Package 'GeneAccord'

October 14, 2021

Type Package

Title Detection of clonally exclusive gene or pathway pairs in a cohort of cancer patients

Version 1.10.0

Author Ariane L. Moore, Jack Kuipers and Niko Beerenwinkel

Maintainer Ariane L. Moore <ariane.moore@bsse.ethz.ch>

Description A statistical framework to examine the combinations of clones that co-exist in tumors. More precisely, the algorithm finds pairs of genes that are mutated in the same tumor but in different clones, i.e. their subclonal mutation profiles are mutually exclusive. We refer to this as clonally exclusive. It means that the mutations occurred in different branches of the tumor phylogeny, indicating parallel evolution of the clones. Our statistical framework assesses whether a pattern of clonal exclusivity occurs more often than expected by chance alone across a cohort of patients. The required input data are the mutated gene-to-clone assignments from a cohort of cancer patients, which were obtained by running phylogenetic tree inference methods. Reconstructing the evolutionary history of a tumor and detecting the clones is challenging. For nondeterministic algorithms, repeated tree inference runs may lead to slightly different mutation-to-clone assignments. Therefore, our algorithm was designed to allow the input of multiple gene-to-clone assignments per patient. They may have been generated by repeatedly performing the tree inference, or by sampling from the posterior distribution of trees. The tree inference methods designate the mutations to individual clones. The mutations can then be mapped to genes or pathways. Hence our statistical framework can be applied on the gene level, or on the pathway level to detect clonally exclusive pairs of pathways. If a pair is significantly clonally exclusive, it points towards the fact that this specific clone configuration confers a selective advantage, possibly through synergies between the clones with these mutations.

Depends R (>= 3.5)

Suggests assertthat, BiocStyle, devtools, knitr, rmarkdown, testthat

Imports biomaRt, caTools, dplyr, ggplot2, graphics, grDevices, gtools, ggpubr, magrittr, maxLik, RColorBrewer, reshape2, stats, tibble, utils

biocViews BiomedicalInformatics, GeneticVariability, GenomicVariation, SomaticMutation, FunctionalGenomics, Genetics, MathematicalBiology, SystemsBiology, FeatureExtraction, PatternLogic, Pathways

License file LICENSE

Encoding UTF-8

LazyData true

RoxygenNote 6.0.1

VignetteBuilder knitr

URL https://github.com/cbg-ethz/GeneAccord

git_url https://git.bioconductor.org/packages/GeneAccord

git_branch RELEASE_3_13

git_last_commit cfc8768

git_last_commit_date 2021-05-19

Date/Publication 2021-10-14

R topics documented:

`all_genes_tbl` *Different gene id's and their chromosomal location.*

Description

This is a tibble that contains mappings between different gene identifiers. It can be created with the function `create_ensembl_gene_tbl_hg`. These are the human genes from the human genome version `hg19/GRCh37`, and Ensembl Genes version 88.

Usage

```
all_genes_tbl
```

Format

A tibble with 41'393 rows and seven variables:

ensembl_gene_id the Ensembl gene id as a character

hgnc_symbol the HGNC gene symbol as a character

entrezgene the Entrez gene id as an integer

uniprotswissprot the UniProtKB/Swiss-Prot gene id's as a character

chromosome_name the name of the chromosome where the gene is located as a character, e.g. "3" for chromosome three

start_position the nucleotide start position of the gene as an integer

end_position the nucleotide end position of the gene as an integer

Source

The tibble can be generated with `create_ensembl_gene_tbl_hg()`, which uses the R-package `biomaRt` and the Ensembl data base `www.ensembl.org`.

`avg_rates_m` *The average rates of clonal exclusivity of the example data set used in the vignette*

Description

This is a named vector that contains the average rate of clonal exclusivity for each of the 82 patients as described in the vignette.

Usage

```
avg_rates_m
```

Format

A vector with the average rates of clonal exclusivity of each patient. The names of each element is the respective patient name.

