Package 'emba'

January 7, 2021

Type Package

Title Ensemble Boolean Model Biomarker Analysis

Version 0.1.8

Description Analysis and visualization of an ensemble of boolean models for biomarker discovery in cancer cell networks. The package allows to easily load the simulation data results of the DrugLogics software pipeline which predicts synergistic drug combinations in cancer cell lines (developed by the DrugLogics research group in NTNU). It has generic functions that can be used to split a boolean model dataset to model groups with regards to the models predictive performance (number of true positive predictions/Matthews correlation coefficient score) or synergy prediction based on a given set of gold standard synergies and find the average activity difference per network node between all model group pairs. Thus, given user-specific thresholds, important nodes (biomarkers) can be accessed in the sense that they make the models predict specific synergies (synergy biomarkers) or have better performance in general (performance biomarkers). Lastly, if the boolean models have a specific equation form and differ only in their link operator, link operator biomarkers can also be found.

License MIT + file LICENSE

URL https://bblodfon.github.io/emba/, https://github.com/bblodfon/emba, https://github.com/druglogics/

BugReports https://github.com/bblodfon/emba/issues

Encoding UTF-8

LazyData true

RoxygenNote 7.1.1

Depends R (>= 2.10)

Imports graphics, grDevices, utils, purrr, rje (>= 1.10), igraph (>= 1.2.4), visNetwork (>= 2.0.9), Ckmeans.1d.dp (>= 4.2.2), usefun (>= 0.4.3), readr (>= 1.3.0), dplyr (>= 1.0.0), tidyr (>= 1.1.0), tidyselect (>= 1.0.0), stringr (>= 1.4.0), tibble (>= 3.0.0)

Suggests testthat, knitr, rmarkdown, xfun

VignetteBuilder knitr

NeedsCompilation no

Author John Zobolas [aut, cph, cre] (<https://orcid.org/0000-0002-3609-8674>)

Maintainer John Zobolas <bblodfon@gmail.com>

Repository CRAN

Date/Publication 2021-01-07 04:00:02 UTC

R topics documented:

`add_numbers_above_the_bars` *Add numbers horizontally above the bars of a barplot*

Description

Add numbers horizontally above the bars of a barplot

Usage

```
add_numbers_above_the_bars(stats, bp, color)
```

Arguments

`stats`	a numeric vector
`bp`	the result of `barplot` command, usually a numeric vector or matrix
`color`	string. The color for the numbers

`assign_link_operator_value_to_equation` *Assign link operator value to boolean equation*

Description

Assign link operator value to boolean equation

Usage

```
assign_link_operator_value_to_equation(equation)
```

Arguments

`equation`	string. The boolean equation in the form $Target* = (ActivatororActivatoror...)andnot(Inhibitororl$

Value

1 if the `equation` has the '*or not*' link operator, **0** if the `equation` has the '*and not*' link operator and **NA** if it has neither.

`biomarker_mcc_analysis` *Biomarker analysis based on MCC model classification*

Description

Use this function to perform a full biomarker analysis on an ensemble boolean model dataset where the model classification is based on the *Matthews correlation coefficient score (MCC)*. This analysis enables the discovery of *performance biomarkers*, nodes whose activity and/or boolean model parameterization (link operator) affects the prediction performance of the models (as measured by the MCC score).

Usage

```
biomarker_mcc_analysis(
  model.predictions,
  models.stable.state,
  models.link.operator = NULL,
  observed.synergies,
  threshold,
  num.of.mcc.classes = 5,
  penalty = 0.1
)
```

Arguments

`model.predictions`
: a `data.frame` object with rows the models and columns the drug combinations. Possible values for each *model-drug combination element* are either *0* (no synergy predicted), *1* (synergy was predicted) or *NA* (couldn't find stable states in either the drug combination inhibited model or in any of the two single-drug inhibited models).

`models.stable.state`
: a `data.frame` (nxm) with n models and m nodes. The row names specify the models' names whereas the column names specify the network nodes (gene, proteins, etc.). Possible values for each *model-node element* can be between *0* (inactive node) and *1* (active node) inclusive. Note that the rows (models) have to be in the same order as in the `model.predictions` parameter.

`models.link.operator`
: a `data.frame` (nxm) with n models and m nodes. The row names specify the models' names (same order as in the `model.predictions` parameter) whereas the column names specify the network nodes (gene, proteins, etc.). Possible values for each *model-node element* are either *0* (**AND NOT** link operator), *1* (**OR NOT** link operator) or *0.5* if the node is not targeted by both activating and inhibiting regulators (no link operator). Default value: NULL (no analysis on the models parameterization regarding the mutation of the boolean equation link operator will be done).

`observed.synergies`
: a character vector with elements the names of the drug combinations that were found as synergistic. This should be a subset of the tested drug combinations, that is the column names of the `model.predictions` parameter.

`threshold`
: numeric. A number in the [0,1] interval, above which (or below its negative value) a biomarker will be registered in the returned result. Values closer to 1 translate to a more strict threshold and thus less biomarkers are found.

`num.of.mcc.classes`
: numeric. A positive integer larger than 2 that signifies the number of mcc classes (groups) that we should split the models MCC values. Default value: 5.

`penalty`
: value between 0 and 1 (inclusive). A value of 0 means no penalty and a value of 1 is the strickest possible penalty. Default value is 0.1. This penalty is used as part of a weighted term to the difference in a value of interest (e.g. activity or link operator difference) between two group of models, to account for the difference in the number of models from each respective model group.

Value

a list with various elements:

- `predicted.synergies`: a character vector of the synergies (drug combination names) that were predicted by **at least one** of the models in the dataset.
- `models.mcc`: a numeric vector of MCC scores, one for each model. Values are in the [-1,1] interval.

- `diff.state.mcc.mat`: a matrix whose rows are **vectors of average node activity state differences** between two groups of models where the classification was based on the *MCC score* of each model and was found using an optimal univariate k-means clustering method (`Ckmeans.1d.dp`). Rows represent the different classification group matchings, e.g. (1,2) means the models that were classified into the first MCC class vs the models that were classified in the 2nd class (higher is better). The columns represent the network's node names. Values are in the [-1,1] interval.
- `biomarkers.mcc.active`: a character vector whose elements are the names of the *active state* biomarkers. These nodes appear more active in the better performance models.
- `biomarkers.mcc.inhibited`: a character vector whose elements are the names of the *inhibited state* biomarkers. These nodes appear more inhibited in the better performance models.
- `diff.link.mcc.mat`: a matrix whose rows are **vectors of average node link operator differences** between two groups of models where the classification was based on the *MCC score* of each model and was found using an optimal univariate k-means clustering method (`Ckmeans.1d.dp`). Rows represent the different classification group matchings, e.g. (1,2) means the models that were classified into the first MCC class vs the models that were classified in the 2nd class (higher is better). The columns represent the network's node names. Values are in the [-1,1] interval.
- `biomarkers.mcc.or`: a character vector whose elements are the names of the *OR* link operator biomarkers. These nodes have mostly the *OR* link operator in their respective boolean equations in the better performance models.
- `biomarkers.mcc.and`: a character vector whose elements are the names of the *AND* link operator biomarkers. These nodes have mostly the *AND* link operator in their respective boolean equations in the better performance models.

See Also

Other general analysis functions: `biomarker_synergy_analysis()`, `biomarker_tp_analysis()`

`biomarker_synergy_analysis`

Biomarker analysis per synergy predicted

Description

Use this function to discover *synergy biomarkers*, i.e. nodes whose activity and/or boolean equation parameterization (link operator) affect the manifestation of synergies in the boolean models. Models are classified to groups based on whether they predict or not each of the predicted synergies.

Usage

```
biomarker_synergy_analysis(
  model.predictions,
  models.stable.state,
  models.link.operator = NULL,
```

```
  observed.synergies,
  threshold,
  calculate.subsets.stats = FALSE,
  penalty = 0.1
)
```

Arguments

`model.predictions`
: a `data.frame` object with rows the models and columns the drug combinations. Possible values for each *model-drug combination element* are either *0* (no synergy predicted), *1* (synergy was predicted) or *NA* (couldn't find stable states in either the drug combination inhibited model or in any of the two single-drug inhibited models).

`models.stable.state`
: a `data.frame` (nxm) with n models and m nodes. The row names specify the models' names whereas the column names specify the network nodes (gene, proteins, etc.). Possible values for each *model-node element* can be between *0* (inactive node) and *1* (active node) inclusive. Note that the rows (models) have to be in the same order as in the `model.predictions` parameter.

`models.link.operator`
: a `data.frame` (nxm) with n models and m nodes. The row names specify the models' names (same order as in the `model.predictions` parameter) whereas the column names specify the network nodes (gene, proteins, etc.). Possible values for each *model-node element* are either *0* (**AND NOT** link operator), *1* (**OR NOT** link operator) or *0.5* if the node is not targeted by both activating and inhibiting regulators (no link operator). Default value: NULL (no analysis on the models parameterization regarding the mutation of the boolean equation link operator will be done).

`observed.synergies`
: a character vector with elements the names of the drug combinations that were found as synergistic. This should be a subset of the tested drug combinations, that is the column names of the `model.predictions` parameter.

`threshold`
: numeric. A number in the [0,1] interval, above which (or below its negative value) a biomarker will be registered in the returned result. Values closer to 1 translate to a more strict threshold and thus less biomarkers are found.

`calculate.subsets.stats`
: logical. If *TRUE*, then the results will include a vector of integers, representing the number of models that predicted every subset of the given `observed.synergies` (where at least one model predicts every synergy in the subset). The default value is *FALSE*, since the powerset of the predicted `observed.synergies` can be very large to compute.

`penalty`
: value between 0 and 1 (inclusive). A value of 0 means no penalty and a value of 1 is the strickest possible penalty. Default value is 0.1. This penalty is used as part of a weighted term to the difference in a value of interest (e.g. activity or link operator difference) between two group of models, to account for the difference in the number of models from each respective model group.

Value

a list with various elements:

- `predicted.synergies`: a character vector of the synergies (drug combination names) that were predicted by **at least one** of the models in the dataset.
- `synergy.subset.stats`: an integer vector with elements the number of models the predicted each **observed synergy subset** if the *calculate.subsets.stats* option is enabled.
- `synergy.comparison.sets`: a `data.frame` with pairs of *(set, subset)* for each model-predicted synergy where each respective subset misses just one synergy from the larger set (present only if the *calculate.subsets.stats* option is enabled). Can be used to refine the synergy biomarkers by comparing any two synergy sets with the functions `get_avg_activity_diff_based_on_synergy_set_cmp` or `get_avg_link_operator_diff_based_on_synergy_set_cmp`.
- `diff.state.synergies.mat`: a matrix whose rows are **vectors of average node activity state differences** between two groups of models where the classification for each individual row was based on the prediction or not of a specific synergistic drug combination. The row names are the predicted synergies, one per row, while the columns represent the network's node names. Values are in the [-1,1] interval.
- `activity.biomarkers`: a `data.frame` object with rows the `predicted synergies` and columns the nodes (column names of the `models.stable.states` matrix). Possible values for each *synergy-node* element are either *1* (*active state* biomarker), *-1* (*inhibited state* biomarker) or *0* (not a biomarker) for the given `threshold` value.
- `diff.link.synergies.mat`: a matrix whose rows are **vectors of average node link operator differences** between two groups of models where the classification for each individual row was based on the prediction or not of a specific synergistic drug combination. The row names are the predicted synergies, one per row, while the columns represent the network's node names. Values are in the [-1,1] interval.
- `link.operator.biomarkers`: a `data.frame` object with rows the `predicted synergies` and columns the nodes (column names of the `models.link.operator` matrix). Possible values for each *synergy-node* element are either *1* (*OR* link operator biomarker), *-1* (*AND* link operator biomarker) or *0* (not a biomarker) for the given `threshold` value.

See Also

Other general analysis functions: `biomarker_mcc_analysis()`, `biomarker_tp_analysis()`

`biomarker_tp_analysis` *Biomarker analysis based on TP model classification*

Description

Use this function to perform a full biomarker analysis on an ensemble boolean model dataset where the model classification is based on the number of *true positive* (TP) predictions. This analysis enables the discovery of *performance biomarkers*, nodes whose activity and/or boolean model parameterization (link operator) affects the prediction performance of the models (as measured by the number of TPs).

Usage

```
biomarker_tp_analysis(
  model.predictions,
  models.stable.state,
  models.link.operator = NULL,
  observed.synergies,
  threshold,
  penalty = 0.1
)
```

Arguments

`model.predictions`
: a `data.frame` object with rows the models and columns the drug combinations. Possible values for each *model-drug combination element* are either *0* (no synergy predicted), *1* (synergy was predicted) or *NA* (couldn't find stable states in either the drug combination inhibited model or in any of the two single-drug inhibited models).

