Package 'iRefR'

July 2, 2014

Version 1.13

Date 2013-12-17

Title iRefIndex Manager

Description

``iRefR" allows the user to load any version of the consolidated protein interaction database ``iRefIndex" and perform tasks such as: selecting databases, pmids, experimental methods, searching for specific proteins, separate binary interactions from complexes and polymers, generate complexes according to an algorithm that looks after possible binary-represented complexes, make general database statistics and create network graphs, among others.

Author Antonio Mora <antoniocmora@gmail.com>, Ian Donaldson <ian.oslo@gmail.com>

Maintainer Antonio Mora <antoniocmora@gmail.com>

Depends R (>= 2.12.1), igraph (>= 0.6), graph, RBGL

License GPL (>= 2)

URL http://irefindex.org/wiki/index.php?title=iRefR

BuildVignettes FALSE

NeedsCompilation no

Repository CRAN

Date/Publication 2013-12-17 19:35:08

R topics documented:

`convert_complexList_to_MITAB`

Convert complexList format to MITAB format

Description

Convert a table from complexList format to MITAB format.

The "complexList" format is one of the three table formats used by "iRefR", together with the MITAB format and the edgeList. The "complexList" format is a table with two columns, where the first column corresponds to an identifier and the second column to a group of proteins, commonly a complex, written as a comma-separated string. This representation has less information than MITAB but simplifies the visualization of groups of proteins, such as complexes.

On iROGs and icROGs: "iRefIndex" guarantees the non-redundancy of protein information by assigning a different protein identifier (called Redundant Object Group, ROG) to every different protein sequence. This is called the "non-canonical" representation. At the same time, groups of non-redundant proteins might be different isoforms of a given protein, and, in this case, the identifier of one protein of the group (called cROG or canonical ROG) is chosen to represent the entire group of similar proteins. This is called the "canonical" representation of proteins. The iROG and icROG IDs correspond to integer representations of the ROG and cROG, respectively. Most of the "iRefR" functions work with iROGs and icROGs.

Usage

```
convert_complexList_to_MITAB(complexList, MITAB_table, canonical_rep,
include_generated_complexes)
```

Arguments

`complexList`	iRefIndex/complexList R table.
`MITAB_table`	Full iRefIndex/MITAB R table for a given organism.
`canonical_rep`	Either "yes" for working with the canonical representation (icROGs) or "no" for working with the non-canonical representation (iROGs). Default="yes".

include_generated_complexes
: Parameter to include the set of complexes generated from binary-represented interactions that might be misrepresented complexes, according to an "iRefR" algorithm that searches for groups of proteins coming from co-precipitation experiments, sharing the same PMID, source database and bait (when baits are available). Such set of complexes might be in the complexList only if the user generated them using "convert_MITAB_to_complexList", and can be easily identified by their complex ID constructed from their source database, PMID and experimental method names. If it is set to "yes", the user will get the original binary interactions these complexes were generated from. Default="no".

Value

MITAB_table
: iRefIndex/MITAB R table corresponding to the original complexList table.

Author(s)

Antonio Mora <antoniocmora@gmail.com>

Examples

```
    ## Not run:
    ## get tables
    irefindex_curr_ecoli = get_irefindex("562", "13.0", tempdir())
    iRef_binary = select_interaction_type("binary", irefindex_curr_ecoli)
    iRef_complex = select_interaction_type("complex", irefindex_curr_ecoli)
    known_complexes_ecoli_complexList = convert_MITAB_to_complexList(iRef_complex,
canonical_rep="yes", include_generated_complexes="no", list_methods="default")

    ## execute function
    reconstructed_known_ecoli_MITAB = convert_complexList_to_MITAB(
known_complexes_ecoli_complexList, irefindex_curr_ecoli, "yes", "no")
    setequal(dim(iRef_complex), dim(reconstructed_known_ecoli_MITAB))

    ## more examples
    generated_complexes_ecoli_complexList = convert_MITAB_to_complexList(iRef_binary,
canonical_rep="yes", include_generated_complexes="yes", list_methods="default")
    reconstructed_generated_ecoli_MITAB = convert_complexList_to_MITAB(
generated_complexes_ecoli_complexList, irefindex_curr_ecoli, "yes", "yes")
    reconstructed_complexList = convert_MITAB_to_complexList(
reconstructed_generated_ecoli_MITAB, canonical_rep="yes", include_generated_complexes=
"yes")
   setequal(dim(generated_complexes_ecoli_complexList), dim(reconstructed_complexList))

   all_complexes_ecoli_complexList = convert_MITAB_to_complexList(irefindex_curr_ecoli,
canonical_rep="yes", include_generated_complexes="yes", list_methods="default")
    reconstructed_all_ecoli_MITAB = convert_complexList_to_MITAB(
all_complexes_ecoli_complexList, irefindex_curr_ecoli, "yes", "yes")
    reconstructed_all_complexList = convert_MITAB_to_complexList(
reconstructed_all_ecoli_MITAB, canonical_rep="yes", include_generated_complexes="yes")
    setequal(dim(all_complexes_ecoli_complexList), dim(reconstructed_all_complexList))
```

```
## End(Not run)
```

---

convert_edgeList_to_graph *Convert iRefIndex/edgeList table to Graph*

Description

Create R graphical objects of an iRefIndex/edgeList interaction table.

Usage

```
convert_edgeList_to_graph(edgeList, directionality, graphical_package)
```

Arguments

edgeList	iRefIndex/edgeList R table.
directionality	Either "directed" for directed graphs or "undirected" for undirected graphs. The chosen value must be the same used to generate the edgeList table. Default="undirected".
graphical_package	Either "graph" or "igraph", depending on the R package the user wants to use. If you use "graph", no loop edges will be allowed. Default="igraph".

Value

output	R graphical object corresponding to the original edgeList.