Source

The rates can be generated for each patient separately with `compute_rates_clon_excl`, and then taking the mean(). This is demonstrated in the vignette.

`build_null_test_statistic` *Simulate pairs to generate values of the test statistic under the null distribution*

Description

Generate samples from the test statistic under the null distribution - here we take the average rates of clonal exclusivity across trees, and also the histogram for each patient over all pairs with the values # clon. excl./#trees.

Usage

```
build_null_test_statistic(avg_rates_m, list_of_clon_excl_frac_trees_all_pats,
  num_pat_pair, num_pairs_sim, beta_distortion = 1000)
```

Arguments

avg_rates_m	The average rates of clonal exclusivity to be sampled from.
list_of_clon_excl_frac_trees_all_pats	The list of two lists. The first one contains a list entry for each patient containing the vector with the values of the information from each pair in a patient of how often it was mutated across trees. The second list entry is a list with an entry for each patient that is a vector with the values of in how many trees the pair was clonally exclusive. The patient ordering in the lists has to be the same as in avg_rates_m.
num_pat_pair	The number of patients the simulated pairs are mutated in.
num_pairs_sim	The number of simulated gene/pathway pairs to be generated.
beta_distortion	The value M=alpha + beta for the beta distribution, with which the average rates will be distorted. The bigger the M the higher the distribution is peaked around the actual rate. Therefore, the lesser the M, the more distorted the rates will be. Default: 1000.

Details

This function simulates gene pairs for the likelihood ratio test to generate values from the test statistic under the null. It draws the average rates of clonal exclusivity from the ones provided by the user. That is, the average rates of clonal exclusivity have to be computed first for each patient. The number of patients the simulated pairs are mutated in can be specified with num_pat_pair. This function can be used to build the ecdf of the test statistic under the null hypothesis (see Examples). The patients in which the simulated pairs are mutated in are randomly selected proportional to the number of pairs in a patient.

Value

The return value is a tibble with the columns 'test_statistic', 'mle_delta', and num_pat_pair columns with the respective rates that were drawn for each of the patients, num_pat_pair columns with the respective number of mutated times across trees, and num_pat_pair columns with the respective number of times of being clonally exclusive across trees, and num_pat_pair columns with the rate that was distorted by the beta distribution. The 'test_statistic' is the test statistic of the likelihood ratio test. The 'mle_delta' is the maximum likelihood estimate of the delta for the elevated clonal exclusivity rate in the alternative model of the likelihood ratio test.

Author(s)

Ariane L. Moore

Examples

```
avg_rates_m=c(0.4, 0.3)
list_of_clon_excl_frac_trees_all_pats <- list(list(c(5, 4, 5), c(5, 4)),
                                    list(c(4, 4, 3), c(3, 2)))
sim_pairs <- build_null_test_statistic(avg_rates_m,
             list_of_clon_excl_frac_trees_all_pats, 2, 100,
```

```
             beta_distortion=100)
ecdf_test_stat <-
  ecdf(as.numeric(as.character(sim_pairs$test_statistic)))
plot(ecdf_test_stat,
 main="ECDF of the test statistic when num_pat_pair=2")
# assume the observed test statistic t=6.0,
# compute a p-value given the ecdf of
# the test statistic ecdf(T) from the null distribution
# p_value=P(T>t | H_0 true)=1-ecdf(t) ## (upper-tailed test)
p_value <- 1-ecdf_test_stat(6.0)
```

`clone_tbl_all_pats_all_trees` *The tibble with gene-to-clone assignments from all patients and all trees*

Description

This is a tibble that contains the information, which gene is mutated in which clone from which patient.