`models.stable.state`
: a `data.frame` (nxm) with n models and m nodes. The row names specify the models' names whereas the column names specify the network nodes (gene, proteins, etc.). Possible values for each *model-node element* can be between *0* (inactive node) and *1* (active node) inclusive. Note that the rows (models) have to be in the same order as in the `model.predictions` parameter.

`models.link.operator`
: a `data.frame` (nxm) with n models and m nodes. The row names specify the models' names (same order as in the `model.predictions` parameter) whereas the column names specify the network nodes (gene, proteins, etc.). Possible values for each *model-node element* are either *0* (**AND NOT** link operator), *1* (**OR NOT** link operator) or *0.5* if the node is not targeted by both activating and inhibiting regulators (no link operator). Default value: NULL (no analysis on the models parameterization regarding the mutation of the boolean equation link operator will be done).

`observed.synergies`
: a character vector with elements the names of the drug combinations that were found as synergistic. This should be a subset of the tested drug combinations, that is the column names of the `model.predictions` parameter.

`threshold`
: numeric. A number in the [0,1] interval, above which (or below its negative value) a biomarker will be registered in the returned result. Values closer to 1 translate to a more strict threshold and thus less biomarkers are found.

`penalty`
: value between 0 and 1 (inclusive). A value of 0 means no penalty and a value of 1 is the strickest possible penalty. Default value is 0.1. This penalty is used as part of a weighted term to the difference in a value of interest (e.g. activity or link operator difference) between two group of models, to account for the difference in the number of models from each respective model group.

Value

a list with various elements:

- `predicted.synergies`: a character vector of the synergies (drug combination names) that were predicted by **at least one** of the models in the dataset.
- `models.synergies.tp`: an integer vector of true positive (TP) values, one for each model.
- `diff.tp.mat`: a matrix whose rows are **vectors of average node activity state differences** between two groups of models where the classification was based on the number of true positive predictions. Rows represent the different classification group matchings, e.g. (1,2) means the models that predicted 1 TP synergy vs the models that predicted 2 TP synergies and the columns represent the network's node names. Values are in the [-1,1] interval.
- `biomarkers.tp.active`: a character vector whose elements are the names of the *active state* biomarkers. These nodes appear as more active in the better performance models.
- `biomarkers.tp.inhibited`: a character vector whose elements are the names of the *inhibited state* biomarkers. These nodes appear as more inhibited in the better performance models.
- `diff.link.tp.mat`: a matrix whose rows are **vectors of average node link operator differences** between two groups of models where the classification was based on the number of true positive predictions. Rows represent the different classification group matchings, e.g. (1,2) means the models that predicted 1 TP synergy vs the models that predicted 2 TP synergies and the columns represent the network's node names. Values are in the [-1,1] interval.
- `biomarkers.tp.or`: a character vector whose elements are the names of the *OR* link operator biomarkers. These nodes have mostly the *OR* link operator in their respective boolean equations in the better performance models.
- `biomarkers.tp.and`: a character vector whose elements are the names of the *AND* link operator biomarkers. These nodes have mostly the *AND* link operator in their respective boolean equations in the better performance models.

See Also

Other general analysis functions: `biomarker_mcc_analysis()`, `biomarker_synergy_analysis()`

`calculate_mcc` *Calculate Matthews correlation coefficient vector*

Description

Use this function to calculate the MCC scores given vectors of *TP* (true positives), *FP* (false positives), *TN* (true negatives) and *FN* (false negatives) values. Note that the input vectors have to be of the same size and have one-to-one value correspondence for the output MCC vector to make sense.

Usage

```
calculate_mcc(tp, tn, fp, fn)
```

Arguments

tp	numeric vector of TPs
tn	numeric vector of TNs
fp	numeric vector of FPs
fn	numeric vector of FNs

Value

a numeric vector of MCC values, each value being in the [-1,1] interval. If any of the four sums of the MCC formula are zero, then we return an MCC score of zero, which can be shown to be the correct limiting value (model is no better than a random predictor, see Chicco et al. (2020), doi: 10.1186/s1286401964137).

See Also

Other confusion matrix calculation functions: `calculate_models_mcc()`, `calculate_models_synergies_fn()`, `calculate_models_synergies_fp()`, `calculate_models_synergies_tn()`, `calculate_models_synergies_tp()`

`calculate_models_mcc` *Calculate the Matthews correlation coefficient for each model*

Description

Calculate the Matthews correlation coefficient for each model

Usage

```
calculate_models_mcc(
  observed.model.predictions,
  unobserved.model.predictions,
  number.of.drug.comb.tested
)
```

Arguments

observed.model.predictions	`data.frame` object with rows the models and columns the drug combinations that were found as **synergistic** (*positive results*). Possible values for each *model-drug combination element* are either *0* (no synergy predicted), *1* (synergy was predicted) or *NA* (couldn't find stable states in either the drug combination inhibited model or in any of the two single-drug inhibited models)
unobserved.model.predictions	`data.frame` object with rows the models and columns the drug combinations that were found as **non-synergistic** (*negative results*). Possible values for each *model-drug combination element* are either *0* (no synergy predicted), *1* (synergy was predicted) or *NA* (couldn't find stable states in either the drug combination inhibited model or in any of the two single-drug inhibited models)

number.of.drug.comb.tested
: numeric. The total number of drug combinations tested, which should be equal to the sum of the columns of the observed.model.predictions and the unobserved.model.predictio

Value

a numeric vector of MCC values, each value being in the [-1,1] interval. The *names* attribute holds the models' names if applicable (i.e. the input data.frames have *rownames*).

See Also

Other confusion matrix calculation functions: calculate_mcc(), calculate_models_synergies_fn(), calculate_models_synergies_fp(), calculate_models_synergies_tn(), calculate_models_synergies_tp()

---

calculate_models_synergies_fn

Count the non-synergies of the observed synergies per model (FN)

Description

Since the given observed.model.predictions data.frame has only the positive results, this function returns the total number of 0's *and* NA's in each row.

Usage

```
calculate_models_synergies_fn(observed.model.predictions)
```

Arguments

observed.model.predictions
: data.frame object with rows the models and columns the drug combinations that were found/observed as **synergistic** (*negative results*). Possible values for each *model-drug combination element* are either *0* (no synergy predicted), *1* (synergy was predicted) or *NA* (couldn't find stable states in either the drug combination inhibited model or in any of the two single-drug inhibited models)

Value

an integer vector with elements the number of false negative predictions per model. The model names are given in the *names* attribute (same order as in the *rownames* attribute of the observed.model.predictions data.frame).

See Also

Other confusion matrix calculation functions: calculate_mcc(), calculate_models_mcc(), calculate_models_synergi
calculate_models_synergies_tn(), calculate_models_synergies_tp()

`calculate_models_synergies_fp` *Count the predictions of the non-synergistic drug combinations per model (FP)*

Description

Since the given `unobserved.model.predictions` data.frame has only the negative results, this function returns the total number of 1's in each row.

Usage

```
calculate_models_synergies_fp(unobserved.model.predictions)
```

Arguments

`unobserved.model.predictions`
: `data.frame` object with rows the models and columns the drug combinations that were found/observed as **non-synergistic** (*negative results*). Possible values for each *model-drug combination element* are either *0* (no synergy predicted), *1* (synergy was predicted) or *NA* (couldn't find stable states in either the drug combination inhibited model or in any of the two single-drug inhibited models)

Value

an integer vector with elements the number of false positive predictions per model. The model names are given in the *names* attribute (same order as in the *rownames* attribute of the unobserved.model.predictions `data.frame`).

See Also

Other confusion matrix calculation functions: `calculate_mcc()`, `calculate_models_mcc()`, `calculate_models_synergi`
`calculate_models_synergies_tn()`, `calculate_models_synergies_tp()`

`calculate_models_synergies_tn` *Count the non-synergies of the non-synergistic drug combinations per model (TN)*

Description

Since the given `unobserved.model.predictions` data.frame has only the negative results, this function returns the total number of 0's *and* NA's in each row.

Usage

```
calculate_models_synergies_tn(unobserved.model.predictions)
```

Arguments

`unobserved.model.predictions`
: `data.frame` object with rows the models and columns the drug combinations that were found/observed as **non-synergistic** (*negative results*). Possible values for each *model-drug combination element* are either *0* (no synergy predicted), *1* (synergy was predicted) or *NA* (couldn't find stable states in either the drug combination inhibited model or in any of the two single-drug inhibited models)

Value

an integer vector with elements the number of true negative predictions per model. The model names are given in the *names* attribute (same order as in the *rownames* attribute of the unobserved.model.predictions `data.frame`).

See Also

Other confusion matrix calculation functions: `calculate_mcc()`, `calculate_models_mcc()`, `calculate_models_synergi`
`calculate_models_synergies_fp()`, `calculate_models_synergies_tp()`

`calculate_models_synergies_tp` *Count the predictions of the observed synergies per model (TP)*

Description

Since the given `observed.model.predictions` data.frame has only the positive results, this function returns the total number of 1's in each row.

Usage

```
calculate_models_synergies_tp(observed.model.predictions)
```

Arguments

`observed.model.predictions`
: `data.frame` object with rows the models and columns the drug combinations that were found/observed as **synergistic** (*positive results*). Possible values for each *model-drug combination element* are either *0* (no synergy predicted), *1* (synergy was predicted) or *NA* (couldn't find stable states in either the drug combination inhibited model or in any of the two single-drug inhibited models)

Value

an integer vector with elements the number of true positive predictions per model. The model names are given in the *names* attribute (same order as in the *rownames* attribute of the observed.model.predictions `data.frame`).

See Also

Other confusion matrix calculation functions: `calculate_mcc()`, `calculate_models_mcc()`, `calculate_models_synergi`
`calculate_models_synergies_fp()`, `calculate_models_synergies_tn()`

`construct_network` *Construct igraph network graph*

Description

Use this function to create an igraph graph object based on the topology .sif file given. It automatically sets various visualization graph properties and checks if the node names from the topology file are the same as in the models inside the given `models.dir` (if not NULL).

Usage

```
construct_network(topology.file, models.dir = NULL)
```

Arguments

`topology.file`	string. The name of the .sif file (can be a full path name).
`models.dir`	string. A dir with *.gitsbe* files/models. Default value: NULL. If specified, it is used for the validation of the node names.

Value

an igraph graph object representing the network as defined in the topology file

See Also

`graph_from_data_frame`, `get_edges_from_topology_file`, `get_node_names`

Examples

```
topology.file = system.file("extdata", "example.sif", package = "emba", mustWork = TRUE)
net = construct_network(topology.file)
```

count_models_that_predict_synergies *Count models that predict a set of synergies*

Description

Use this function to find the number of models that predict a given set of drug combinations (usually the ones found as synergies).

Usage

```
count_models_that_predict_synergies(drug.comb.vec, model.predictions)
```

Arguments

`drug.comb.vec`	a character vector. Elements are (synergistic) drug combinations, each one being a string in the form *A-B* - no spaces between the names and the hyphen '-')
`model.predictions`	a `data.frame` object with rows as the models and columns the drug combinations tested. Possible values for each *model-drug combination element* are either *0* (no synergy predicted), *1* (synergy was predicted) or *NA*

Value

the number of models that predict the given drug combination set (have a value of 1 in the respective columns of the `model.predictions` data.frame). If the given set is empty, we return the number of models that predicted no synergies at all (after the *NA* values are discarded, the number of rows in the `model.predictions` data.frame that have only zero values)

emba *emba*

Description

Analysis and visualization of an ensemble of boolean models for biomarker discovery in cancer cell networks.

Details

For a complete list of functions, use `library(help = "emba")`

filter_network *Filter the network's vertices*

Description

Produce an **induced subgraph** of the given *net* igraph object. How many vertices/nodes will be kept in the result graph object is determined by the initial nodes given and the level provided. A level equal to 0 corresponds to a subgraph with only the given nodes, a level equal to 1 to a subgraph with the nodes + their neighbors (the closed neighbourhood set where every node is within 1 edge distnace from the given ones) and a level equal to 2 to a subgraph with the nodes + their neighbors + the nodes neighbor neighbors! (so the neighbourhood of the neighbourhood or every node is within 2 edges distance from the given ones).

Usage

```
filter_network(net, nodes, level)
```

Arguments

net	an igraph object.
nodes	character vector of node names. It must be a subset of the nodes of the *net* object.
level	integer. Can be only 0, 1 or 2 and specifies the neighbourhood depth of the result graph.

Value

an induced subgraph of the net igraph object.

get_alt_drugname *Get alternative drug combination name*

Description

Use this function on a string *A-B* that represents a drug combination, to get the reverse combination name - *B-A* - for testing/checking data.