Author(s)

Antonio Mora <antoniocmora@gmail.com>

Examples

```
    ## get tables
    irefindex_curr_ecoli = get_irefindex("562", "13.0", tempdir())
    all_INTACT = select_database("intact", irefindex_curr_ecoli)
    binary_INTACT = select_interaction_type("binary", all_INTACT)
    complex_INTACT = select_interaction_type("complex", all_INTACT)

    edgeList_binary_INTACT = convert_MITAB_to_edgeList(binary_INTACT)
    edgeList_complex_INTACT_s = convert_MITAB_to_edgeList(complex_INTACT, "default",
"spoke")
    edgeList_complex_INTACT_m = convert_MITAB_to_edgeList(complex_INTACT, "default",
"matrix")
    edgeList_all_INTACT = convert_MITAB_to_edgeList(all_INTACT, "default", "spoke")
    edgeList_binary_INTACT_dir = convert_MITAB_to_edgeList(binary_INTACT, "default",
```

```
"bipartite", "yes", "directed")
    edgeList_complex_INTACT_sdir = convert_MITAB_to_edgeList(complex_INTACT, "default",
"spoke", "yes", "directed")

    ## execute function
    graph_binary_INTACT = convert_edgeList_to_graph(edgeList_binary_INTACT,
"undirected", "igraph")
    graph_complex_INTACT_s = convert_edgeList_to_graph(edgeList_complex_INTACT_s)
    graph_complex_INTACT_m = convert_edgeList_to_graph(edgeList_complex_INTACT_m)

    ## Not run:
    graph_all_INTACT = convert_edgeList_to_graph(edgeList_all_INTACT)

    graph_binary_INTACT_dir = convert_edgeList_to_graph(edgeList_binary_INTACT_dir,
"directed")
    graph_complex_INTACT_sdir = convert_edgeList_to_graph(edgeList_complex_INTACT_sdir,
"directed")
    # For the binary case, edge number is smaller than the undirected case because only
    # interactions with bait information are used. For the complex spoke case, the edge
    # number is bigger than the undirected case because edges with two baits give raise
    # to two directed edges and only one undirected.

    ## Additional info if you are using "graph":
    library(Rgraphviz)
    edgeList_complex = convert_MITAB_to_edgeList(complex_INTACT, node_names="uids")
    graph_complex = convert_edgeList_to_graph(edgeList_complex, graphical_package=
"graph")
    plot(graph_complex, "neato")

    nodes(graph_complex)
    edges(graph_complex)
    degree(graph_complex)
    numNoEdges(graph_complex)
    mostEdges(graph_complex)
    adj(graph_complex,"uniprotkb:P28043")
    acc(graph_complex, "uniprotkb:P28043")
    sg <- subGraph(nodes(graph_complex)[1:20], graph_complex)
    plot(sg, "neato")

    ## Additional info if you are using "igraph":
    V(graph_binary_INTACT)$name
    V(graph_binary_INTACT)[igraph::degree(graph_binary_INTACT) == 1]$name
    par(mfrow=c(2,3))
    plot(graph_binary_INTACT, layout=layout.fruchterman.reingold, vertex.size=3,
vertex.color="green", frame=TRUE, main="Binary", vertex.label=NA)
    plot(graph_complex_INTACT_s, layout=layout.fruchterman.reingold, vertex.size=3,
vertex.color="green", frame=TRUE, main="Complex Spoke", vertex.label=NA)
    plot(graph_complex_INTACT_m, layout=layout.fruchterman.reingold, vertex.size=3,
vertex.color="green", frame=TRUE, main="Complex Matrix", vertex.label=NA)
    plot(graph_all_INTACT, layout=layout.fruchterman.reingold, vertex.size=3,
vertex.color="green", frame=TRUE, main="Binary and Complex Spoke", vertex.label=NA)
    plot(graph_binary_INTACT_dir, layout=layout.fruchterman.reingold, vertex.size=3,
edge.arrow.size=0.3, vertex.color="green", frame=TRUE, main="Binary Directed",
```

```
 vertex.label=NA)
     plot(graph_complex_INTACT_sdir, layout=layout.fruchterman.reingold, vertex.size=3,
 edge.arrow.size=0.3, vertex.color="green", frame=TRUE, main="Complex Spoke Directed",
 vertex.label=NA)
     x11(); par(mfrow=c(1,2))
     plot(degree.distribution(graph_binary_INTACT), main="Degree Distribution (Binary,
 INTACT, E.coli)")
     plot(degree.distribution(graph_binary_INTACT),log="xy", main="Log-Log Degree
 Distribution (Binary, INTACT, E.coli)")

## End(Not run)
```

`convert_edgeList_to_MITAB` *Convert edgeList format to MITAB format*

Description

Convert a table from edgeList format to MITAB format.

On iROGs and icROGs: "iRefIndex" guarantees the non-redundancy of protein information by assigning a different protein identifier (called Redundant Object Group, ROG) to every different protein sequence. This is called the "non-canonical" representation. At the same time, groups of non-redundant proteins might be different isoforms of a given protein, and, in this case, the identifier of one protein of the group (called cROG or canonical ROG) is chosen to represent the entire group of similar proteins. This is called the "canonical" representation of proteins. The iROG and icROG IDs correspond to integer representations of the ROG and cROG, respectively. Most of the "iRefR" functions work with iROGs and icROGs.

Usage

```
convert_edgeList_to_MITAB(edgeList, MITAB_table, canonical_rep, int_type)
```

Arguments

`edgeList`	iRefIndex/edgeList R table.
`MITAB_table`	Full iRefIndex/MITAB R table for a given organism.
`canonical_rep`	Either "yes" for working with the canonical representation (icROGs) or "no" for working with the non-canonical representation (iROGs). Default="yes".
`int_type`	Interaction Type: "binary" for binary interactions, "complex" for complexes, "polymer" for polymers and "all" for the three subsets. Default="all".

Value

`MITAB_table`	iRefIndex/MITAB R table corresponding to the original edgeList.