Usage

```
clone_tbl_all_pats_all_trees
```

Format

A tibble containing the following columns:

file_name the name of the csv-file from which the data was read

patient_id the patient identifier

altered_entity ensembl gene identifier of the mutated gene

clone1 the indication whether the current gene is mutated in this clone

clone2 the indication whether the current gene is mutated in this clone

clone3 the indication whether the current gene is mutated in this clone

clone4 the indication whether the current gene is mutated in this clone

clone5 the indication whether the current gene is mutated in this clone

clone6 the indication whether the current gene is mutated in this clone

clone7 the indication whether the current gene is mutated in this clone

tree_id the identifier that tells from which tree this gene-to-clone assignment comes

Source

The tibble can be generated for each patient separately with `create_tbl_tree_collection` as demonstrated in the vignette.

compute_rates_clon_excl *Get rates of clonal exclusivity for each tree inference*

Description

Compute the clonal exclusivity rates for each gene-to-clone-assignment from the collection of tree inferences.

Usage

```
compute_rates_clon_excl(pat_tbl)
```

Arguments

pat_tbl	A tibble with the information of which gene/pathway is altered in which clone in the patient, and including this information from the collection of trees. Can be created with with create_tbl_tree_collection.

Details

Takes the gene-to-clone assignment tibble as created with create_tbl_tree_collection and computes for each instance from the collection of trees the rate of clonal exclusivity. This rate is the fraction of gene/pathway pairs that were on a different branch in the tumor phylogeny, i.e. the fraction of pairs that was clonally exclusive.

Value

A vector with all rates of clonal exclusivity from all tree inferences.

Author(s)

Ariane L. Moore

Examples

```
clone_tbl <- dplyr::tibble(file_name =
    rep("fn1", 10),
    "patient_id"=rep("pat1", 10),
    "altered_entity"=paste0("gene",
    LETTERS[seq_len(10)]),
    "clone1"=c(0, 1, 0, 1, 0, 1, 0, 1, 1, 1),
    "clone2"=c(1, 0, 1, 0, 1, 1, 1, 0, 0, 1),
    "tree_id"=c(rep(1, 5), rep(2, 5)))
compute_rates_clon_excl(clone_tbl)
```

`compute_test_stat_avg_rate` *Compute the test statistic of the clonal exclusivity test (lrtest).*

Description

Compute test statistic that is based on the average rates of clonal exclusivity of a patient, and the observed number of times a pair was clonally exclusive across several trees of the tree inference.

Usage

```
compute_test_stat_avg_rate(avg_rates_m, num_trees_pair, num_clon_excl)
```

Arguments

`avg_rates_m`	The vector of average rates of clonal exclusivity of each patient the pair is mutated in. It was computed for each patient separately, and is averaged over all gene pairs and all trees. Expected to be in the same (patient) order as the other inputs to this function.
`num_trees_pair`	The vector with the number of tree inferences in which the pair was occurring in. Has to be the same order as `avg_rates_m`.
`num_clon_excl`	The vector with the number of times the pair was clonally exclusive in the trees in each patient. Has to be the same order as `avg_rates_m`.

Details

For a given gene/pathway pair, this function takes as input: the average rates of clonal exclusivity of all patients in which the pair is mutated, the number of trees among the trees in the collection of trees from each patient in which the pair was occurring, and the number of times it was clonally exclusive

Value

A list with the test statistic of the clonal exclusivity test (lrtest), and the maximum likelihood estimate of delta.

Author(s)

Ariane L. Moore

Examples

```
compute_test_stat_avg_rate(c(0.1, 0.2), c(10, 10), c(9, 7))
compute_test_stat_avg_rate(c(0.05, 0.23), c(20, 20), c(8, 5))
```

`convert_ensembl_to_reactome_pw_tbl` *Map ensembl gene id clone tibble to reactome pathway clone tibble.*

Description

For a tibble that contains the information which ensembl gene id is mutated in which clone, map the ensembl gene id to the reactome pathways that contain this gene.

Usage

```
convert_ensembl_to_reactome_pw_tbl(mutated_gene_tbl, ensg_reactome_path_map)
```

Arguments

`mutated_gene_tbl`
: The tibble containing the information of which ensembl gene id is altered in which patient and clone. Can be created with e.g. `create_tbl_ent_clones`.

`ensg_reactome_path_map`
: A tibble with all ensembl id's and their reactome pathways. Can be loaded with `data("ensg_reactome_path_map")`.