Usage

```
get_alt_drugname(drug.comb)
```

Arguments

drug.comb	a string in the form *drugname.1-drugname.2* (no spaces between the names and the hyphen '-')

Value

the alternative, yet equivalent drug combination

Examples

```
drug.comb = "A-B"
alt.drug.comb = get_alt_drugname(drug.comb)
```

`get_avg_activity_diff_based_on_mcc_clustering` *Get the average activity difference based on MCC clustering*

Description

This function splits the models to 'good' and 'bad' based on an MCC value clustering method: *class.id.high* denotes the group id with the higher MCC values (good model group) vs *class.id.low* which denotes the group id with the lower MCC values (bad model group). Then, for each network node, the function finds the node's average activity in each of the two classes (a value in the [0,1] interval) and then subtracts the bad class average activity value from the good one, taking into account the given `penalty` factor and the number of models in each respective model group.

Usage

```
get_avg_activity_diff_based_on_mcc_clustering(
  models.mcc,
  models.stable.state,
  mcc.class.ids,
  models.cluster.ids,
  class.id.low,
  class.id.high,
  penalty = 0
)
```

Arguments

`models.mcc`	a numeric vector of Matthews Correlation Coefficient (MCC) scores, one for each model. The *names* attribute holds the models' names. Can be the result of using the function `calculate_models_mcc`.
`models.stable.state`	a `data.frame` (nxm) with n models and m nodes. The row names specify the models' names whereas the column names specify the network nodes (gene, proteins, etc.). Possible values for each *model-node element* can be between *0* (inactive node) and *1* (active node) inclusive.
`mcc.class.ids`	a numeric vector of group/class ids starting from 1, e.g. `c(1,2,3)` (3 MCC classes).

`models.cluster.ids`
: a numeric vector of cluster ids assigned to each model. It is the result of using `Ckmeans.1d.dp` with input the vector of the models' MCC values.

`class.id.low`
: integer. This number specifies the MCC class id of the 'bad' models.

`class.id.high`
: integer. This number specifies the MCC class id of the 'good' models and needs to be strictly higher than `class.id.low`.

`penalty`
: value between 0 and 1 (inclusive). A value of 0 means no penalty and a value of 1 is the strickest possible penalty. Default value is 0. This penalty is used as part of a weighted term to the difference in a value of interest (e.g. activity or link operator difference) between two group of models, to account for the difference in the number of models from each respective model group.

Value

a numeric vector with values in the [-1,1] interval (minimum and maximum possible average difference) and with the *names* attribute representing the name of the nodes.

Details

So, if a node has a value close to -1 it means that on average, this node is more **inhibited** in the 'good' models compared to the 'bad' ones while a value closer to 1 means that the node is more **activated** in the 'good' models. A value closer to 0 indicates that the activity of that node is **not so much different** between the 'good' and 'bad' models and so it won't not be a node of interest when searching for indicators of better performance (higher MCC score/class) in the good models.

See Also

`get_vector_diff`

Other average data difference functions: `get_avg_activity_diff_based_on_specific_synergy_prediction()`, `get_avg_activity_diff_based_on_synergy_set_cmp()`, `get_avg_activity_diff_based_on_tp_predictions()`, `get_avg_activity_diff_mat_based_on_mcc_clustering()`, `get_avg_activity_diff_mat_based_on_specific_syn`
`get_avg_activity_diff_mat_based_on_tp_predictions()`, `get_avg_link_operator_diff_based_on_synergy_set_`
`get_avg_link_operator_diff_mat_based_on_mcc_clustering()`, `get_avg_link_operator_diff_mat_based_on_sp`
`get_avg_link_operator_diff_mat_based_on_tp_predictions()`

---

get_avg_activity_diff_based_on_specific_synergy_prediction

Get average activity difference based on specific synergy prediction

---

Description

Given a specific drug combination, this function splits the models to good (those that predicted that particular combination, i.e. found it as synergistic - a value of *1* in the `model.predictions`) and bad (those that found it as non-synergistic - a value of *0* in the `model.predictions`). The models whose predicted value for that synergy is marked as *NA* are excluded from the analysis. Then, for each network node, the function finds the node's average activity in each of the two model groups (a

value in the [0,1] interval) and then subtracts the bad group's average activity value from the good one, taking into account the given `penalty` factor and the number of models in each respective model group.

Usage

```
get_avg_activity_diff_based_on_specific_synergy_prediction(
  model.predictions,
  models.stable.state,
  drug.comb,
  penalty = 0
)
```

Arguments

`model.predictions`	a `data.frame` object with rows the models and columns the drug combinations. Possible values for each *model-drug combination element* are either *0* (no synergy predicted), *1* (synergy was predicted) or *NA* (couldn't find stable states in either the drug combination inhibited model or in any of the two single-drug inhibited models)
`models.stable.state`	a `data.frame` (nxm) with n models and m nodes. The row names specify the models' names whereas the column names specify the network nodes (gene, proteins, etc.). Possible values for each *model-node element* can be between *0* (inactive node) and *1* (active node) inclusive.
`drug.comb`	string. The drug combination which will be used to split the models. It must be included in the column names of the `model.predictions` object.
`penalty`	value between 0 and 1 (inclusive). A value of 0 means no penalty and a value of 1 is the strickest possible penalty. Default value is 0. This penalty is used as part of a weighted term to the difference in a value of interest (e.g. activity or link operator difference) between two group of models, to account for the difference in the number of models from each respective model group.

Value

a numeric vector with values in the [-1,1] interval (minimum and maximum possible average difference) and with the *names* attribute representing the name of the nodes.

Details

So, if a node has a value close to -1 it means that on average, this node is more **inhibited** in the models that predicted the specific drug combination given, whereas a value closer to 1 means that the node is more **activated** in these models. A value closer to 0 indicates that the activity of that node is **not so much different** between the models that predicted the synergy and those that did not and so it won't not be a node of interest when searching for *synergy biomarkers* - nodes whose activity is important for the manifestation of the synergy.

See Also

get_vector_diff

Other average data difference functions: get_avg_activity_diff_based_on_mcc_clustering(), get_avg_activity_diff_based_on_synergy_set_cmp(), get_avg_activity_diff_based_on_tp_predictions(), get_avg_activity_diff_mat_based_on_mcc_clustering(), get_avg_activity_diff_mat_based_on_specific_syn get_avg_activity_diff_mat_based_on_tp_predictions(), get_avg_link_operator_diff_based_on_synergy_set get_avg_link_operator_diff_mat_based_on_mcc_clustering(), get_avg_link_operator_diff_mat_based_on_sp get_avg_link_operator_diff_mat_based_on_tp_predictions()

---

get_avg_activity_diff_based_on_synergy_set_cmp

Get the average activity difference based on the comparison of two synergy sets

---

Description

This function splits the models to 'good' and 'bad' based on the predictions of two different synergy sets, one of them being a subset of the other. The 'good' models are those that predict the `synergy.set.str` (e.g. "A-B,A-C,B-C") while the 'bad' models are those that predict the `synergy.subset.str` (e.g. "A-B,B-C"). Then, for each network node, the function finds the node's average activity in each of the two classes (a value in the [0,1] interval) and then subtracts the bad class average activity value from the good one, taking into account the given `penalty` factor and the number of models in the 'good' and 'bad' class respectively.

Usage

```
get_avg_activity_diff_based_on_synergy_set_cmp(
  synergy.set.str,
  synergy.subset.str,
  model.predictions,
  models.stable.state,
  penalty = 0
)
```

Arguments

`synergy.set.str`
: a string of drug combinations, comma-separated. The number of the specified combinations must be larger than the ones defined in the `synergy.subset.str` parameter. They also must be included in the tested drug combinations, i.e. the columns of the `model.predictions` parameter.

`synergy.subset.str`
: a string of drug combinations, comma-separated. There must be at least one combination defined and all of them should also be included in the `synergy.set.str` parameter.

`model.predictions`
: a `data.frame` object with rows the models and columns the drug combinations. Possible values for each *model-drug combination element* are either *0* (no synergy predicted), *1* (synergy was predicted) or *NA* (couldn't find stable states in either the drug combination inhibited model or in any of the two single-drug inhibited models)

`models.stable.state`
: a `data.frame` (nxm) with n models and m nodes. The row names specify the models' names whereas the column names specify the network nodes (gene, proteins, etc.). Possible values for each *model-node element* can be between *0* (inactive node) and *1* (active node) inclusive.

`penalty`
: value between 0 and 1 (inclusive). A value of 0 means no penalty and a value of 1 is the strickest possible penalty. Default value is 0. This penalty is used as part of a weighted term to the difference in a value of interest (e.g. activity or link operator difference) between two group of models, to account for the difference in the number of models from each respective model group.

Value

a numeric vector with values in the [-1,1] interval (minimum and maximum possible average difference) and with the names attribute representing the name of the nodes.

Details

So, if a node has a value close to -1 it means that on average, this node is more **inhibited** in the models that predicted the extra synergy(-ies) that are included in the `synergy.set.str` but not in the `synergy.subset.str`, whereas a value closer to 1 means that the node is more **activated** in these models. These nodes are **potential biomarkers** because their activity state can influence the prediction performance of a model and make it predict the extra synergy(-ies). A value closer to 0 indicates that the activity of that node is **not so much different** between the models that predicted the synergy set and those that predicted it's subset, so it won't be a node of interest when searching for potential biomarkers for the extra synergy(-ies).

See Also

`get_vector_diff`

Other average data difference functions: `get_avg_activity_diff_based_on_mcc_clustering()`, `get_avg_activity_diff_based_on_specific_synergy_prediction()`, `get_avg_activity_diff_based_on_tp_pred`
`get_avg_activity_diff_mat_based_on_mcc_clustering()`, `get_avg_activity_diff_mat_based_on_specific_syn`
`get_avg_activity_diff_mat_based_on_tp_predictions()`, `get_avg_link_operator_diff_based_on_synergy_set_`
`get_avg_link_operator_diff_mat_based_on_mcc_clustering()`, `get_avg_link_operator_diff_mat_based_on_sp`
`get_avg_link_operator_diff_mat_based_on_tp_predictions()`

`get_avg_activity_diff_based_on_tp_predictions` *Get the average activity difference based on the number of true positives*

Description

This function splits the models to 'good' and 'bad' based on the number of true positive predictions: *num.high* TPs (good) vs *num.low* TPs (bad). Then, for each network node, it finds the node's average activity in each of the two classes (a value in the [0,1] interval) and then subtracts the 'bad' average activity value from the good' one, taking into account the given `penalty` factor and the number of models in each respective model group.

Usage

```
get_avg_activity_diff_based_on_tp_predictions(
  models.synergies.tp,
  models.stable.state,
  num.low,
  num.high,
  penalty = 0
)
```

Arguments

`models.synergies.tp`	an integer vector of TP values. The *names* attribute holds the models' names and must be a subset of the row names of the `models.stable.state` parameter. Consider using the function `calculate_models_synergies_tp`.
`models.stable.state`	a `data.frame` (nxm) with n models and m nodes. The row names specify the models' names whereas the column names specify the network nodes (gene, proteins, etc.). Possible values for each *model-node element* can be between *0* (inactive node) and *1* (active node) inclusive.
`num.low`	integer. The number of true positives representing the 'bad' model class.
`num.high`	integer. The number of true positives representing the 'good' model class. This number has to be strictly higher than `num.low`.
`penalty`	value between 0 and 1 (inclusive). A value of 0 means no penalty and a value of 1 is the strickest possible penalty. Default value is 0. This penalty is used as part of a weighted term to the difference in a value of interest (e.g. activity or link operator difference) between two group of models, to account for the difference in the number of models from each respective model group.

Value

a numeric vector with values in the [-1,1] interval (minimum and maximum possible average difference) and with the *names* attribute representing the name of the nodes.

Details

So, if a node has a value close to -1 it means that on average, this node is more **inhibited** in the 'good' models compared to the 'bad' ones while a value closer to 1 means that the node is more **activated** in the 'good' models. A value closer to 0 indicates that the activity of that node is **not so much different** between the 'good' and 'bad' models and so it won't not be a node of interest when searching for indicators of better performance (higher number of true positives) in the good models.