Author(s)

Antonio Mora <antoniocmora@gmail.com>

Examples

```
    ## get tables
    irefindex_curr_ecoli = get_irefindex("562", "13.0", tempdir())
    all_INTACT = select_database("intact", irefindex_curr_ecoli, "this_database")
    binary_INTACT = select_interaction_type("binary", all_INTACT)
    complex_INTACT = select_interaction_type("complex", all_INTACT)
    edgeList_binary_INTACT = convert_MITAB_to_edgeList(binary_INTACT)
    edgeList_complex_INTACT_s = convert_MITAB_to_edgeList(complex_INTACT, "default",
"spoke")
    edgeList_all_INTACT = convert_MITAB_to_edgeList(all_INTACT, "default", "spoke")

    ## execute function
   new_binary = convert_edgeList_to_MITAB(edgeList_binary_INTACT, irefindex_curr_ecoli,
"yes", "binary")
   reconstructed_binary_INTACT = select_database("intact", new_binary, "this_database")
    setequal(dim(binary_INTACT), dim(reconstructed_binary_INTACT))

    new_complex_s = convert_edgeList_to_MITAB(edgeList_complex_INTACT_s,
irefindex_curr_ecoli, "yes", "complex")
    reconstructed_complex_INTACT_s = select_database("intact", new_complex_s,
"this_database")
    setequal(dim(complex_INTACT), dim(reconstructed_complex_INTACT_s))

    new_all = convert_edgeList_to_MITAB(edgeList_all_INTACT, irefindex_curr_ecoli)
    reconstructed_all_INTACT = select_database("intact", new_all, "this_database")
    setequal(dim(all_INTACT), dim(reconstructed_all_INTACT))

    # Note that the dimensions would not be equal for the directed case at previous
    # examples because only records with baits can be reconstructed. Directed edgeLists
    # only include information with baits, so that all other cases are ignored during
    # edgeList generation and there is no way to go back to the original MITAB.
```

---

convert_graph_to_edgeList

Convert Graph to iRefIndex/edgeList table

Description

Generate an iRefIndex/edgeList interaction table from an R graphical object.

Usage

```
convert_graph_to_edgeList(graph, directionality, graphical_package)
```

Arguments

graph	R graphical object.
directionality	Either "directed" for directed graphs or "undirected" for undirected graphs. Default="undirected".
graphical_package	Either "graph" or "igraph", depending on the R package the user used to generate the graph. If "graph" was used, no loop edges were allowed and can not be reconstructed. Default="igraph".

Value

output	iRefIndex/edgeList R table corresponding to the original graph.

Author(s)

Antonio Mora <antoniocmora@gmail.com>

Examples

```
    ## get tables
    irefindex_curr_ecoli = get_irefindex("562", "13.0", tempdir())
    all_INTACT = select_database("intact", irefindex_curr_ecoli)
    binary_INTACT = select_interaction_type("binary", all_INTACT)
    complex_INTACT = select_interaction_type("complex", all_INTACT)

    ## generate graphs
    edgeList_binary_INTACT = convert_MITAB_to_edgeList(binary_INTACT)
    edgeList_binary_INTACT_dir = convert_MITAB_to_edgeList(binary_INTACT, "default",
"bipartite", "yes", "directed")
    edgeList_complex_INTACT_s = convert_MITAB_to_edgeList(complex_INTACT, "default",
"spoke")
    edgeList_complex_INTACT_sdir = convert_MITAB_to_edgeList(complex_INTACT, "default",
"spoke", "yes", "directed")

    graph_binary_INTACT = convert_edgeList_to_graph(edgeList_binary_INTACT,
"undirected")
    graph_binary_INTACT_dir = convert_edgeList_to_graph(edgeList_binary_INTACT_dir,
"directed")
    graph_complex_INTACT_s = convert_edgeList_to_graph(edgeList_complex_INTACT_s)
    graph_complex_INTACT_sdir = convert_edgeList_to_graph(edgeList_complex_INTACT_sdir,
"directed")

    ## execute function
    reconstructed_edgeList_binary_INTACT = convert_graph_to_edgeList(
graph_binary_INTACT)
    setequal(dim(edgeList_binary_INTACT), dim(reconstructed_edgeList_binary_INTACT))

    reconstructed_edgeList_binary_INTACT_dir = convert_graph_to_edgeList(
graph_binary_INTACT_dir, "directed")
    setequal(dim(edgeList_binary_INTACT_dir), dim(
reconstructed_edgeList_binary_INTACT_dir))
```

```
    reconstructed_edgeList_complex_INTACT_s = convert_graph_to_edgeList(
graph_complex_INTACT_s)
    setequal(dim(edgeList_complex_INTACT_s), dim(
reconstructed_edgeList_complex_INTACT_s))

    reconstructed_edgeList_complex_INTACT_sdir = convert_graph_to_edgeList(
graph_complex_INTACT_sdir, "directed")
    setequal(dim(edgeList_complex_INTACT_sdir), dim(
reconstructed_edgeList_complex_INTACT_sdir))
    # NOTE: All four tests give "TRUE" when using "igraph" as graphical package, while
    # give "FALSE" when using "graph", due to the lack of polymer information in the
    # "graph" option.
```

`convert_MITAB_to_complexList` *Convert MITAB format to complexList format*

Description

Convert a table from MITAB format to complexList format. To do that, convert all interactions listed as complexes in the MITAB table and, optionally, check if some groups of binary interactions can be interpreted as complexes (possibly misrepresented or binary-represented), according to an "iRefR" algorithm that searches for groups of proteins coming from co-precipitation experiments, sharing the same PMID, source database and bait (when available).

This function might generate repeated complexes; f.ex. when the same complex is generated from two different sources. If the user wants a list of unique complexes, the "unique" function must be applied after generating the complexList.

The "complexList" format is one of the three table formats used by "iRefR", together with the MITAB format and the edgeList. The "complexList" format is a table with two columns, where the first column corresponds to an identifier and the second column to a group of proteins, commonly a complex, written as a comma-separated string. This representation has less information than MITAB but simplifies the visualization of groups of proteins, such as complexes.

