped$STRING_id[1:200]
```

```
> string_db$plot_network( hits )
```

proteins: 200
interactions: 382
expected interactions: 229 (p−value: 0)



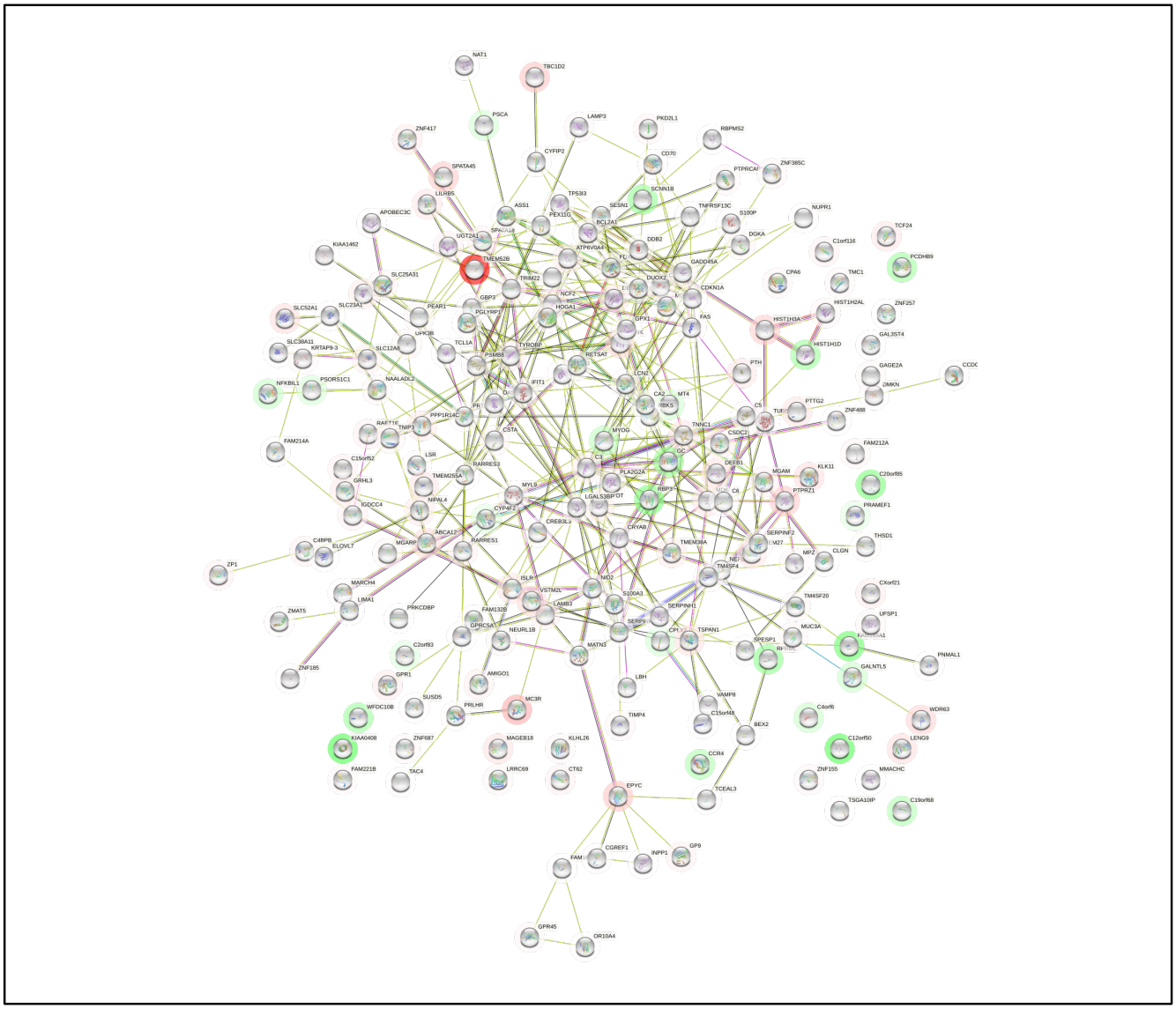
## 2 PAYLOAD MECHANISM

This R library provides the ability to interact with the STRING payload mechanism. The payload appears as an additional colored "halo" around the bubbles.

For example, this allows to color in green the genes that are down-regulated and in red the genes that are up-regulated. For this mechanism to work, we provide a function that posts the information on our web server.