Details

Such a tibble can be generated with e.g. the function `create_tbl_tree_collection`. If the altered entities in the lists were the ensembl gene id's, this function can convert the tibble into a tibble with the altered reactome pathways. It has the columns 'file_name', 'patient_id', 'altered_entity', 'clone1', 'clone2', ... up to the maximal number of clones (Default: until 'clone7'). If the mutated entities are ensembl gene id's, they can be mapped with this function to the pathways from 'reactome'. The pathways are from the lowest level of hierarchy.

Value

The tibble containing the information of which pathway is altered in which clone.

Author(s)

Ariane L. Moore

Examples

```
data("ensg_reactome_path_map")
mutated_gene_tbl <-
  dplyr::tibble(file_name=c("pat1.csv", "pat1.csv"),
patient_id=c("1","1"),
altered_entity=c("ENSG00000134086",
"ENSG00000141510"),
clone1=c(1,0),
```

```
clone2=c(0,1))
convert_ensembl_to_reactome_pw_tbl(mutated_gene_tbl,
    ensg_reactome_path_map)
```

`create_ensembl_gene_tbl_hg` *Get a tibble of all gene ensembl id's, gene names (hgnc), entrez gene id's, uniprot/swissprot gene id's and genomic coordinates.*

Description

Retrieve a mapping between different gene identifiers.

Usage

```
create_ensembl_gene_tbl_hg(GRCh = 37, ensembl_version = 88)
```

Arguments

`GRCh`	The human genome version. Default: 37.
`ensembl_version`	The version of the ensembl data base. Default: 88.

Details

This function retrieves the ensembl gene id's from biomart together with the hgnc gene symbol, the entrez gene id, the uniprot/swissprot gene id, as well as chromosome, start and end position. This is done for the human genes from the human genome version `hg19/GRCh37`, and Ensembl Genes version 88. The user can also specify other human genome or ensembl versions.

Value

A tibble with the following columns: `ensembl_gene_id`, `hgnc_symbol`, `entrezgene`, `uniprotswissprot`, `chromosome_name`, `start_position`, `end_position`. The entrez gene id, as well as the start and end positions are numeric, and the other columns are characters. The chromosome is specified without "chr", i.e. the chromosome 13 for example, would be specified with "13".

Author(s)

Ariane L. Moore

Examples

```
## Not run:
create_ensembl_gene_tbl_hg()

## End(Not run)
```

`create_tbl_ent_clones` *Get clone alteration tibble.*

Description

Creates a tibble containing the information of which genes/pathways are altered in a patient in which clone.

Usage

```
create_tbl_ent_clones(path_to_file, max_num_clones = 7)
```

Arguments

`path_to_file`	The path to the file with the table of altered genes/pathways and their clone affiliation.
`max_num_clones`	The upper bound for the number of clones that were found per tumor. Default: 7.

Details

It expects a comma-separated table where the first column is the name of the altered gene or pathway. The other columns are for the clones in the respective tumor. Such a table can be generated with a tool that identifies clones in tumor samples, e.g. `Cloe`.

The table is expected to be comma-separated and to have the columns 'altered_entity', 'clone1', 'clone2', ..., 'cloneN', depending on how many clones were detected in the respective tumor. Each row then contains in the first column the name of the mutated gene or affected pathway, e.g. "ENSG00000134086", and in the other columns it has either zeros or ones, indicating in which clone the respective gene/pathway is altered.

Value

The tibble containing the information of which gene/pathway is altered in which clone in a patient. Has the columns 'file_name', 'patient_id', 'altered_entity', 'clone1', 'clone2', ... up to the maximal number of clones (Default: until 'clone7'). Note that the labelling of the clones does not matter and only needs to stay fixed within each patient and tree inference.