See Also

`get_vector_diff`

Other average data difference functions: `get_avg_activity_diff_based_on_mcc_clustering()`, `get_avg_activity_diff_based_on_specific_synergy_prediction()`, `get_avg_activity_diff_based_on_synergy`
`get_avg_activity_diff_mat_based_on_mcc_clustering()`, `get_avg_activity_diff_mat_based_on_specific_syn`
`get_avg_activity_diff_mat_based_on_tp_predictions()`, `get_avg_link_operator_diff_based_on_synergy_set`
`get_avg_link_operator_diff_mat_based_on_mcc_clustering()`, `get_avg_link_operator_diff_mat_based_on_sp`
`get_avg_link_operator_diff_mat_based_on_tp_predictions()`

---

get_avg_activity_diff_mat_based_on_mcc_clustering

Get average activity difference matrix based on MCC clustering

---

Description

This function splits the Matthews correlation coefficient (MCC) scores of the models to specific groups using the **Ckmeans.1d.dp** package (groups are denoted by ids, e.g. 1,2,3, etc. where a larger id corresponds to a group of models with higher MCC scores) and for each pairwise combination of group id matchings (e.g. (0,1), (1,3), etc.), it uses the `get_avg_activity_diff_based_on_mcc_clustering` function, comparing thus all groups of models that belong to different MCC classes while taking into account the given `penalty` factor and the number of models in each respective model MCC group.

Usage

```
get_avg_activity_diff_mat_based_on_mcc_clustering(
  models.mcc,
  models.stable.state,
  num.of.mcc.classes,
  penalty = 0
)
```

Arguments

`models.mcc`	a numeric vector of Matthews Correlation Coefficient (MCC) scores, one for each model. The *names* attribute holds the models' names. Can be the result of using the function `calculate_models_mcc`.

`models.stable.state`
: a `data.frame` (nxm) with n models and m nodes. The row names specify the models' names whereas the column names specify the network nodes (gene, proteins, etc.). Possible values for each *model-node element* can be between *0* (inactive node) and *1* (active node) inclusive.

`num.of.mcc.classes`
: numeric. A positive integer larger than 2 that signifies the number of mcc classes (groups) that we should split the models MCC values.

`penalty`
: value between 0 and 1 (inclusive). A value of 0 means no penalty and a value of 1 is the strickest possible penalty. Default value is 0. This penalty is used as part of a weighted term to the difference in a value of interest (e.g. activity or link operator difference) between two group of models, to account for the difference in the number of models from each respective model group.

Value

a matrix whose rows are **vectors of average node activity state differences** between two groups of models where the classification was based on the models' MCC values. Rows represent the different classification group matchings, e.g. (1,2) means the models that belonged to the 1st group of MCC values vs the models that belonged to the 2nd group. The columns represent the network's node names. Values are in the [-1,1] interval.

See Also

`get_vector_diff`

Other average data difference functions: `get_avg_activity_diff_based_on_mcc_clustering()`, `get_avg_activity_diff_based_on_specific_synergy_prediction()`, `get_avg_activity_diff_based_on_synergy`
`get_avg_activity_diff_based_on_tp_predictions()`, `get_avg_activity_diff_mat_based_on_specific_synergy`
`get_avg_activity_diff_mat_based_on_tp_predictions()`, `get_avg_link_operator_diff_based_on_synergy_set`
`get_avg_link_operator_diff_mat_based_on_mcc_clustering()`, `get_avg_link_operator_diff_mat_based_on_sp`
`get_avg_link_operator_diff_mat_based_on_tp_predictions()`

---

get_avg_activity_diff_mat_based_on_specific_synergy_prediction

Get average activity difference matrix based on specific synergy prediction

---

Description

This function uses the `get_avg_activity_diff_based_on_specific_synergy_prediction` function on a vector of drug combinations that were observed as synergistic (e.g. by experiments) but also found as such by at least one of the models (these drug combinations are the `predicted.synergies`).

Usage

```
get_avg_activity_diff_mat_based_on_specific_synergy_prediction(
  model.predictions,
  models.stable.state,
  predicted.synergies,
  penalty = 0
)
```

Arguments

`model.predictions`
: a `data.frame` object with rows the models and columns the drug combinations. Possible values for each *model-drug combination element* are either *0* (no synergy predicted), *1* (synergy was predicted) or *NA* (couldn't find stable states in either the drug combination inhibited model or in any of the two single-drug inhibited models)

`models.stable.state`
: a `data.frame` (nxm) with n models and m nodes. The row names specify the models' names whereas the column names specify the network nodes (gene, proteins, etc.). Possible values for each *model-node element* can be between *0* (inactive node) and *1* (active node) inclusive.

`predicted.synergies`
: a character vector of the synergies (drug combination names) that were predicted by **at least one** of the models in the dataset. It must be a subset of the column names (the drug combinations) of the `model.predictions` object.

`penalty`
: value between 0 and 1 (inclusive). A value of 0 means no penalty and a value of 1 is the strickest possible penalty. Default value is 0. This penalty is used as part of a weighted term to the difference in a value of interest (e.g. activity or link operator difference) between two group of models, to account for the difference in the number of models from each respective model group.

Value

a matrix whose rows are **vectors of average node activity state differences** between two groups of models where the classification for each individual row was based on the prediction or not of a specific synergistic drug combination. The row names are the predicted synergies, one per row, while the columns represent the network's node names. Values are in the [-1,1] interval.

See Also

`get_vector_diff`

Other average data difference functions: `get_avg_activity_diff_based_on_mcc_clustering()`, `get_avg_activity_diff_based_on_specific_synergy_prediction()`, `get_avg_activity_diff_based_on_synergy`
`get_avg_activity_diff_based_on_tp_predictions()`, `get_avg_activity_diff_mat_based_on_mcc_clustering()`
`get_avg_activity_diff_mat_based_on_tp_predictions()`, `get_avg_link_operator_diff_based_on_synergy_set`
`get_avg_link_operator_diff_mat_based_on_mcc_clustering()`, `get_avg_link_operator_diff_mat_based_on_sp`
`get_avg_link_operator_diff_mat_based_on_tp_predictions()`

`get_avg_activity_diff_mat_based_on_tp_predictions` *Get average activity difference matrix based on the number of true positives*

Description

This function finds all the TP values of the models given (e.g. 0,1,2,3) and generates every pairwise combination (e.g. the group matchings: (0,1), (1,3), etc.). Then, it uses the `get_avg_activity_diff_based_on_tp_predic` function on each generated classification group matching, comparing thus all groups of models with different true positive (TP) values, while taking into account the given `penalty` factor and the number of models in each respective model group.

Usage

```
get_avg_activity_diff_mat_based_on_tp_predictions(
  models.synergies.tp,
  models.stable.state,
  penalty = 0
)
```

Arguments

`models.synergies.tp`
: an integer vector of TP values. The *names* attribute must hold the models' names. Consider using the function `calculate_models_synergies_tp`.

`models.stable.state`
: a `data.frame` (nxm) with n models and m nodes. The row names specify the models' names (same order as in the `models.synergies.tp` parameter) whereas the column names specify the network nodes (gene, proteins, etc.). Possible values for each *model-node element* can be between *0* (inactive node) and *1* (active node) inclusive.

`penalty`
: value between 0 and 1 (inclusive). A value of 0 means no penalty and a value of 1 is the strickest possible penalty. Default value is 0. This penalty is used as part of a weighted term to the difference in a value of interest (e.g. activity or link operator difference) between two group of models, to account for the difference in the number of models from each respective model group.

Value

a matrix whose rows are **vectors of average node activity state differences** between two groups of models where the classification was based on the number of true positive predictions. Rows represent the different classification group matchings, e.g. (1,2) means the models that predicted 1 TP synergy vs the models that predicted 2 TP synergies and the columns represent the network's node names. Values are in the [-1,1] interval.

See Also

get_vector_diff

Other average data difference functions: get_avg_activity_diff_based_on_mcc_clustering(), get_avg_activity_diff_based_on_specific_synergy_prediction(), get_avg_activity_diff_based_on_synergy, get_avg_activity_diff_based_on_tp_predictions(), get_avg_activity_diff_mat_based_on_mcc_clustering(), get_avg_activity_diff_mat_based_on_specific_synergy_prediction(), get_avg_link_operator_diff_based_on, get_avg_link_operator_diff_mat_based_on_mcc_clustering(), get_avg_link_operator_diff_mat_based_on_sp, get_avg_link_operator_diff_mat_based_on_tp_predictions()

---

get_avg_link_operator_diff_based_on_synergy_set_cmp

Get the average link operator difference based on the comparison of two synergy sets

---

Description

This function uses the get_avg_activity_diff_based_on_synergy_set_cmp which splits the models to 'good' and 'bad' based on the predictions of two different synergy sets, one of them being a subset of the other. The 'good' models are those that predict the `synergy.set.str` (e.g. "A-B,A-C,B-C") while the 'bad' models are those that predict the `synergy.subset.str` (e.g. "A-B,B-C"). Then, for each network node, the function finds the node's average link operator value in each of the two classes (a value in the [0,1] interval, 0 being *AND NOT* and 1 being *OR NOT*) and then subtracts the bad class average link operator value from the good one, taking into account the given `penalty` factor and the number of models in the 'good' and 'bad' class respectively.

Usage

```
get_avg_link_operator_diff_based_on_synergy_set_cmp(
  synergy.set.str,
  synergy.subset.str,
  model.predictions,
  models.link.operator,
  penalty = 0
)
```

Arguments

`synergy.set.str`
: a string of drug combinations, comma-separated. The number of the specified combinations must be larger than the ones defined in the `synergy.subset.str` parameter. They also must be included in the tested drug combinations, i.e. the columns of the `model.predictions` parameter.

`synergy.subset.str`
: a string of drug combinations, comma-separated. There must be at least one combination defined and all of them should also be included in the `synergy.set.str` parameter.

`model.predictions`
: a `data.frame` object with rows the models and columns the drug combinations. Possible values for each *model-drug combination element* are either *0* (no synergy predicted), *1* (synergy was predicted) or *NA* (couldn't find stable states in either the drug combination inhibited model or in any of the two single-drug inhibited models)

`models.link.operator`
: a `data.frame` (nxm) with n models and m nodes. The row names specify the models' names whereas the column names specify the network nodes (gene, proteins, etc.). Possible values for each *model-node element* are either *0* (**AND NOT** link operator), *1* (**OR NOT** link operator) or *0.5* if the node is not targeted by both activating and inhibiting regulators (no link operator).

`penalty`
: value between 0 and 1 (inclusive). A value of 0 means no penalty and a value of 1 is the strickest possible penalty. Default value is 0. This penalty is used as part of a weighted term to the difference in a value of interest (e.g. activity or link operator difference) between two group of models, to account for the difference in the number of models from each respective model group.

Value

a numeric vector with values in the [-1,1] interval (minimum and maximum possible average difference) and with the names attribute representing the name of the nodes.

Details

So, if a node has a value close to -1 it means that on average, this node's boolean equation has the **AND NOT** link operator in the models that predicted the extra synergy(-ies) that are included in the `synergy.set.str` but not in the `synergy.subset.str`, whereas a value closer to 1 means that the node's boolean equation has mostly the **OR NOT** link operator in these models. These nodes are potential **link operator biomarkers** because the structure of their respective boolean equations (denoted by their link operator) can influence the prediction performance of a model and make it predict the extra synergy(-ies). A value closer to 0 indicates that the link operator in the node's boolean equation is **not so much different** between the models that predicted the synergy set and those that predicted it's subset, so it won't not be a node of interest when searching for potential link operator biomarkers for the extra synergy(-ies). A value exactly equal to 0 can also mean that this node didn't not have a link operator in its boolean equation, again making it a non-important indicator of difference in model performance.

See Also

`get_vector_diff`

Other average data difference functions: `get_avg_activity_diff_based_on_mcc_clustering()`, `get_avg_activity_diff_based_on_specific_synergy_prediction()`, `get_avg_activity_diff_based_on_synergy`
`get_avg_activity_diff_based_on_tp_predictions()`, `get_avg_activity_diff_mat_based_on_mcc_clustering()`
`get_avg_activity_diff_mat_based_on_specific_synergy_prediction()`, `get_avg_activity_diff_mat_based_on`
`get_avg_link_operator_diff_mat_based_on_mcc_clustering()`, `get_avg_link_operator_diff_mat_based_on_sp`
`get_avg_link_operator_diff_mat_based_on_tp_predictions()`

`get_avg_link_operator_diff_mat_based_on_mcc_clustering`
Get average link operator difference matrix based on MCC clustering

Description

This function uses the `get_avg_activity_diff_mat_based_on_mcc_clustering` function with the parameter `models.link.operator` as input in the place of `models.stable.state`, since the two matrices representing the two inputs have the same data format (rows represent models, columns represent nodes, and each value is a number in the [0,1] interval).