```
> # filter by p-value and add a color column
> # (i.e. green down-regulated gened and red for up-regulated genes)
> example1_mapped_pval05 <- string_db$add_diff_exp_color( subset(example1_mapped, pvalue<0.05),
+                                                         logFcColStr="logFC" )

> # post payload information to the STRING server
> payload_id <- string_db$post_payload( example1_mapped_pval05$STRING_id,
+                                       colors=example1_mapped_pval05$color )

> # display a STRING network png with the "halo"
> string_db$plot_network( hits, payload_id=payload_id )
```

proteins: 200
interactions: 382
expected interactions: 229 (p−value: 0)



# 3 ENRICHMENT

We provide a method to compute the enrichment in Gene Ontology (Process, Function and Component), KEGG and Reactome pathways, PubMed publications, UniProt Keywords, and PFAM/INTERPRO/SMART domains for your set of proteins all in one simple call. The enrichment itself is computed using an hypergeometric test and the FDR is calculated using Benjamini-Hochberg procedure.

```
> enrichment <- string_db$get_enrichment( hits )
> head(enrichment, n=20)
```

```
       category        term number_of_genes number_of_genes_in_background
1       Process  GO:0006952              34                           1296
2       Process  GO:0010951              12                            248
3       Process  GO:0051707              31                           1256
4     Component  GO:0005576              66                           4166
5     Component  GO:0005615              55                           3195
6     Component  GO:0070062              41                           2099
7     Component  GO:1903561              42                           2121
8       TISSUES BTO:0004850              10                            170
9       Keyword     KW-0732              57                           3233
10      Keyword     KW-0391              17                            522
11      Keyword     KW-0964              37                           1818
12         KEGG    hsa04115               6                             72
13 WikiPathways      WP4963               8                             67
   ncbiTaxonId
1         9606
2         9606
3         9606
4         9606
5         9606
6         9606
7         9606
8         9606
9         9606
10        9606
11        9606
12        9606
13        9606

1
2
3
4  9606.ENSP00000008938,9606.ENSP00000014914,9606.ENSP00000187762,9606.ENSP00000216286,9606.ENSP00000
5
6
7
8
9
10
11
12
13
```

```
1
2
3
4  PGLYRP1,GPRC5A,TMEM38A,NID2,C5,RARRES1,C4BPB,CD70,C3,ISLR,SERPINF1,THSD1,EPYC,LGALS3BP,C6,VAMP8,CS
5                                                                                PGLYRP1,GPRC5A,TMEM38A,NID2,C5,RAR
6
7
8
9                                                                                  PGLYRP1,NID2,C5,C4BPB,C3,ISLR,SE
10
11
12
13
    p_value     fdr                                      description
1  5.07e-07 0.00650                                  Defense response
2  1.43e-05 0.03780 Negative regulation of endopeptidase activity
3  5.89e-06 0.03780                      Response to other organism
4  9.03e-05 0.04080                            Extracellular region
5  5.17e-05 0.04080                             Extracellular space
6  4.18e-05 0.04080                           Extracellular exosome
7  2.40e-05 0.04080                           Extracellular vesicle
8  1.54e-05 0.02940                                Bone marrow cell
9  1.78e-05 0.01200                                         Signal
10 3.64e-05 0.01230                                       Immunity
11 4.61e-05 0.01230                                       Secreted
12 1.40e-04 0.04690                           p53 signaling pathway
13 9.02e-07 0.00061                 p53 transcriptional gene network
```

If you have performed your experiment on a predefined set of proteins, it is important to run the enrichment statistics using that set as a background (otherwise you would get a wrong p-value !). Hence, before to launch the method above, you may want to set the background:

```
> backgroundV <- example1_mapped$STRING_id[1:2000]   # as an example, we use the first 2000 genes
> string_db$set_background(backgroundV)
```

You can also set the background when you instantiate the STRINGdb object:

```
> string_db <- STRINGdb$new( score_threshold=200, backgroundV = backgroundV )
```

If you just want to know terms are assigned to your set of proteins (and not necessary enriched) you can use "get_annotations" method. This method will output all the terms from most of the categories (the exceptions are KEGG terms due to licensing issues and PubMed due to the size of the output) that are associated with your set of proteins.

```
> annotations <- string_db$get_annotations( hits )
> head(annotations, n=20)
```