Author(s)

Ariane L. Moore

Examples

```
ext_data_dir <- system.file('extdata', package='GeneAccord')
create_tbl_ent_clones(paste(ext_data_dir,
    "/clonal_genotypes/cloe_seed5/01.csv", sep=""))
```

`create_tbl_tree_collection` *Get clone alteration tibble across the collection of trees.*

Description

Read in the patient's gene-to-clone assignment across a collection of trees

Usage

```
create_tbl_tree_collection(input_files, no_noisy_ents = 0.9,
  max_num_clones = 7)
```

Arguments

`input_files`	A vector containing the paths to the files with the tables of altered genes/pathways and their clone affiliation from the collection of tree inferences.
`no_noisy_ents`	Minimum fraction for genes/pathways of how often they have to occur across the collection of trees in order to be in the tibble. This makes sure that noisy genes, which were not assigned to many trees are excluded. Default: 0.9.
`max_num_clones`	The upper bound for the number of clones that were found per tumor. Default: 7.

Details

Creates a tibble containing the information of which genes/pathways are altered in which clone in a patient across a collection of tree inferences. It expects a list containing the paths to the comma-separated tables where the first column is the name of the altered gene or pathway. The other columns are for the clones in the respective tumor. Such tables can be generated by repeatedly performing the phylogenetic tree inference with e.g. the package `Cloe`, or by sampling from the posterior. The tables are expected to be comma-separated and to have the columns 'altered_entity', 'clone1', 'clone2', ..., 'cloneN', depending on how many clones were detected in the respective tumor. Each row then contains in the first column the name of the mutated gene or affected pathway, e.g. "ENSG00000134086", and in the other columns it has either zeros or ones, indicating in which clone the respective gene/pathway is altered.

Value

A clean tibble with the information of which gene/pathway is altered in which clone in the patient, and with an entry for each tree inference where it occurred. Has the columns 'file_name', 'patient_id', 'altered_entity', 'clone1', 'clone2', ... up to the maximal number of clones (Default: until 'clone7'), and 'tree_id' as an indication in which tree the assignment was found. Note that the labelling of the clones does not matter and only needs to stay fixed within each patient and tree inference.

Author(s)

Ariane L. Moore

Examples

```
ext_data_dir <- system.file('extdata', package='GeneAccord')
this_patient <- "01"
input_files_01 <- paste(ext_data_dir,
    "/clonal_genotypes/cloe_seed", seq(5, 100, by=5),
    "/", this_patient, ".csv", sep="")
create_tbl_tree_collection(input_files_01)
```

---

ecdf_list *The list with the ECDF's of the test statistic under the null hypothesis*

Description

This is a list whose entries are the empirical cumulative distribution functions for different number of patients that pairs can be mutated in.

Usage

```
ecdf_list
```

Format

A list whose entries are the empirical cumulative distribution functions. Entry 1 is set to NULL, because GeneAccord does not test pairs that occur in just one patient. Entry 2 then contains the ECDF of the test statistic under the null hypothesis for the case that pairs are mutated in two patients. Entry 3 contains the ECDF for the case where pairs occur in three patients.

Source

The list was generated with the function `generate_ecdf_test_stat` as demonstrated in the vignette, just that the following parameter was set as `num_pairs_sim = 100000`.

---

ecdf_lr_test_clon_excl_avg_rate *Compare observed likelihood ratio test statistic to its ecdf under null.*

Description

Compare the likelihood ratio test statistic to its ecdf under the null for two mutated genes/pathways in clones of patients.

Usage

```
ecdf_lr_test_clon_excl_avg_rate(entA, entB, clone_tbl, avg_rates_m, ecdf_list,
  alternative)
```

Arguments

entA	One gene/pathway of the pair.
entB	The other gene/pathway of the pair.
clone_tbl	The clone tibble as generated with create_tbl_tree_collection from several trees of the tree inference, i.e. it also contains a column 'tree_id'.
avg_rates_m	The average rates of clonal exclusivity for each patient. The name of each rate is the respective patient_id.
ecdf_list	The list of ECDF's of the test statistic under the null distribution. Can be generated with generate_ecdf_test_stat. It is important that the rates that are used for that are the same as the avg_rates_m here.
alternative	The character indicating whether pairs should only be tested if delta > 0 or if all pairs should be tested. Can be one of "greater" or "two.sided".