Usage

```
get_avg_link_operator_diff_mat_based_on_mcc_clustering(
  models.mcc,
  models.link.operator,
  num.of.mcc.classes,
  penalty = 0
)
```

Arguments

`models.mcc`	a numeric vector of Matthews Correlation Coefficient (MCC) scores, one for each model. The *names* attribute holds the models' names. Can be the result of using the function `calculate_models_mcc`.
`models.link.operator`	a `data.frame` (nxm) with n models and m nodes. The row names specify the models' names (same order as in the `models.mcc` parameter) whereas the column names specify the network nodes (gene, proteins, etc.). Possible values for each *model-node element* are either *0* (**AND NOT** link operator), *1* (**OR NOT** link operator) or *0.5* if the node is not targeted by both activating and inhibiting regulators (no link operator).
`num.of.mcc.classes`	numeric. A positive integer larger than 2 that signifies the number of mcc classes (groups) that we should split the models MCC values.
`penalty`	value between 0 and 1 (inclusive). A value of 0 means no penalty and a value of 1 is the strickest possible penalty. Default value is 0. This penalty is used as part of a weighted term to the difference in a value of interest (e.g. activity or link operator difference) between two group of models, to account for the difference in the number of models from each respective model group.

Value

a matrix whose rows are **vectors of average node link operator differences** between two groups of models where the classification was based on the models' MCC values. Rows represent the different classification group matchings, e.g. (1,2) means the models that belonged to the 1st group

of MCC values vs the models that belonged to the 2nd group. The columns represent the network's node names. Values are in the [-1,1] interval.

Details

So, if a node has a value close to -1 it means that on average, this node's boolean equation has the **AND NOT** link operator in the 'good' models compared to the 'bad' ones while a value closer to 1 means that the node's boolean equation has mostly the **OR NOT** link operator in the 'good' models. A value closer to 0 indicates that the link operator in the node's boolean equation is **not so much different** between the 'good' and 'bad' models and so it won't not be a node of interest when searching for indicators of better performance (higher average MCC value) in the parameterization of the good models (the boolean equations). A value exactly equal to 0 can also mean that this node didn't not have a link operator in its boolean equation, again making it a non-important indicator of difference in model performance.

See Also

`get_vector_diff`

Other average data difference functions: `get_avg_activity_diff_based_on_mcc_clustering()`, `get_avg_activity_diff_based_on_specific_synergy_prediction()`, `get_avg_activity_diff_based_on_synergy`
`get_avg_activity_diff_based_on_tp_predictions()`, `get_avg_activity_diff_mat_based_on_mcc_clustering()`
`get_avg_activity_diff_mat_based_on_specific_synergy_prediction()`, `get_avg_activity_diff_mat_based_on`
`get_avg_link_operator_diff_based_on_synergy_set_cmp()`, `get_avg_link_operator_diff_mat_based_on_speci`
`get_avg_link_operator_diff_mat_based_on_tp_predictions()`

---

get_avg_link_operator_diff_mat_based_on_specific_synergy_prediction

Get average link operator difference matrix based on specific synergy prediction

---

Description

This function uses the `get_avg_activity_diff_mat_based_on_specific_synergy_prediction` function with the parameter `models.link.operator` as input in the place of `models.stable.state`, since the two matrices representing the two inputs have the same data format (rows represent models, columns represent nodes, and each value is a number in the [0,1] interval).

Usage

```
get_avg_link_operator_diff_mat_based_on_specific_synergy_prediction(
  model.predictions,
  models.link.operator,
  predicted.synergies,
  penalty = 0
)
```

Arguments

`model.predictions`
: a `data.frame` object with rows the models and columns the drug combinations. Possible values for each *model-drug combination element* are either *0* (no synergy predicted), *1* (synergy was predicted) or *NA* (couldn't find stable states in either the drug combination inhibited model or in any of the two single-drug inhibited models)

`models.link.operator`
: a `data.frame` (nxm) with n models and m nodes. The row names specify the models' names (same order as in the `model.predictions` parameter) whereas the column names specify the network nodes (gene, proteins, etc.). Possible values for each *model-node element* are either *0* (**AND NOT** link operator), *1* (**OR NOT** link operator) or *0.5* if the node is not targeted by both activating and inhibiting regulators (no link operator).

`predicted.synergies`
: a character vector of the synergies (drug combination names) that were predicted by **at least one** of the models in the dataset. It must be a subset of the column names (the drug combinations) of the `model.predictions` object.

`penalty`
: value between 0 and 1 (inclusive). A value of 0 means no penalty and a value of 1 is the strickest possible penalty. Default value is 0. This penalty is used as part of a weighted term to the difference in a value of interest (e.g. activity or link operator difference) between two group of models, to account for the difference in the number of models from each respective model group.

Value

a matrix whose rows are **vectors of average node link operator differences** between two groups of models where the classification for each individual row was based on the prediction or not of a specific synergistic drug combination. The row names are the predicted synergies, one per row, while the columns represent the network's node names. Values are in the [-1,1] interval.

Details

So, if a node has a value close to -1 it means that on average, this node's boolean equation has the **AND NOT** link operator in the models that predicted the specified synergy while a value closer to 1 means that the node's boolean equation has mostly the **OR NOT** link operator in these models. A value closer to 0 indicates that the link operator in the node's boolean equation is **not so much different** between the models that predicted the synergy and those that did not and so it won't not be a node of interest when searching for *synergy biomarkers* - nodes whose parameterization (value of the link operator) affects the manifestation of synergy. A value exactly equal to 0 can also mean that this node didn't not have a link operator in its boolean equation (making it thus a non-important node with regard to the parameterization).

See Also

`get_vector_diff`

Other average data difference functions: `get_avg_activity_diff_based_on_mcc_clustering()`, `get_avg_activity_diff_based_on_specific_synergy_prediction()`, `get_avg_activity_diff_based_on_synergy`

get_avg_activity_diff_based_on_tp_predictions(), get_avg_activity_diff_mat_based_on_mcc_clustering(),
get_avg_activity_diff_mat_based_on_specific_synergy_prediction(), get_avg_activity_diff_mat_based_on_
get_avg_link_operator_diff_based_on_synergy_set_cmp(), get_avg_link_operator_diff_mat_based_on_mcc_c
get_avg_link_operator_diff_mat_based_on_tp_predictions()

`get_avg_link_operator_diff_mat_based_on_tp_predictions` *Get average link operator difference matrix based on the number of true positives*

Description

This function uses the `get_avg_activity_diff_mat_based_on_tp_predictions` function with the parameter `models.link.operator` as input in the place of `models.stable.state`, since the two matrices representing the two inputs have the same data format (rows represent models, columns represent nodes, and each value is a number in the [0,1] interval).

Usage

```
get_avg_link_operator_diff_mat_based_on_tp_predictions(
  models.synergies.tp,
  models.link.operator,
  penalty = 0
)
```

Arguments

`models.synergies.tp`	an integer vector of TP values. The *names* attribute must hold the models' names. Consider using the function `calculate_models_synergies_tp`.
`models.link.operator`	a `data.frame` (nxm) with n models and m nodes. The row names specify the models' names (same order as in the `models.synergies.tp` parameter) whereas the column names specify the network nodes (gene, proteins, etc.). Possible values for each *model-node element* are either *0* (**AND NOT** link operator), *1* (**OR NOT** link operator) or *0.5* if the node is not targeted by both activating and inhibiting regulators (no link operator).
`penalty`	value between 0 and 1 (inclusive). A value of 0 means no penalty and a value of 1 is the strickest possible penalty. Default value is 0. This penalty is used as part of a weighted term to the difference in a value of interest (e.g. activity or link operator difference) between two group of models, to account for the difference in the number of models from each respective model group.

Value

a matrix whose rows are **vectors of average node link operator differences** between two groups of models based on some kind of classification (e.g. number of TP predictions) and whose names are set in the `rownames` attribute of the matrix (usually denoting the different classification groups, e.g. (1,2) means the models that predicted 1 TP synergy vs the models that predicted 2 TP synergies, if the classification is done by number of TP predictions). The columns represent the network's node names. Values are in the [-1,1] interval.

Details

So, if a node has a value close to -1 it means that on average, this node's boolean equation has the **AND NOT** link operator in the 'good' models compared to the 'bad' ones while a value closer to 1 means that the node's boolean equation has mostly the **OR NOT** link operator in the 'good' models. A value closer to 0 indicates that the link operator in the node's boolean equation is **not so much different** between the 'good' and 'bad' models and so it won't not be a node of interest when searching for indicators of better performance (higher number of true positives) in the parameterization of the good models (the boolean equations). A value exactly equal to 0 can also mean that this node didn't not have a link operator in its boolean equation, again making it a non-important indicator of difference in model performance.

See Also

`get_vector_diff`

Other average data difference functions: `get_avg_activity_diff_based_on_mcc_clustering()`, `get_avg_activity_diff_based_on_specific_synergy_prediction()`, `get_avg_activity_diff_based_on_synergy`
`get_avg_activity_diff_based_on_tp_predictions()`, `get_avg_activity_diff_mat_based_on_mcc_clustering()`
`get_avg_activity_diff_mat_based_on_specific_synergy_prediction()`, `get_avg_activity_diff_mat_based_on`
`get_avg_link_operator_diff_based_on_synergy_set_cmp()`, `get_avg_link_operator_diff_mat_based_on_mcc_c`
`get_avg_link_operator_diff_mat_based_on_specific_synergy_prediction()`

---

get_biomarkers *Get total biomarkers from average data differences matrix*

Description

Use this function to find all biomarkers across multiple performance classification group matchings based on a given threshold between 0 and 1.

Usage

```
get_biomarkers(diff.mat, threshold)
```

Arguments

`diff.mat`	a matrix whose rows are vectors of average node data differences between two groups of models based on some kind of classification (e.g. number of TP predictions) and whose names are set in the `rownames` attribute of the matrix (usually denoting the different classification groups, e.g. (1,2) means the models that predicted 1 TP synergy vs the models that predicted 2 TP synergies, if the classification is done by number of TP predictions). The columns represent the network's node names.
`threshold`	numeric. A number in the [0,1] interval, above which (or below its negative value) a biomarker will be registered in the returned result. Values closer to 1 translate to a more strict threshold and thus less biomarkers are found.

Value

a list with two elements:

- `biomarkers.pos`: a character vector that includes the node names of the *positive* biomarkers
- `biomarkers.neg`: a character vector that includes the node names of the *negative* biomarkers

Details

This function uses the `get_biomarkers_per_type` function to get the biomarkers (nodes) of both types (positive and negative) from the average data differences matrix. The logic behind the biomarker selection is that if there is at least one value in a column of the `diff.mat` matrix that surpasses the threshold given, then the corresponding node (name of the column) is returned as a biomarker. This means that for a single node, if at least one value that represents an average data difference (for example, the average activity state difference) between any of the given classification group comparisons is above the given threshold (or below the negative symmetric threshold), then a *positive* (*negative*) biomarker is reported.

In the case of a node which is found to surpass the significance threshold level given *both negatively and positively*, we will keep it as a biomarker in the category which corresponds to the **comparison of the highest classification groups**. For example, if the data comes from a model performance classification based on the MCC score and in the comparison of the MCC classes (1,3) the node of interest had an average difference of *-0.89* (a negative biomarker) while for the comparison of the (3,4) MCC classes it had a value of *0.91* (a positive biomarker), then we will keep that node *only as a positive biomarker*. The logic behind this is that the 'higher' performance-wise are the classification groups that we compare, the more sure we are that the average data difference corresponds to a *better indicator* for the type of the biomarker found.

See Also

Other biomarker functions: `get_biomarkers_per_type()`

`get_biomarkers_per_type` *Get biomarkers from average data differences matrix (per type)*

Description

Use this function to find either positive or negative biomarkers across multiple performance classification group matchings based on a given threshold between 0 and 1.

Usage

```
get_biomarkers_per_type(diff.mat, threshold, type)
```

Arguments

`diff.mat`	a matrix whose rows are vectors of average node data differences between two groups of models based on some kind of classification (e.g. number of TP predictions) and whose names are set in the `rownames` attribute of the matrix (usually denoting the different classification groups, e.g. (1,2) means the models that predicted 1 TP synergy vs the models that predicted 2 TP synergies, if the classification is done by number of TP predictions). The columns represent the network's node names.
`threshold`	numeric. A number in the [0,1] interval, above which (or below its negative value) a biomarker will be registered in the returned result. Values closer to 1 translate to a more strict threshold and thus less biomarkers are found.
`type`	character. Accepted values are *positive* or *negative*.

Details

The logic behind the biomarker selection is that if there is at least one value in a column of the `diff.mat` matrix that surpasses the threshold given, then the corresponding node (name of the column) is return as a biomarker. This means that for a single node, if at least one value that represents an average data difference (for example, the average activity state difference) between any of the given classification group comparisons is above the given threshold (or below the negative symmetric threshold), then a *positive* (*negative*) biomarker is reported.

Value

a character vector that includes the node names that were found either as *positive* or *negative*.

See Also

Other biomarker functions: `get_biomarkers()`

get_edges_from_topology_file

Get the edges from a specified topology

Description

Use this function to read a topology .sif file (either space or tab-delimited) and get a matrix of network edges specifying the source and target name, the regulation effect (activation or inhibition) and the color (green or red) of each interaction.