```
       category      term_id number_of_genes ratio_in_set species
1  COMPARTMENTS GOCC:0000109               1        0.005    9606
2  COMPARTMENTS GOCC:0000139               2        0.010    9606
3  COMPARTMENTS GOCC:0000151               1        0.005    9606
4  COMPARTMENTS GOCC:0000228               1        0.005    9606
5  COMPARTMENTS GOCC:0000307               1        0.005    9606
6  COMPARTMENTS GOCC:0000323               6        0.030    9606
7  COMPARTMENTS GOCC:0000502               1        0.005    9606
8  COMPARTMENTS GOCC:0000785               2        0.010    9606
9  COMPARTMENTS GOCC:0000786               1        0.005    9606
10 COMPARTMENTS GOCC:0000791               1        0.005    9606
11 COMPARTMENTS GOCC:0001533               1        0.005    9606
12 COMPARTMENTS GOCC:0001650               1        0.005    9606
13 COMPARTMENTS GOCC:0001669               1        0.005    9606
14 COMPARTMENTS GOCC:0001725               1        0.005    9606
15 COMPARTMENTS GOCC:0001726               2        0.010    9606
16 COMPARTMENTS GOCC:0002133               1        0.005    9606
17 COMPARTMENTS GOCC:0005576              40        0.200    9606
18 COMPARTMENTS GOCC:0005577               1        0.005    9606
19 COMPARTMENTS GOCC:0005579               3        0.015    9606
20 COMPARTMENTS GOCC:0005604               2        0.010    9606

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17 9606.ENSP00000008938,9606.ENSP00000216286,9606.ENSP00000223642,9606.ENSP00000237696,9606.ENSP00000
18
19
20

1
2
3
4
5
6
7
8
```

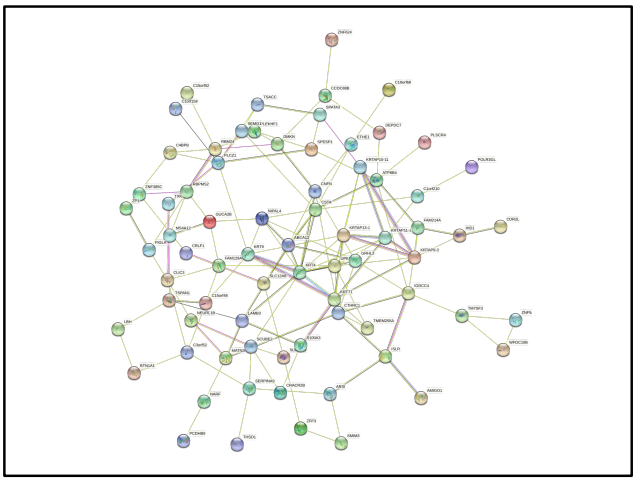
```
9
10
11
12
13
14
15
16
17 PGLYRP1,NID2,C5,RARRES1,C4BPB,CD70,C3,SERPINF1,THSD1,LGALS3BP,C6,CSTA,ANXA3,PTH,CA2,TIMP4,DEFB1,PS
18
19
20
                                      description
1                  Nucleotide-excision repair complex
2                                    Golgi membrane
3                            Ubiquitin ligase complex
4                                  Nuclear chromosome
5  Cyclin-dependent protein kinase holoenzyme complex
6                                       Lytic vacuole
7                                  Proteasome complex
8                                           Chromatin
9                                          Nucleosome
10                                        Euchromatin
11                                 Cornified envelope
12                                   Fibrillar center
13                                  Acrosomal vesicle
14                                       Stress fiber
15                                             Ruffle
16                                 Polycystin complex
17                               Extracellular region
18                                 Fibrinogen complex
19                             Membrane attack complex
20                                   Basement membrane
```

# 4 CLUSTERING

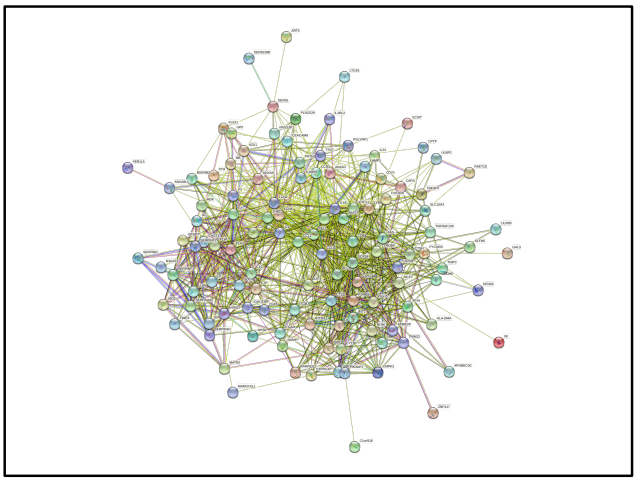
The iGraph package provides several clustering/community algorithms: "fastgreedy", "walktrap", "spinglass", "edge.betweenness". We encapsulate this in an easy-to-use function that returns the clusters in a list.