On iROGs and icROGs: "iRefIndex" guarantees the non-redundancy of protein information by assigning a different protein identifier (called Redundant Object Group, ROG) to every different protein sequence. This is called the "non-canonical" representation. At the same time, groups of non-redundant proteins might be different isoforms of a given protein, and, in this case, the identifier of one protein of the group (called cROG or canonical ROG) is chosen to represent the entire group of similar proteins. This is called the "canonical" representation of proteins. The iROG and icROG IDs correspond to integer representations of the ROG and cROG, respectively. Most of the "iRefR" functions work with iROGs and icROGs.

Usage

```
convert_MITAB_to_complexList(MITAB_table, canonical_rep,
include_generated_complexes, list_methods, bait_use, node_names)
```

Arguments

MITAB_table	iRefIndex/MITAB R table.
canonical_rep	Either "yes" for working with the canonical representation (icROGs) or "no" for working with the non-canonical representation (iROGs). Default="yes".
include_generated_complexes	Parameter to include a set of complexes generated from binary-represented interactions that might be spoke-represented or misrepresented complexes, according to an "iRefR" algorithm that searches for groups of proteins coming from co-precipitation experiments, sharing the same PMID, source database and bait (when available). The user can choose between ignoring those records for which bait-prey information is not available or generating complexes from proteins with no bait information but 3 or more neighbors. To be used when the user can accept hypothetical complexes in the analysis. Default="no".
list_methods	Vector with a list of experimental methods that the user assumes should offer co-complexing information (when regenerating complexes). Methods should be described according to the MI-ontology format (see: http://www.ebi.ac.uk/ontology-lookup/browse.do?ontName=MI under "interaction detection method"). The user can either introduce a vector with the MI terms corresponding to the methods of interest or use the "default" parameter. The methods listed under "default" are: MI:0004, MI:0006, MI:0007, MI:0019, MI:0027, MI:0028, MI:0029, MI:0059, MI:0061, MI:0071, MI:0096, MI:0114, MI:0226, MI:0227, MI:0401, MI:0437, MI:0676, MI:0858 and MI:0963. The description of these methods can be found at the above-mentioned web address. Some other methods the user might be interested to add might be: MI:0067, MI:0038, MI:0104, MI:0826, MI:0069, MI:0894, MI:0893, MI:0938, MI:0099, MI:0888, MI:0966, MI:0825, MI:0824, MI:0025, MI:0928, MI:1022, MI:0982, MI:0807, MI:0276, MI:0404, MI:0808, MI:0412 and MI:0413. Default="default".
bait_use	Decision of either using only records with bait-prey information ("only_baits") or including records with no bait-prey information ("include_non_baits"), when regenerating complexes from binaries that share the same selected interaction method, PMID and source database. If the first case is chosen, all complexes will be generated as a spoke model around the bait protein. If the second case is chosen, complexes will be additionally generated as spoke models around every protein with 3 or more neighbors, for groups of proteins without bait-prey information. Default="only_baits".
node_names	Decision of either using iRefIndex ROG Identifiers or iRefIndex UIDs. Default="rogs".

Value

complexList	iRefIndex/complexList R table corresponding to the original MITAB table.

Author(s)

Antonio Mora <antoniocmora@gmail.com>

Examples

```
        ## Not run:
        ## get tables
        irefindex_curr_ecoli = get_irefindex("562", "13.0", tempdir())
        iRef_binary = select_interaction_type("binary", irefindex_curr_ecoli)

        ## execute function
        known_complexes_ecoli_complexList = convert_MITAB_to_complexList(
    irefindex_curr_ecoli, canonical_rep="yes", include_generated_complexes="no")

        ## more examples
        generated_complexes_ecoli_complexList = convert_MITAB_to_complexList(iRef_binary,
    canonical_rep="yes", include_generated_complexes="yes", list_methods="default")
       all_complexes_ecoli_complexList = convert_MITAB_to_complexList(irefindex_curr_ecoli,
    canonical_rep="yes", include_generated_complexes="yes", list_methods="default")
        number_unique_complexes_all = length(unique(all_complexes_ecoli_complexList[,2]))

        known_complexes_ecoli_complexList_extended = convert_MITAB_to_complexList(
    irefindex_curr_ecoli, canonical_rep="yes", include_generated_complexes="no",
    list_methods="default", bait_use="include_non_baits")
        generated_complexes_ecoli_complexList_extended = convert_MITAB_to_complexList(
    iRef_binary, canonical_rep="yes", include_generated_complexes="yes", list_methods=
    "default", bait_use="include_non_baits")
        all_complexes_ecoli_complexList_extended = convert_MITAB_to_complexList(
    irefindex_curr_ecoli, canonical_rep="yes", include_generated_complexes="yes",
    list_methods="default", bait_use="include_non_baits")
        number_unique_complexes_all_extended = length(unique(
    all_complexes_ecoli_complexList_extended[,2]))

    ## End(Not run)
```

---

convert_MITAB_to_edgeList

Convert MITAB format to edgeList format

Description

Convert a table from MITAB format to edgeList format.

On iROGs and icROGs: "iRefIndex" guarantees the non-redundancy of protein information by assigning a different protein identifier (called Redundant Object Group, ROG) to every different protein sequence. This is called the "non-canonical" representation. At the same time, groups of non-redundant proteins might be different isoforms of a given protein, and, in this case, the identifier of one protein of the group (called cROG or canonical ROG) is chosen to represent the entire group of similar proteins. This is called the "canonical" representation of proteins. The iROG and icROG IDs correspond to integer representations of the ROG and cROG, respectively. Most of the "iRefR" functions work with iROGs and icROGs.

"iRefIndex" also stores the Unique Identifier, UID, which is a protein identifier that represents a protein with its UniProt identifier wherever possible. If this is unknown, uses its RefSeq identifier and, if none of these exists, it uses the ROG ID.