Usage

```
get_edges_from_topology_file(topology.file)
```

Arguments

`topology.file`	string. The name of the .sif file (can be a full path name).

Value

a matrix with as many rows as in the .sif topology file (each row is an edge) and 4 columns defining the source and target node name, the regulation (activation or inhibition) and the color (green or red) of the signed interaction.

Examples

```
topology.file = system.file("extdata", "example.sif", package = "emba", mustWork = TRUE)
edges = get_edges_from_topology_file(topology.file)
```

get_fitness_from_models_dir

Load the models fitness scores

Description

Use this function to merge the fitness scores from all models into a single vector (the fitness score is a value between 0 and 1 and denotes how close was the model fitted to one or more training data observations). Each model's fitness value is loaded from the respective *.gitsbe* file that can be found inside the given `models.dir` directory (other kind of files are discarded).

Usage

```
get_fitness_from_models_dir(models.dir)
```

Arguments

`models.dir`	string. A dir with *.gitsbe* files/models

Value

a numeric vector with elements the fitness scores and the names of the models included in the *names* attribute.

Examples

```
models.dir = system.file("extdata", "models", package = "emba", mustWork = TRUE)
models.fitness = get_fitness_from_models_dir(models.dir)
```

`get_link_operators_from_models_dir` *Load the models boolean equation link operator data*

Description

Use this function to merge the link operator data used in the boolean equations of the models into a single `data.frame` object. Every boolean model is defined by a series of boolean equations in the form $Target* = (ActivatororActivatoror...)andnot(InhibitororInhibitoror...)$". The **link operator** can be either *and not*, *or not* or non-existent if the target has only activating regulators or only inhibiting ones (the *not* remains in the latter case). The models are loaded from *.gitsbe* files (and only these) that can be found inside the given `models.dir` directory.

Usage

```
get_link_operators_from_models_dir(
  models.dir,
  remove.equations.without.link.operator = TRUE
)
```

Arguments

`models.dir`	string. A directory path with *.gitsbe* files/models. **Do not** include the ending path character in the string (/).
`remove.equations.without.link.operator`	logical. Should we keep the nodes (columns in the returned matrix) which do not have both type of regulators (so no link operator)? Default value: TRUE (remove these nodes).

Value

a `data.frame` (nxm) with n models and m nodes. The row names specify the models' names whereas the column names specify the network nodes (gene, proteins, etc.). Possible values for each *model-node element* are either *0* (**and not** link operator), *1* (**or not** link operator) or *0.5* if the node is not targeted by both activating and inhibiting regulators (no link operator).

Examples

```
models.dir = system.file("extdata", "models", package = "emba", mustWork = TRUE)
models.link.operator = get_link_operators_from_models_dir(models.dir)
models.link.operator.with.extra.nodes =
  get_link_operators_from_models_dir(models.dir, FALSE)
```

---

get_models_based_on_mcc_class_id

Get models based on the MCC class id

Description

This helper function finds all the models that belong to a specific MCC cluster, i.e. their MCC values belong to the same cluster id.

Usage

```
get_models_based_on_mcc_class_id(class.id, models.cluster.ids, models.mcc)
```

Arguments

`class.id`	an integer specifying the class id.
`models.cluster.ids`	a numeric vector of cluster ids assigned to each model. It is the result of using `Ckmeans.1d.dp` with input the sorted vector of the models' MCC values.
`models.mcc`	a numeric sorted vector of Matthews Correlation Coefficient (MCC) scores, one for each model. The *names* attribute holds the models' names.

Value

a character vector of model names

get_model_names *Get the model names*

Description

Get the model names

Usage

```
get_model_names(models.dir)
```

Arguments

models.dir	string. A directory with *.gitsbe* files/models (non-gitsbe files are disregarded).

Value

a character vector of the model names, corresponding to the names of the *.gitsbe* files (the extension is pruned).

Examples

```
models.dir = system.file("extdata", "models", package = "emba", mustWork = TRUE)
models = get_model_names(models.dir)
```

get_model_predictions *Load the models predictions data*

Description

Use this function to read a file that has the model predictions data and output it to a `data.frame` object.

Usage

```
get_model_predictions(model.predictions.file)
```

Arguments

model.predictions.file	a tab-delimited file (for the specific format check the example below)

Value

a `data.frame` object with rows the models and columns the drug combinations. Possible values for each *model-drug combination element* are either *0* (no synergy predicted), *1* (synergy was predicted) or *NA* (couldn't find stable states in either the drug combination inhibited model or in any of the two single-drug inhibited models)

Examples

```
model.predictions.file = system.file("extdata", "model_predictions",
  package = "emba", mustWork = TRUE)
model.predictions = get_model_predictions(model.predictions.file)
```

`get_neighbors` *Get neighbor nodes*

Description

Given an igraph network object and vector of node names, this function returns the set of unique neighbor nodes considering both ingoing and outgoing edges (the closed neighbourhood node set).

Usage

```
get_neighbors(net, nodes)
```

Arguments

`net`	igraph object
`nodes`	character vector of node names

Value

a character vector of all the unique neighbors of the given `nodes` in the `net` graph.

`get_node_colors` *Get the node colors*

Description

This function splits the [-1,1] interval into **2000** smaller ones and matches each value of the `diff` vector to a specific hex color code, using a spline interpolation of the colors as defined in the `col` parameter.

Usage

```
get_node_colors(net, diff, col)
```

Arguments

net	an igraph graph object with the node names defined in V(net)$name
diff	numeric vector. Every value is in the [-1,1] interval and represents the average activity difference of each node. The node names have to be specified in the *names* attribute of the given diff vector and have to be the same as in V(net)$name.
col	a character vector of colors to do the color interpolation in the [-1,1] interval. Usually a two-element vector specifying the colors matching the start and end of the interval (-1 and 1 respectively) or a three-element vector specifying the colors matching the values -1, 0 and 1 (can be more of course, you get the idea).

Value

a character vector of hex color codes where the *names* attribute corresponds to the nodes of the given igraph object. Will be used to fill in the V(net)$color property of the net object. If there are nodes that are part of the network object net but not present in the diff vector, then a *NA* value will be given for the color of these nodes.

---

get_node_names *Get the node names*

Description

This function uses the first *.gitsbe* file that it finds inside the given directory to output a vector of the network node names (which should be the same for every model)

Usage

```
get_node_names(models.dir)
```

Arguments

models.dir	string. A directory with at least one *.gitsbe* file/model.

Value

a character vector of the node names (protein and/or gene names)

Examples

```
models.dir = system.file("extdata", "models", package = "emba", mustWork = TRUE)
nodes = get_node_names(models.dir)
```

get_observed_model_predictions

Subset the model predictions to the (true) observed synergies

Description

Subset the model predictions to the (true) observed synergies

Usage

```
get_observed_model_predictions(model.predictions, observed.synergies)
```

Arguments

model.predictions	a data.frame object with rows the models and columns the drug combinations. Possible values for each *model-drug combination element* are either *0* (no synergy predicted), *1* (synergy was predicted) or *NA* (couldn't find stable states in either the drug combination inhibited model or in any of the two single-drug inhibited models)
observed.synergies	a character vector with elements the names of the drug combinations that were found as synergistic

Value

a data.frame object with rows the models and columns the drug combinations that were found/observed as **synergistic** (*positive results*). Possible values for each *model-drug combination element* are either *0* (no synergy predicted), *1* (synergy was predicted) or *NA* (couldn't find stable states in either the drug combination inhibited model or in any of the two single-drug inhibited models)

get_observed_synergies

Load the observed synergies data

Description

Use this function to read a file that has the observed synergies data and output it to a character vector. If drug.combinations.tested is NULL (the default), no data validation is done, otherwise we check that the observed synergies are indeed a subset of the tested drug combinations.

Usage

```
get_observed_synergies(file, drug.combinations.tested = NULL)
```

Arguments

`file`	string. The name of the file, can be a full path. See example below for the format of an observed synergies file.
`drug.combinations.tested`	a character vector with drug combinations as elements. Default value: NULL.

Value

a character vector with elements the names of the drug combinations that were found as synergistic

Examples

```
observed.synergies.file = system.file("extdata", "observed_synergies",
  package = "emba", mustWork = TRUE)
observed.synergies = get_observed_synergies(observed.synergies.file)
```

`get_observed_synergies_per_cell_line` *Get observed synergies per cell line*

Description

Use this function to get the observed synergies from the respective files inside the given list of cell line directories.

Usage

```
get_observed_synergies_per_cell_line(cell.line.dirs, drug.combos)
```

Arguments

`cell.line.dirs`	a character vector of the cell line directories, in the form of *{path}/cell_line_name*. The cell line name directory should be different for each element of the vector as we use it to fill in the `rownames` of the result `data.frame` object. Inside each cell line directory we read the observed synergies from a file called *observed_synergies* (if it exists and is non-empty). This file has the names of the observed drug combinations, one in each line.
`drug.combos`	a character vector with elements the names of all the drug combinations that were tested in the analysis.

Value

a data.frame, whose columns represent the drug combinations tested and the rows the cell lines. Possible values for each *cell line-drug combination* element are either *1* (an observed synergy) or *0* (non-observed synergy).

`get_perf_biomarkers_per_cell_line` *Get performance biomarkers per cell line*

Description

Use this function to get the performance biomarkers from the respective files inside the given list of directories.

Usage

```
get_perf_biomarkers_per_cell_line(biomarkers.dirs, node.names)
```

Arguments

`biomarkers.dirs`	a character vector of the biomarker directories, in the form of *{path}/cell_line_name/{dir}*. The cell line name directory should be different for each element of the vector as we use it to fill in the `rownames` of the result `data.frame` object. Inside each *{dir}* (the directory name does not matter, but 'biomarkers' is a good choice), we read the biomarkers from two files (if they exist and are non-empty): *biomarkers_active* and *biomarkers_inhibited*, which have the active and inhibited performance biomarkers for each cell line (these files have a list of node names/biomarkers, one in each line).
`node.names`	a character vector of the node names used in the analysis. The biomarker names taken from the files inside the given directories must be a subset of this vector.

Value

a data.frame, whose columns represent the network nodes and the rows the cell lines. Possible values for each *cell line-node* element are either *1* (*active state* biomarker), *-1* (*inhibited state* biomarker) or *0* (not a biomarker).

Examples

```
dir = system.file("extdata", "AGS", "bio", package = "emba", mustWork = TRUE)
res = get_perf_biomarkers_per_cell_line(biomarkers.dirs = c(dir),
  node.names = paste0("x", 1:20))
```

`get_stable_state_from_models_dir` *Load the models stable state data*

Description

Use this function to merge the stable states from all boolean models into a single `data.frame` object. The models stable states are loaded from *.gitsbe* files that can be found inside the given `models.dir` directory.

Usage

```
get_stable_state_from_models_dir(models.dir, all.ss = FALSE)
```

Arguments

`models.dir`	string. A directory with *.gitsbe* files/models. **Do not** include the ending path character in the string (*/*). Only files that include the string *gitsbe* are parsed.
`all.ss`	logical. Should all stable states be included in the returned object? Default value is *FALSE* (only the 1 stable state models are included).

Value

The format of the returned object depends on the `all.ss` value. If:

- `all.ss` is *FALSE* (default): a `data.frame` (nxm) with n models and m nodes. The row names specify the models names (taken from the file names without the *gitsbe* extension) whereas the column names specify the name of the network nodes (gene, proteins, etc.). Possible values for each *model-node element* are either *0* (inactive node) or *1* (active node). If a *.gitsbe* file/model has zero (0) or more than 1 stable states, a diagnostic message is printed and the corresponding model is discarded, i.e. it will not be included in the returned `data.frame` object.
- `all.ss` is *TRUE*: a `tibble` object where each row stores a separate stable state and the columns correspond to network nodes (as before) with an extra last column that has the name of the model that produced that stable state. As such, models that have multiple stable states will occupy several rows with the last column having the same name/model. Models with no stable states are discarded.

Examples

```
models.dir = system.file("extdata", "models", package = "emba", mustWork = TRUE)
models.stable.state = get_stable_state_from_models_dir(models.dir)
models.stable.state = get_stable_state_from_models_dir(models.dir, all.ss = TRUE)
```

get_synergy_biomarkers_from_dir *Get synergy biomarkers from dir*

Description

This function reads the synergy biomarker files inside the given directory and merges the results into a `data.frame` which it returns. This functions should be used when the synergy biomarker results are in separate files inside the directory given (see `biomarkers.dir` parameter).