```
> # get clusters
> clustersList <- string_db$get_clusters(example1_mapped$STRING_id[1:600])

> # plot first 4 clusters
> par(mfrow=c(2,2))
> for(i in seq(1:4)){
+  string_db$plot_network(clustersList[[i]])
+ }
```

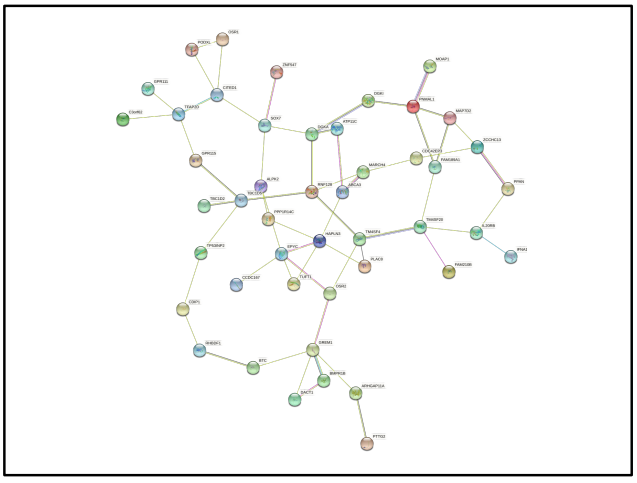
proteins: 74
interactions: 137
expected interactions: 13 (p−value: 0)



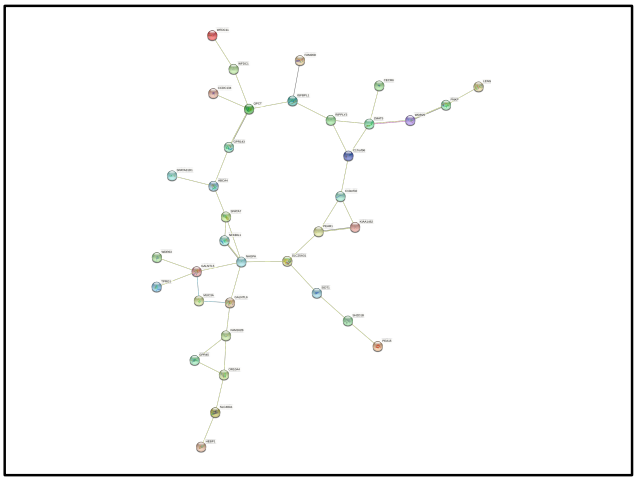
proteins: 119
interactions: 934
expected interactions: 175 (p−value: 0)



proteins: 46
interactions: 59
expected interactions: 8 (p−value: 0)



proteins: 36
interactions: 41
expected interactions: 3 (p−value: 0)



# 5 ADDITIONAL PROTEIN INFORMATION

You can get a table that contains all the proteins that are present in our database of the species of interest. The protein table also include the preferred name, the size and a short description of each protein.

```
> string_proteins <- string_db$get_proteins()
```

In the following section we will show how to query STRING with R on some specific proteins. In the examples, we will use the famous tumor proteins TP53 and ATM .

First we need to get the STRING identifier of those proteins, using our mp method:

```
> tp53 = string_db$mp( "tp53" )
> atm = string_db$mp( "atm" )
```

The mp method (i.e. map proteins) is an alternative to our map method, to be used when you need to map only one or few proteins.
It takes in input a vector of protein aliases and returns a vector with the STRING identifiers of those proteins.

Using the following method, you can see the proteins that interact with one or more of your proteins:

```
> string_db$get_neighbors( c(tp53, atm) )
```

It is also possible to retrieve the interactions that connect certain input proteins between each other. Using the "get_interactions" method we can clearly see that TP53 and ATM interact with each other with a good evidence/score.

```
> string_db$get_interactions( c(tp53, atm) )
```

```
                  from                   to combined_score
1 9606.ENSP00000269305 9606.ENSP00000278616            999
2 9606.ENSP00000269305 9606.ENSP00000278616            999
```

STRING provides a way to get homologous proteins: in our database we store ALL-AGAINST-ALL alignments within all 5090 organisms. You can retrive all of the paralogs of the protein using "get_paralogs" method.

```
> # Get all homologs of TP53 in human.
> string_db$get_paralogs(tp53)
```

STRING also stores best hits (as measured by bitscore) between the proteins from different species. "get_homologs_besthits" lets you retrieve these homologs.

```
> # get the best hits of the following protein in all the STRING species
> string_db$get_homologs_besthits(tp53)
```

... or you can specify the species of interest (i.e. all the blast hits):

```
> # get the homologs of the following two proteins in the mouse (i.e. species_id=10090)
> string_db$get_homologs_besthits(c(tp53, atm), target_species_id=10090, bitscore_threshold=60)
```

# 6 CITATION

Please cite:

Szklarczyk D, Gable AL, Nastou KC, Lyon D, Kirsch R, Pyysalo S, Doncheva NT, Legeay M, Fang T, Bork P, Jensen LJ, von Mering C. 'The STRING database in 2021: customizable protein-protein networks, and functional characterization of user-uploaded gene/measurement sets.' Nucleic Acids Res. 2021 Jan 8;49(D1):D605-12