Usage

```
    convert_MITAB_to_edgeList(MITAB_table, edge_weight_col, complex_rep, canonical_rep,
 directionality, node_names, multi_edge)
```

Arguments

MITAB_table	iRefIndex/MITAB R table.
edge_weight_col	The number of the column of the MITAB_table that includes some quantitative score of an interaction that we would like to use as edge weight in a graphical representation of the table. In case the user is not interested in weighting edges, the "default" option sets the value of "1" for all edge weights. Default="default".
complex_rep	Three possible ways of representing a protein complex: "bipartite" is the format followed by iRefIndex, consisting in pairwise interactions between a node that represents the complex and each of the protein members of the complex. "matrix" generates the matrix model of the complex, i.e., all possible pairwise interactions among the members of the complex. "spoke" generates the spoke model of the complex, i.e., pairwise interactions between one single chosen protein of the complex and each of the other members of the complex. Default="spoke".
canonical_rep	Either "yes" for working with the canonical representation (icROGs) or "no" for working with the non-canonical representation (iROGs). Default="yes".
directionality	Either "directed" if we want to produce an ordered list of directed edges (where edge A-B != edge B-A), or "undirected" in the opposite case (edge A-B = edge B-A). "Directed" lists are only possible for binary interactions and spoke models of complex interactions. In case of binary interactions, directed edges point from the bait to a prey or to another bait. In case of spoke models of complexes, the bait is chosen as the center of the spoke model; if there is no bait, the first protein of the list is arbitrarily chosen as a center. Default="undirected".
node_names	Either "uids" if we want to identify the proteins according to the iRefIndex's Unique Identifier, UID, or "rogs" if we want to use the iRefIndex ROG identifiers. Default="rogs".
multi_edge	Either "yes" if we want to observe all edges with different RIG IDs between two nodes (for example, edges between two nodes belonging to different complexes or to a binary and a complex interaction), or "no" if we want to combine such edges and work with maximum one edge between every pair of nodes. Default="no".

Value

edgeList	iRefIndex/edgeList R table with the previously specified parameters.

Author(s)

Antonio Mora <antoniocmora@gmail.com>

Examples

```
    ## get tables
    irefindex_curr_ecoli = get_irefindex("562", "13.0", tempdir())
    all_INTACT = select_database("intact", irefindex_curr_ecoli, "this_database")
    binary_INTACT = select_interaction_type("binary", all_INTACT)
    complex_INTACT = select_interaction_type("complex", all_INTACT)

    ## execute function
    # NOTE: The matrix model is more time-consuming.
    edgeList_binary_INTACT = convert_MITAB_to_edgeList(binary_INTACT)
    edgeList_complex_INTACT_s = convert_MITAB_to_edgeList(complex_INTACT, "default",
"spoke")

    ## Not run:
    edgeList_complex_INTACT_m = convert_MITAB_to_edgeList(complex_INTACT, "default",
"matrix")
    edgeList_all_INTACT = convert_MITAB_to_edgeList(all_INTACT, "default", "spoke")

    edgeList_binary_INTACT_dir = convert_MITAB_to_edgeList(binary_INTACT, "default",
"bipartite", "yes", "directed")
    edgeList_complex_INTACT_sdir = convert_MITAB_to_edgeList(complex_INTACT, "default",
"spoke", "yes", "directed")
    edgeList_all_INTACT_dir = convert_MITAB_to_edgeList(all_INTACT, "default", "spoke",
"yes", "directed")

    # Multi-edge interaction (C|X case) using UIDs:
    edgeList_irefindex = convert_MITAB_to_edgeList(irefindex_curr_ecoli, node_names=
"uids", multi_edge="no")
    chosen_int = 1 # Choose first multiple edge in the data frame
    rigids_multi_edge = edgeList_irefindex[grep("C\|X", edgeList_irefindex[,4])[[
chosen_int]], 3]
    rigids_multi_edge = strsplit(rigids_multi_edge, "|", fixed="TRUE")[[1]]

    #edgeList_irefindex = convert_MITAB_to_edgeList(irefindex_curr_ecoli, node_names=
    #"uids", multi_edge="yes")
    index_interesting_multi_edge = unlist(lapply(rigids_multi_edge, grep,
edgeList_irefindex[,3]))
    edgeList_interesting_complex = edgeList_irefindex[index_interesting_multi_edge,]
    graph_irefindex = convert_edgeList_to_graph(edgeList_interesting_complex)
    list_edge_colors = NULL
    list_edge_colors[which(edgeList_interesting_complex[,3]=="625881")] = "blue"
    list_edge_colors[which(edgeList_interesting_complex[,3]=="634347")] = "red"
    list_edge_colors[which(edgeList_interesting_complex[,3]=="922617")] = "green"
    list_edge_colors[which(edgeList_interesting_complex[,3]=="982855")] = "purple"
    graph_irefindex <- set.edge.attribute(graph_irefindex, "color", value=
list_edge_colors)

    plot(graph_irefindex, layout=layout.kamada.kawai, vertex.label=
```

```
 V(graph_irefindex)$name, vertex.size=7, main="multi_edge interactions")
     legend("bottomright", c("complex1", "binary", "complex2", "complex3"),
 fill=c("blue", "red", "green", "purple"))

## End(Not run)
```

`convert_protein_ID` *Convert Protein Identifier*

Description

Convert protein identifiers from one of the following formats: iROG ID, icROG ID, gene ID, RefSeq and UniProt, to any other protein identifier in the list.

On iROGs and icROGs: "iRefIndex" guarantees the non-redundancy of protein information by assigning a different protein identifier (called Redundant Object Group, ROG) to every different protein sequence. This is called the "non-canonical" representation. At the same time, groups of non-redundant proteins might be different isoforms of a given protein, and, in this case, the identifier of one protein of the group (called cROG or canonical ROG) is chosen to represent the entire group of similar proteins. This is called the "canonical" representation of proteins. The iROG and icROG IDs correspond to integer representations of the ROG and cROG, respectively. Most of the "iRefR" functions work with iROGs and icROGs.