Usage

```
get_synergy_biomarkers_from_dir(
  predicted.synergies,
  biomarkers.dir,
  models.dir = NULL,
  node.names = NULL
)
```

Arguments

`predicted.synergies`	a character vector of the synergies (drug combination names) that were predicted by **at least one** of the models in the dataset.
`biomarkers.dir`	string. It specifies the full path name of the directory which holds the biomarker files (without the ending character */*). The biomarker files must be formatted as: *%drug.comb%_biomarkers_active* or *%drug.comb%_biomarkers_inhibited*, where %drug.comb% is an element of the `predicted.synergies` vector.
`models.dir`	string. A directory with *.gitsbe* files/models. It's needed in order to call `get_node_names`.
`node.names`	a character vector which has the names of the nodes. If it's not NULL, then it will be used instead of the `models.dir` parameter. The `node.names` should include all the nodes that are reported as biomarkers in the biomarker files inside the `biomarkers.dir` directory. Note that the biomarker nodes in the files will be included in the returned `data.frame` object no matter the `node.names` specified. Default value: NULL.

Value

a data.frame, whose columns represent the network nodes and the rows the predicted synergies. Possible values for each *synergy-node* element are either *1* (*active state* biomarker), *-1* (*inhibited state* biomarker) or *0* (not a biomarker or the node is not at all present in the network or the drug combination is not a synergistic one).

get_synergy_biomarkers_per_cell_line *Get synergy biomarkers per cell line*

Description

Use this function to get the synergy biomarkers for each cell line. The biomarkers must be stored in a single file inside each given cell line-specific directory.

Usage

```
get_synergy_biomarkers_per_cell_line(biomarkers.dirs)
```

Arguments

`biomarkers.dirs`
: a character vector of the biomarker directories, in the form of *{path}/cell_line_name/{dir}*. The cell line name directory should be different for each element of the vector as we use it to fill in the `rownames` of each cell line-specific `data.frame` object. Inside each *{dir}* (the directory name does not matter, but 'biomarkers' is a good choice), we read the synergy biomarkers from a file (if it exists and is non-empty) with the name *biomarkers_per_synergy*. This file has as first row the node names (columns) while every next row starts with the row name (drug combination name) followed by a series of numbers from the ternary set {1,-1,0}, denoting thus which nodes where found as active biomarkers for that synergy, inhibited or not at all as biomarkers.

Value

a list of cell line-specific data frames (each element from the list takes its name from the respective cell line). Each cell-line specific `data.frame` object has as rows the **true positive predicted synergies** for that particular cell line and columns the network nodes (should be the same for all cell lines). Possible values for each *synergy-node* element in each cell line-specific `data.frame` are either *1* (*active state* biomarker), *-1* (*inhibited state* biomarker) or *0* (not a biomarker).

Examples

```
dir = system.file("extdata", "AGS", "bio", package = "emba", mustWork = TRUE)
res_list = get_synergy_biomarkers_per_cell_line(biomarkers.dirs = c(dir))
```

get_synergy_comparison_sets *Get synergy comparison sets*

Description

This helper function identifies pairs of (*set*, *subset*) for each synergy (implicitly given through the `synergy.subset.stats` object) where each respective *subset* misses just one synergy from the larger *set*.

Usage

```
get_synergy_comparison_sets(synergy.subset.stats)
```

Arguments

`synergy.subset.stats`	integer vector with values the amount of models that predicted each synergy subset, defined as a comma-separated string of drug combinations in the *names* attribute of the vector. It can be the result of using the function `get_synergy_subset_stats`.

Value

`data.frame` object with 3 columns. For each row, the 1st column defines a *single synergy* of interest (e.g. drug combination "A-B"), the 2nd a *synergy set* that includes the single one (e.g. the set "F-G,A-B,C-D") and the 3rd the *synergy subset* of the *set* that does not include the single synergy of the first column (e.g. "F-G,C-D").

get_synergy_scores *Get synergy scores from file*

Description

Use this function to read Drabme's output files that have synergy scores for a list of tested drug perturbations.

Usage

```
get_synergy_scores(file_name, file_type = "ensemblewise")
```

Arguments

`file_name`	string. The name of the file, can be a full path.
`file_type`	string. The type of input file, can be either *ensemblewise* (default option) or *modelwise*.

Details

Two types of files can be read: the *model-wise* and the *ensemble-wise* synergies:

- The *model-wise* synergies data is structured as a 3-column table, first being the names of the tested drug combinations, second the number of models that predicted that combination as synergistic and the third the number of models that predicted the drug combination as non-synergistic.
- The *ensemble-wise* synergies data is structured as a 2-column table, the first being the names of drug combinations and the second the ensemble-wise synergy scores for those perturbations.

Note that no matter the file type, the first line of the file is always skipped and the columns must be *tab-separated*. See example below for the format of such files generated by the Drabme module.

Value

a `tibble` containing a representation of the data in the file.

Examples

```
ensemblewise_synergies_file = system.file("extdata", "ensemblewise_synergies",
  package = "emba", mustWork = TRUE)
modelwise_synergies_file = system.file("extdata", "modelwise_synergies",
  package = "emba", mustWork = TRUE)
data_ensemble = get_synergy_scores(ensemblewise_synergies_file) # file_type = "ensemblewise"
data_modelwise = get_synergy_scores(modelwise_synergies_file, file_type = "modelwise")
```

---

get_synergy_subset_stats

Find the number of predictive models for every synergy subset

---

Description

Use this function to find for each possible subset of drug combinations out of a given list of synergies, the number of models that predicted it given the models' predictions. So, if for example the set of synergies is this one: {'A-B','C-D','E-F'}, we want to know how many models predicted none of them, just the single subsets (e.g. the {'A-B'}), the two-element subsets (e.g. the {'A-B','C-D'}) and all 3 of them.

Usage

```
get_synergy_subset_stats(model.predictions, synergies)
```

Arguments

`model.predictions`
: a `data.frame` object with rows the models and columns the drug combinations. Possible values for each *model-drug combination element* are either *0* (no synergy predicted), *1* (synergy was predicted) or *NA* (couldn't find stable states in either the drug combination inhibited model or in any of the two single-drug inhibited models).

`synergies`
: a character vector with elements the synergistic drug combinations. Note that these synergies should be a subset of the column names of the `model.predictions` data.frame.

Value

an integer vector with elements the number of models the predicted each synergy subset. The *names* attribute has the names of each synergistic drug combination subset, which are the drug combinations comma separated (e.g. 'A-B,C-D').

Details

Note that if the `synergies` vector has more than 10-15 elements, then this function might take long time to execute even with an optimal implementation of `count_models_that_predict_synergies`.

`get_unobserved_model_predictions` *Subset the model predictions to the (false) non-observed synergies*

Description

Subset the model predictions to the (false) non-observed synergies

Usage

```
get_unobserved_model_predictions(model.predictions, observed.synergies)
```

Arguments

`model.predictions`
: a `data.frame` object with rows the models and columns the drug combinations. Possible values for each *model-drug combination element* are either *0* (no synergy predicted), *1* (synergy was predicted) or *NA* (couldn't find stable states in either the drug combination inhibited model or in any of the two single-drug inhibited models)

`observed.synergies`
: a character vector with elements the names of the drug combinations that were found as synergistic

Value

a `data.frame` object with rows the models and columns the drug combinations that were found/observed as **non-synergistic** (*negative results*). Possible values for each *model-drug combination element* are either *0* (no synergy predicted), *1* (synergy was predicted) or *NA* (couldn't find stable states in either the drug combination inhibited model or in any of the two single-drug inhibited models)

`get_vector_diff` *Calculate difference vector with penalty term*

Description

This function calculates the difference between two given numeric vectors while adding a penalty term (weight) to account for the number of models/instances that each vector's values were calculated from. Thus, if the models/instances are disproportionate and a penalty is included, the difference vector's values will be changed accordingly to reflect that.

Usage

```
get_vector_diff(vec1, vec2, m1 = 1, m2 = 1, penalty = 0)
```

Arguments

`vec1`	numeric vector
`vec2`	numeric vector
`m1`	integer > 0
`m2`	integer > 0
`penalty`	value between 0 and 1 (inclusive). A value of 0 means no penalty (`m1`,`m2` don't matter) and a value of 1 is the strickest possible penalty. Default value is 0.

Value

the vector of differences between the two given vectors based on the formula:

$$(vec1 - vec2) * w$$

, where $w = (min(m1, m2)/max(m1, m2))^p$ and $p = penalty$.

See also related StackOverflow question. If `vec1` has `names`, the returned vector will have the same names attribute as `vec1`.

get_x_axis_values *Get the refined x-axis values*

Description

This function returns the x-axis values that are going to be used by make_barplot_on_models_stats to render the bar plot.

Usage

```
get_x_axis_values(models.stats, there.is.one.NaN.category, cont.values)
```

Arguments

models.stats	table object, the result of using table on a (numeric) vector. Usually it represents some models statistics summary - counts for each TP prediction value for example.
there.is.one.NaN.category	logical. Is there one *NaN* category? (check is done before on the *names* attribute of the models.stats)
cont.values	logical. If TRUE, the values of the x-axis will be trimmed to 3 digits after the decimal point. Otherwise, they will be returned as they are.

is_comb_element_of *Is drug combination element of given vector?*

Description

Use this function to determine if a drug combination is part of a vector of other drug combinations. We take care only of pair-wise drug combinations and an internal check is done for alternative drug names, e.g. we check if *A-B* combination is included, but also for *B-A*.

Usage

```
is_comb_element_of(drug.comb, comb.vector)
```

Arguments

drug.comb	a string in the form *A-B* (no spaces between the names and the hyphen '-')
comb.vector	a character vector of drug combinations, each one in the form *drugname.1-drugname.2*

Value

logical, depending if the drug combination is element of the given vector or not.

Examples

```
# TRUE
is_comb_element_of("A-B", c("E-F", "A-B"))
is_comb_element_of("B-A", c("E-F", "A-B"))

# FALSE
is_comb_element_of("A-B", c("E-F", "A-D"))
is_comb_element_of("A-B", c())
```

`make_barplot_on_models_stats` *Bar plot of model stats*

Description

Use this function to produce a bar plot when the input is the result of using the table function to a numeric vector

Usage

```
make_barplot_on_models_stats(
  models.stats,
  cell.line = NULL,
  title,
  xlab,
  ylab,
  cont.values = FALSE,
  threshold = 0,
  ylim.add = 0
)
```

Arguments

`models.stats`	table object, the result of using table on a (numeric) vector. Usually it represents some models statistics summary - counts for each TP prediction value for example.
`cell.line`	string. The name of the cell line to be used in the title of the produced plot. Default value: NULL (the cell line name will not be added to the title)
`title`	string. The title of the plot
`xlab`	string. The title of the x-axis
`ylab`	string. The title of the y-axis
`cont.values`	logical. If TRUE, the values of the x-axis will be trimmed to 3 digits after the decimal point. Default value: FALSE.

`threshold` integer. Values from the `model.stats` that are *less or equal* to the threshold will be pruned. Use it when there too many categories and the figure appears too dense. Default value: 0

`ylim.add` integer. Signifies the height to add to the upper `ylim` parameter on the barplot, in addition to the maximum bar height across the whole plot. Default value is 0.

Examples

```
x = c(rep(1,100), rep(2,423), rep(3,231), rep(NaN,531))
make_barplot_on_models_stats(models.stats = table(x, useNA = "ifany"),
title = "True Positives Distribution across models",
xlab = "Number of TP values", ylab = "Number of models")
```

---

make_barplot_on_synergy_subset_stats

Bar plot of observed synergy subsets

Description

Use this function to easily make a barplot that shows the amount of models that predicted each synergy subset out of the set of all observed synergies.

Usage

```
make_barplot_on_synergy_subset_stats(
  synergy.subset.stats,
  threshold.for.subset.removal,
  bottom.margin,
  ylim.add = 0,
  cell.line = NULL
)
```

Arguments

`synergy.subset.stats` integer vector with values the amount of models that predicted each synergy subset, defined as a comma-separated string of drug combinations in the *names* attribute of the vector

`threshold.for.subset.removal` integer. Use it to discard elements of the `synergy.subset.stats` vector that are strictly less than the specified threshold

`bottom.margin` integer used to vertically fit in the names of the drug combinations in the x-axis (specified in inches). The best `bottom.margin` value depends on the *maximum size* of a synergy subset as defined in the `names` attribute of the `synergy.subset.stats`. Some rules of thumb are: size = 1 => bottom.margin = 4, size = 2 => bottom.margin = 6, size = 3 => bottom.margin = 9, size = 4 => bottom.margin = 12, etc.

ylim.add	integer. Signifies the height to add to the upper ylim parameter on the barplot, in addition to the maximum bar height across the whole plot. Default value is 0.
cell.line	string. The name of the cell line to be used in the title of the produced plot. Default value: NULL (the cell line name will not be added to the title).