Usage

```
     convert_protein_ID(input_id_type, input_id_value, output_id_type,
 id_conversion_table)
```

Arguments

`input_id_type`	Input Type of Protein Identifier: iRefIndex non-canonical ("irogid"), iRefIndex canonical ("icrogid"), UniProt ("uniprotkb"), RefSeq ("refseq"), and GeneID ("entrezgene/locuslink").
`input_id_value`	Input Protein Identifier (value).
`output_id_type`	Output Type of Protein Identifier: iRefIndex non-canonical ("irogid"), iRefIndex canonical ("icrogid"), UniProt ("uniprotkb"), RefSeq ("refseq"), and GeneID ("entrezgene/locuslink").
`id_conversion_table`	Variable storing the ID Conversion Table (generated using "create_id_conversion_table()").

Value

`output_id_value`	Output Protein Identifier (value).

Author(s)

Antonio Mora <antoniocmora@gmail.com>

Examples

```
    ## Not run:
    ## get tables
    irefindex_80_ecoli = get_irefindex("562", "8.0", tempdir())
    id_conversion_table_ecoli = create_id_conversion_table(irefindex_80_ecoli,
tempdir(), "id_conversion_table_562")

    ## converting from geneID to uniprotID
    uniprot_id_value = convert_protein_ID("entrezgene/locuslink", "2986728",
"uniprotkb", id_conversion_table_ecoli)

    ## converting a list of iROG IDs (from version 8.0) to RefSeq IDs
    irog_list = c("9917588", "9917576", "9917577", "369724", "593426", "3033592")
    refseq_list = list()
    for (i in irog_list) {
     refseq_list[[i]] = convert_protein_ID("irogid", i, "refseq",
id_conversion_table_ecoli)
    }

## End(Not run)
```

---

create_id_conversion_table

Create a Protein ID Conversion Table

Description

Use iRefIndex to generate a lookup table to convert protein IDs between iRefIndex's iROG IDs, icROG IDs, and other protein identifiers such as Gene IDs, RefSeq IDs, PDBs and UniProt IDs.

On iROGs and icROGs: "iRefIndex" guarantees the non-redundancy of protein information by assigning a different protein identifier (called Redundant Object Group, ROG) to every different protein sequence. This is called the "non-canonical" representation. At the same time, groups of non-redundant proteins might be different isoforms of a given protein, and, in this case, the identifier of one protein of the group (called cROG or canonical ROG) is chosen to represent the entire group of similar proteins. This is called the "canonical" representation of proteins. The iROG and icROG IDs correspond to integer representations of the ROG and cROG, respectively. Most of the "iRefR" functions work with iROGs and icROGs.

Usage

```
create_id_conversion_table(MITAB_table, data_folder, output_filename, IDs_to_include)
```

Arguments

MITAB_table	iRefIndex/MITAB R table.
data_folder	Folder to save the text file: type "data" to save it in the "iRefR/data" directory, "home" to save it in the "R.home()" directory, or any other destination folder. Default = getwd().
output_filename	File name for the conversion table files. Extensions ".txt" and ".RData" would be automatically added. Default="id_conversion_table"
IDs_to_include	Vector with the protein identifiers you want to convert from/to. iRefIndex iROGs and canonical iROGs are always included in the table. In addition, currently supports: UniProt ("uniprotkb"), RefSeq ("refseq"), GeneID ("entrezgene/locuslink"), Protein Data Bank ("PDB"), c("uniprotkb", "refseq", "entrezgene/locuslink") ("default") and all of them ("all"). Default = "default".

Value

output	Lookup table including iROG ID, icROG ID, and the other specified protein IDs.

Author(s)

Antonio Mora <antoniocmora@gmail.com>

Examples

```
    ## get table
    irefindex_curr_ecoli = get_irefindex("562", "13.0", tempdir())

    ## execute function
    id_conversion_table_ecoli = create_id_conversion_table(irefindex_curr_ecoli,
 tempdir(), "id_conversion_table_562_a")
```

---

get_irefindex *Get iRefIndex*

Description

Get a given distribution of the iRefIndex database from the ftp site.

"iRefIndex" is one of the main efforts to consolidate the information coming from different protein interaction databases in one single repository avoiding redundant information. It has been defined as "an index of protein interactions available in a number of primary interaction databases including BIND, BioGRID, CORUM, DIP, HPRD, IntAct, MINT, Mpact, MPPI and OPHID... This index allows the user to search for a protein and retrieve a non-redundant list of interactors for that protein".

Usage

```
get_irefindex(tax_id, iref_version, data_folder)
```

Arguments

tax_id	Taxon ID of the organism ("10090" for mouse, "10116" for rat, "4932" for S.Cerevisiae, "562" for E.Coli, "6239" for C.Elegans, "7227" for D.Melanogaster, "9606" for H.Sapiens, and "All" for all organisms). Default="All".
iref_version	iRefIndex distribution ("7.0", "8.0", "9.0", "10.0", "11.0", "12.0", "13.0" or "current"). Default="current".
data_folder	Folder to save the iRefIndex flat file: type "data" to save it in the "iRefR/data" directory, "home" to save it in the "R.home()" directory, or any other destination folder. Default=getwd().

Value

irefindex_tab	R Table (data frame) containing the desired iRefIndex distribution.

Author(s)

Antonio Mora <antoniocmora@gmail.com>

Examples

```
    ## execute function
    irefindex_13_ecoli = get_irefindex(tax_id="562", iref_version="13.0", data_folder=
 tempdir())
```

---

merge_complexes_lists *Merge tables in complexList Format*

Description

Merge a list of tables in complexList format, written in a prioritized order, and produce a single table where redundancies where removed according to the specified priorities.

The "complexList" format is one of the three table formats used by "iRefR", together with the MITAB format and the edgeList. The "complexList" format is a table with two columns, where the first column corresponds to an identifier and the second column to a group of proteins, commonly a complex, written in a comma-separated string.