Examples

```
synergy.subset.stats = c(1,4,3,2)
names(synergy.subset.stats) = c("A-B", "B-C", "C-A", "C-D")
make_barplot_on_synergy_subset_stats(synergy.subset.stats,
threshold.for.subset.removal = 0, bottom.margin = 4, ylim.add = 0.5)
```

---

plot_avg_link_operator_diff_graph

Plot the graph of average link operator differences (igraph)

Description

This function uses the plot.igraph package to plot a network of nodes. The nodes are positioned according to the specified coordinates given by the layout parameter and the colors are derived using the diff values and the get_node_colors function. The color of each node indicates if the node's boolean function has on average the *AND NOT* or the *OR NOT* link operator when comparing the average model classified in the 'good' category vs the average bad' one. A non-colored node (white) will indicate nodes that do not have the link operator in their respective boolean equation (where they function as the target).

Usage

```
plot_avg_link_operator_diff_graph(net, diff, layout = NULL, title)
```

Arguments

net	igraph graph object
diff	numeric vector. Every value is in the [-1,1] interval and represents the average link operator value difference of each node. The node names have to be specified in the *names* attribute of the given vector. For example, diff could be the result of using the function get_avg_link_operator_diff_mat_based_on_tp_predictions and getting one vector row from the output matrix. A value closer to -1 means that the 'good' models have more of the *AND NOT* link operator in their respective boolean equations while a value closer to 1 means that the 'good' models have more of the *OR NOT* link operator.
layout	a (nx2) numeric matrix of x-y coordinates (2 columns) for each of the nodes (n) in the net igraph object. If NULL, we use the default layout provided by layout_nicely.
title	string. The title of the igraph plot

See Also

get_node_colors

Other network plotting functions: plot_avg_link_operator_diff_graphs(), plot_avg_state_diff_graph_vis(), plot_avg_state_diff_graphs(), plot_avg_state_diff_graph()

Examples

```
topology.file = system.file("extdata", "example.sif", package = "emba", mustWork = TRUE)
net = construct_network(topology.file)
diff = c(-0.95,-0.05,0.46,0.39,-0.04,0.72,-0.12,-0.51)
names(diff) = c("A","C","B","D","W","I","E","J")
plot_avg_link_operator_diff_graph(net, diff, title = "TEST")
```

---

plot_avg_link_operator_diff_graphs

Plot the graphs from an average link operator differences matrix

---

Description

This function presents a convenient way to use the plot_avg_link_operator_diff_graph function multiple times.

Usage

```
plot_avg_link_operator_diff_graphs(net, diff.mat, layout = NULL)
```

Arguments

net	igraph graph object
diff.mat	a matrix whose rows are **vectors of average node link operator differences** between two groups of models based on some kind of classification (e.g. number of TP predictions) and whose names are set in the rownames attribute of the matrix (usually denoting the different classification groups, e.g. (1,2) means the models that predicted 1 TP synergy vs the models that predicted 2 TP synergies, if the classification is done by number of TP predictions). The columns represent the network's node names. Could be the result of using the function get_avg_link_operator_diff_mat_based_on_tp_predictions.
layout	a (nx2) numeric matrix of x-y coordinates (2 columns) for each of the nodes (n) in the net igraph object. If NULL, we use the default layout provided by layout_nicely.

See Also

Other network plotting functions: plot_avg_link_operator_diff_graph(), plot_avg_state_diff_graph_vis(), plot_avg_state_diff_graphs(), plot_avg_state_diff_graph()

plot_avg_state_diff_graph *Plot the graph of average state differences (igraph)*

Description

This function uses the plot.igraph package to plot a network of nodes. The nodes are positioned according to the specified coordinates given by the layout parameter and the colors are derived using the diff values and the get_node_colors function. The color of each node indicates how much more inhibited or active that node is, when comparing the average model classified in the 'good' category vs the average 'bad' one.

Usage

```
plot_avg_state_diff_graph(net, diff, layout = NULL, title)
```

Arguments

net	igraph graph object
diff	numeric vector. Every value is in the [-1,1] interval and represents the average activity difference of each node. The node names have to be specified in the *names* attribute of the given vector. For example, diff could be the result of using the function get_avg_activity_diff_based_on_tp_predictions.
layout	a (nx2) numeric matrix of x-y coordinates (2 columns) for each of the nodes (n) in the net igraph object. If NULL, we use the default layout provided by layout_nicely.
title	string. The title of the igraph plot

See Also

get_node_colors

Other network plotting functions: plot_avg_link_operator_diff_graphs(), plot_avg_link_operator_diff_graph(), plot_avg_state_diff_graph_vis(), plot_avg_state_diff_graphs()

Examples

```
topology.file = system.file("extdata", "example.sif", package = "emba", mustWork = TRUE)
net = construct_network(topology.file)
diff = c(-0.95,-0.05,0.46,0.39,-0.04,0.72,-0.12,-0.51,-0.86,-0.80)
names(diff) = c("A","C","B","D","W","I","E","J","F","K")
plot_avg_state_diff_graph(net, diff, title = "TEST")
```

plot_avg_state_diff_graphs *Plot the graphs from an average state differences matrix*

Description

This function presents a convenient way to use the function plot_avg_state_diff_graph multiple times.

Usage

```
plot_avg_state_diff_graphs(net, diff.mat, layout = NULL)
```

Arguments

net	igraph graph object
diff.mat	a matrix whose rows are **vectors of average node activity state differences** between two groups of models based on some kind of classification (e.g. number of TP predictions) and whose names are set in the rownames attribute of the matrix (usually denoting the different classification groups, e.g. (1,2) means the models that predicted 1 TP synergy vs the models that predicted 2 TP synergies, if the classification is done by number of TP predictions). The columns represent the network's node names. Could be the result of using the function get_avg_activity_diff_mat_based_on_tp_predictions.
layout	a (nx2) numeric matrix of x-y coordinates (2 columns) for each of the nodes (n) in the net igraph object. If NULL, we use the default layout provided by layout_nicely.

See Also

Other network plotting functions: plot_avg_link_operator_diff_graphs(), plot_avg_link_operator_diff_graph(), plot_avg_state_diff_graph_vis(), plot_avg_state_diff_graph()

plot_avg_state_diff_graph_vis *Plot the graph of average state differences (visNetwork)*

Description

This function uses the visNetwork package to plot a network of nodes. The nodes are positioned by default in a hierarchical layout and their colors are derived using the diff values and the get_node_colors function. The color of each node indicates how much more inhibited or active that node is, when comparing the average model classified in the 'good' category vs the average 'bad' one.

Usage

```
plot_avg_state_diff_graph_vis(net, diff, nodes.size = 20, title)
```

Arguments

net	igraph graph object (to be translated to a visNetwork object)
diff	numeric vector. Every value must be in the [-1,1] interval and represents the average activity difference of each node. The node names have to be specified in the *names* attribute of the given vector. For example, diff could be the result of using the function get_avg_activity_diff_based_on_specific_synergy_prediction
nodes.size	an integer specifying the size of the nodes. Default value: 20.
title	string. The title of the visNetwork plot.

See Also

Other network plotting functions: plot_avg_link_operator_diff_graphs(), plot_avg_link_operator_diff_graph(), plot_avg_state_diff_graphs(), plot_avg_state_diff_graph()

Examples

```
topology.file = system.file("extdata", "example.sif", package = "emba", mustWork = TRUE)
net = construct_network(topology.file)
diff = c(-0.95,-0.05,0.46,0.39,-0.04,0.72,-0.12,-0.51,-0.86,-0.80)
names(diff) = c("A","C","B","D","W","I","E","J","F","K")
plot_avg_state_diff_graph_vis(net, diff, title = "TEST")
```

---

plot_mcc_classes_hist *Plot histogram of the MCC classes*

Description

This function is a wrapper of the ahist function for plotting nicely the distribution of the MCC models' values.

Usage

```
plot_mcc_classes_hist(models.mcc, models.cluster.ids, num.of.mcc.classes)
```

Arguments

models.mcc	a numeric vector of Matthews Correlation Coefficient (MCC) scores, one for each model. The *names* attribute may hold the models' names (but it is not required).

`models.cluster.ids`
: a numeric vector of cluster ids assigned to each model. It can be the result of using `Ckmeans.1d.dp` with input the models' MCC values (`models.mcc`) and the number of clusters (`num.of.mcc.classes`).

`num.of.mcc.classes`
: numeric. A positive integer (>2) that signifies the number of mcc classes (groups) that we should split the models MCC values.

Examples

```
models.mcc = c(-0.04, -0.17, 0.15, -0.24, -0.02 , 0.27, -0.42 , 0.38)
models.cluster.ids = c(2,2,3,1,2,3,1,3)
num.of.mcc.classes = 3
plot_mcc_classes_hist(models.mcc, models.cluster.ids, num.of.mcc.classes)
```

---

print_biomarkers_per_predicted_synergy

Print biomarkers for each predicted synergy

Description

Print biomarkers for each predicted synergy

Usage

```
print_biomarkers_per_predicted_synergy(
  biomarkers.dir,
  predicted.synergies,
  html.output = TRUE
)
```

Arguments

`biomarkers.dir`
: string. It specifies the full path name (without the ending character /) of the directory which holds the biomarker files for each drug combination in the `predicted.synergies`. The biomarker files must be formatted as: *%drug.comb%_biomarkers_active* or *%drug.comb%_biomarkers_inhibited*, where %drug.comb% is an element of the `predicted.synergies` vector. If the files are not properly formatted or don't even exist, zero biomarkers are reported.

`predicted.synergies`
: a character vector of the synergies (drug combination names) that were predicted by **at least one** of the models in the dataset.

`html.output`
: logical. If TRUE, it makes the printed output nice for an HTML document. Default value: TRUE.

print_model_and_drug_stats

Print model and drug statistics

Description

Use this function to pretty print in an R notebook useful statistics for the ensemble model analysis: how many drug combinations were tested by each model, the number of models used and how many nodes each boolean network model had.

Usage

```
print_model_and_drug_stats(drug.combs, models, nodes, html.output)
```

Arguments

drug.combs	integer. Number of drug combinations tested
models	integer. Number of models tested
nodes	integer. Number of network nodes
html.output	logical. If TRUE, the printed output will look nice in an HTML document

update_biomarker_files

Update biomarker files for a specific synergy

Description

This function gets the (previously-found or 'old') synergy biomarkers from their respective files and if any of these files are empty (no 'old' biomarkers found) or non-existent, the 'new' biomarkers (given as input vector parameters) are automatically saved. When the 'new' biomarkers **share common nodes** with the 'old' biomarkers, there exist 3 possible ways to combine the results, given by the method parameter. If **no common nodes** exist, no matter the method selected, the 'new' biomarkers are added to the 'old' ones.

Usage

```
update_biomarker_files(
  biomarkers.dir,
  drug.comb,
  biomarkers.active.new,
  biomarkers.inhibited.new,
  method = "replace"
)
```

Arguments

`biomarkers.dir` string. It specifies the full path name of the directory (without the ending character `/`) which holds the biomarker files for the synergistic drug combination specified in the parameter `drug.comb`. The biomarker files must be formatted as: *%drug.comb%_biomarkers_active* or *%drug.comb%_biomarkers_inhibited*, where %drug.comb% is the value of the `drug.comb` parameter.

`drug.comb` string. The drug combination (e.g. "A-B") that will be used to identify the related biomarker files.

`biomarkers.active.new`
a numeric vector whose *names* attribute includes the node names of the (newly found) *active biomarkers* for the specified synergy. The values of the vector are the average activity difference of each node, derived from a comparison between 2 different groups of models.

`biomarkers.inhibited.new`
a numeric vector whose *names* attribute includes the node names of the (newly found) *inhibited biomarkers* for the specified synergy. The values of the vector are the average activity difference of each node, derived from a comparison between 2 different groups of models.

`method` string. It specifies the method to use to update the biomarker files when there are *common nodes* between the 'old' and 'new' biomarkers:

1. `replace`(DEFAULT): we discard the 'old' biomarkers and keep only the 'new' ones
2. `prune.to.common`: we keep only the common biomarkers
3. `extend`: we add to the 'old' set of biomarkers the extra ones from the 'new' set that are not non-common to the 'old' ones, extending thus the 'old' biomarker set

---

`validate_observed_synergies_data` *Validate observed synergies data*

Description

This function checks that the observed synergies are part (a subset) of the tested drug combinations

Usage

```
validate_observed_synergies_data(observed.synergies, drug.combinations.tested)
```

Arguments

`observed.synergies`
a non-empty character vector of drug combinations

`drug.combinations.tested`
a non-empty character vector of drug combinations

Value

NULL if no errors found, otherwise stops execution.

Index