Usage

```
merge_complexes_lists(list_of_complexLists)
```

Arguments

`list_of_complexLists`
: List of tables in the "complexList" format. All of these tables must share the same representation of proteins, either canonical or non-canonical. Mixes would produce meaningless results. Repeated complexes inside individual tables would be removed. At the same time, when some group of proteins is repeated between two or more tables, it will be removed from all tables with less priority and will only stay in the table with the highest priority. The highest priority would be assigned to the first written table, while the smallest priority would go to the last one.

Value

`result`
: Table in the "complexList" format, with the prioritized merging of a set of tables in the "complexList" format.

Author(s)

Antonio Mora <antoniocmora@gmail.com>

Examples

```
    ## Not run:
    ## get tables
    irefindex_curr_ecoli = get_irefindex("562", "13.0", tempdir())
    iRef_binary = select_interaction_type("binary", irefindex_curr_ecoli)
    iRef_complex = select_interaction_type("complex", irefindex_curr_ecoli)
    complex_complexList = convert_MITAB_to_complexList(iRef_complex, "yes", "no")
    binary_complexList = convert_MITAB_to_complexList(iRef_binary, "yes", "yes")

    ## execute function
    consolidated_complexList = merge_complexes_lists(list(complex_complexList,
binary_complexList))
    number_complexes_repeated_inside_lists_or_between_lists = dim(
complex_complexList)[1] + dim(binary_complexList)[1] - dim(consolidated_complexList)[1]

## End(Not run)
```

---

select_confidence — *Select the subset of iRefIndex corresponding to a set of confidence scores*

Description

Select the subset of records of iRefIndex corresponding to a given set of confidence scores.

The lpr score (lowest pmid re-use) is the lowest number of distinct interactions (irigids) that any PMID (supporting the interaction in this row) is used to support. A value of one indicates that

at least one of the PMIDs supporting this interaction has never been used to support any other interaction. This likely indicates that only one interaction was described by that reference and that the present interaction is not derived from high throughput methods.

The hpr score (highest pmid re-use) is the highest number of interactions that any PMID (supporting the interaction in this row) is used to support. A high value (e.g. greater than 50) indicates that one PMID describes at least 50 other interactions and it is more likely that high-throughput methods were used.

The np score (number pmids) is the total number of unique PMIDs used to support the interaction described in this row.

Usage

```
select_confidence(confidence_type, confidence_value, MITAB_table)
```

Arguments

`confidence_type`	Name of the confidence score for an interaction record. Options: lowest pmid re-use ("lpr"), highest pmid re-use ("hpr"), number pmids ("np"). Default: "lpr".
`confidence_value`	Vector with the values of the score whose interaction records we want to retrieve.
`MITAB_table`	iRefIndex/MITAB R table.

Value

`MITAB_output`	iRefIndex/MITAB R table with the previously specified features.

Author(s)

Antonio Mora <antoniocmora@gmail.com>

Examples

```
## get tables
irefindex_curr_ecoli = get_irefindex("562", "13.0", tempdir())
iRef_complex = select_interaction_type("complex", irefindex_curr_ecoli)

## execute function
high_confidence_complexes = select_confidence("lpr", c(1:3, 10:15), iRef_complex)
```

`select_database` *Select iRefIndex Records belonging to given Primary Interaction Databases*

Description

Select the subset of records in iRefIndex belonging to one or a group of Primary Interaction Databases. In addition, select all records except for that database or those databases.

Usage

```
select_database(database, MITAB_table, flag)
```

Arguments

`database`	Name of the protein interaction DB or vector of databases to extract from iRefIndex (options: "intact", "bind", "bind_translation", "biogrid", "mint", "ophid", "mppi", "hprd", "dip", "corum", "mpact", "innatedb", "matrixdb", "mpi-lit", "mpi-imex"). Before iRefIndex 10.0, the BioGRID database was called "grid" and the "mpi-lit" and "mpi-imex" were called "mpilit" and "mpiimex" respectively.
`MITAB_table`	iRefIndex/MITAB R table.
`flag`	Either "this_database" to get all "database" records, or "except_this_database" to get all records except for the ones belonging to "database". Default="this_database"

Value

`output`	iRefIndex/MITAB R table with the previously specified features.

Author(s)

Antonio Mora <antoniocmora@gmail.com>

Examples

```
    ## get tables
    irefindex_curr_ecoli = get_irefindex("562", "13.0", tempdir())
    iRef_binary = select_interaction_type("binary", irefindex_curr_ecoli)
    iRef_complex = select_interaction_type("complex", irefindex_curr_ecoli)

    ## execute function
    binary_INTACT = select_database("intact", iRef_binary, "this_database")
    binary_non_INTACT = select_database("intact", iRef_binary, "except_this_database")
    complex_INTACT_mint = select_database(c("intact", "mint"), iRef_complex,
"this_database")
    complex_non_INTACT_mint = select_database(c("intact", "mint"), iRef_complex,
"except_this_database")
```

select_interaction_type *Select Binary||Polymer||Complex in iRefIndex*

Description

Select the subset of records in iRefIndex belonging to a certain interaction type (binary, polymer, complex).

Usage

```
select_interaction_type(int_type, MITAB_table)
```

Arguments

int_type	Interaction type ("binary", "complex" or "polymer").
MITAB_table	iRefIndex/MITAB R table.

Value

output	R Table containing the requested subset of iRefIndex.

Author(s)

Antonio Mora <antoniocmora@gmail.com>

Examples

```
## get tables
irefindex_curr_ecoli = get_irefindex("562", "13.0", tempdir())

## execute function
iRef_binary = select_interaction_type("binary", irefindex_curr_ecoli)
iRef_complex = select_interaction_type("complex", irefindex_curr_ecoli)
iRef_polymer = select_interaction_type("polymer", irefindex_curr_ecoli)

## get results
#pie_int_type = pie(c(dim(iRef_binary)[1], dim(iRef_polymer)[1], dim(
#iRef_complex)[1]), labels=c("Binaries","Polymers","Complexes"), main=
#"iRefIndex Records per Interaction Type")
#edit(iRef_complex[1:10,])
```

select_protein *Get all iRefIndex records for a given Protein*

Description

Get all iRefIndex records for a given protein.

Usage

```
select_protein(id_type, id_value, MITAB_table, complex_info)
```

Arguments

id_type	Protein ID type ("irog" for iRefIndex's protein iROG ID; "icrog" for the canonical version of the iRefIndex's protein iROG ID; "uid" for iRefIndex's protein UID, which is formed of mainly UniProt IDs, followed by RefSeq IDs when there is no known UniProt ID and iROG IDs in case no UniProt or RefSeq are known).
id_value	A value or a vector of values of the iROG, icROG or UID to search for in a given iRefIndex/MITAB table.
MITAB_table	iRefIndex/MITAB R table.
complex_info	Flag that gives the user the option to retrieve either all the records of the complex to which such protein belongs to ("full_complex"), or only the record of the complex where the given protein is present ("not_full_complex"). Default="full_complex".

Value

output	iRefIndex/MITAB R table.

Author(s)

Antonio Mora <antoniocmora@gmail.com>

Examples

```
    ## Not run:
    ## get tables
    irefindex_80_mouse = get_irefindex("10090", "8.0", tempdir())

    ## single queries
    output_1 = select_protein("irogid", "4374882", irefindex_80_mouse,
"not_full_complex")
    output_2 = select_protein("icrogid", "4374882", irefindex_80_mouse,
"not_full_complex")
    output_3 = select_protein("icrogid", "2892004", irefindex_80_mouse,
```

```
"not_full_complex")
    output_4 = select_protein("icrogid", "2892004", irefindex_80_mouse,
"full_complex")
    output_5 = select_protein("icrogid", "1365685", irefindex_80_mouse,
"not_full_complex")
    output_6 = select_protein("icrogid", "1365685", irefindex_80_mouse,
"full_complex")

    ## getting list of tables
    irog_list = c("1828087", "2892004", "1365685")
    table_list = list()
    for (i in irog_list) {
     table_list[[i]] = select_protein("irogid", i, irefindex_80_mouse,
"not_full_complex")
    }

    ## getting single table
    irog_list = c("1828087", "2892004", "1365685")
    table_single = select_protein("irogid", irog_list, irefindex_80_mouse,
"not_full_complex")

## End(Not run)
```

---

summary_graph *Get Summary Information for a Graph*

Description

Get statistical information about the number of nodes, edges, degree distribution, nodes per connected component and graphs of connected components in an R Graph.

Here "iRefR" uses the "graph", "igraph" and "RBGL" libraries.

Usage

```
summary_graph(graph_object, graphical_package)
```

Arguments

graph_object	R graph to summarize.
graphical_package	Either "graph" or "igraph", depending on the R package the user wants to use. Default="igraph".

Value

output	R list containing: Number of nodes and edges, degree distribution, nodes per connected component and graphs per connected component. A plot of the graph is also produced if the "igraph" option is chosen.

Author(s)

Antonio Mora <antoniocmora@gmail.com>

Examples

```
## get tables
irefindex_curr_ecoli = get_irefindex("562", "13.0", tempdir())
INTACT_curr_ecoli = select_database("intact", irefindex_curr_ecoli, "this_database")
edgeList_ecoli = convert_MITAB_to_edgeList(INTACT_curr_ecoli, "default", "spoke")
graph_ecoli = convert_edgeList_to_graph(edgeList_ecoli, graphical_package="igraph")

## execute function
if(interactive()) {
 summary = summary_graph(graph_ecoli, "igraph")
 summary$nodes_and_edges
 table(summary$degree_distribution)
 x11(); hist(summary$degree_distribution)

 table(summary$nodes_per_connected_component)
 x11(); barplot(summary$nodes_per_connected_component, xlab="Size Components",
ylab="Number Components")
 title(main="Distribution of Connected Components")

 summary$graph_per_module[[1]]
}
```

---

summary_protein *Get Summary Information for a given Protein*

Description

Get information on interactors and complexes, for a given protein.

Usage

```
summary_protein(id_type, id_value, MITAB_table)
```

Arguments

id_type	Protein Identifier type ("irogid" for iRefIndex's protein iROG ID; "icrogid" for the canonical version of the iRefIndex's protein iROG ID).
id_value	Value of the iROG or icROG to summarize.
MITAB_table	MITAB/iRefIndex R table.

Value

output — R list containing: Number of interactions per interaction type, interactors in binary interactions, interactors in complex interactions and interactors in polymer interactions.

Author(s)

Antonio Mora <antoniocmora@gmail.com>

Examples

```
## Not run:
## get tables
irefindex_80_mouse = get_irefindex("10090", "8.0", tempdir())

## execute function
output_1 = summary_protein("irogid", "4374882", irefindex_80_mouse)
output_2 = summary_protein("icrogid", "4374882", irefindex_80_mouse)
output_3 = summary_protein("icrogid", "2892004", irefindex_80_mouse)
output_4 = summary_protein("icrogid", "1365685", irefindex_80_mouse)

## End(Not run)
```

---

summary_table *Get Summary Information for a MITAB Table*

Description

Get statistical information about some of the columns of a given iRefIndex/MITAB table.

Usage

```
summary_table(MITAB_table)
```

Arguments

MITAB_table — MITAB/iRefIndex R table to summarize.

Value

output — R list containing: Number of non-canonical interactions, number of canonical interactions, number of non-canonical proteins, number of canonical proteins, number of publications, number of experimental methods, distribution of source databases, distribution of interaction types, distribution of number of participants.

Author(s)

Antonio Mora <antoniocmora@gmail.com>

Examples

```
    ## get tables
    irefindex_13_ecoli = get_irefindex("562", "13.0", tempdir())

    ## execute function
    general_ecoli_statistics = summary_table(irefindex_13_ecoli)
    intact_statistics = summary_table(select_database("intact", irefindex_13_ecoli,
"this_database"))
    complexes_statistics = summary_table(select_interaction_type("complex",
irefindex_13_ecoli))
    low_thruput_statistics = summary_table(select_confidence("lpr", c(1,2),
irefindex_13_ecoli))
```

Index