Details

Tests whether the observed number of clonal exclusivities of mutated entities (genes or pathways) A and B in clones of patients is significantly different from what would be expected given the average clonal exclusivity rates. The observed test statistic is compared to the ecdf of the test statistic under the null hypothesis.

Value

Returns list(p_val, num_patients, mle_delta, test_statistic), i.e. a list with the p-value, the number of patients in which both of the genes/pathways were mutated, the maximum likelihood estimate of the delta, and the test statistic.

Author(s)

Ariane L. Moore

Examples

```
clone_tbl <- dplyr::tibble("file_name"=
           rep(c(rep(c("fn1", "fn2"), each=3)), 2),
          "patient_id"=rep(c(rep(c("pat1", "pat2"), each=3)), 2),
          "altered_entity"=c(rep(c("geneA", "geneB", "geneC"), 4)),
          "clone1"=c(0, 1, 0, 1, 0, 1, 0, 1, 1, 1, 0, 0),
          "clone2"=c(1, 0, 1, 0, 1, 1, 1, 0, 0, 1, 0, 1),
          "tree_id"=c(rep(5, 6), rep(10, 6)))
clone_tbl_pat1 <- dplyr::filter(clone_tbl, patient_id == "pat1")
clone_tbl_pat2 <- dplyr::filter(clone_tbl, patient_id == "pat2")
rates_exmpl_1 <- compute_rates_clon_excl(clone_tbl_pat1)
rates_exmpl_2 <- compute_rates_clon_excl(clone_tbl_pat2)
avg_rates_m <- apply(cbind(rates_exmpl_1, rates_exmpl_2), 2, mean)
names(avg_rates_m) <- c(names(rates_exmpl_1)[1],
    names(rates_exmpl_2)[1])
values_clon_excl_num_trees_pat1 <- get_hist_clon_excl(clone_tbl_pat1)
values_clon_excl_num_trees_pat2 <- get_hist_clon_excl(clone_tbl_pat2)
list_of_num_trees_all_pats <-
```

```
 list(pat1=values_clon_excl_num_trees_pat1[[1]],
      pat2=values_clon_excl_num_trees_pat2[[1]])
list_of_clon_excl_all_pats <-
  list(pat1=values_clon_excl_num_trees_pat1[[2]],
       pat2=values_clon_excl_num_trees_pat2[[2]])
num_pat_pair_max <- 2
num_pairs_sim <- 10
ecdf_list <- generate_ecdf_test_stat(avg_rates_m,
                                list_of_num_trees_all_pats,
                                list_of_clon_excl_all_pats,
                                num_pat_pair_max,
                                num_pairs_sim)
ecdf_lr_test_clon_excl_avg_rate("geneA", "geneB", clone_tbl,
                                   avg_rates_m,
                                   ecdf_list, "greater")
```

`ensembl_to_hgnc` *Get the hgnc gene symbol for an ensembl gene id.*

Description

Map a given ensembl gene id to the hgnc gene symbol.

Usage

```
ensembl_to_hgnc(this_ensembl, all_genes_tbl)
```

Arguments

`this_ensembl`	The ensembl id of a gene.
`all_genes_tbl`	A tibble with all genes ensembl id's and hgnc symbols.

Details

For an ensembl id and a tibble with all genes as input, this function returns the matching hgnc gene symbol. The tibble with all genes can be generated with `create_ensembl_gene_tbl_hg`.

Value

The matching hgnc gene symbol.

Author(s)

Ariane L. Moore

Examples

```
## Not run:
all_genes_tbl <- create_ensembl_gene_tbl_hg()
ensembl_to_hgnc("ENSG00000134086", all_genes_tbl)
ensembl_to_hgnc("ENSG00000141510", all_genes_tbl)

## End(Not run)
```

`ensembl_to_reactome` *Get the reactome pathways for an ensembl gene id.*

Description

Map a given ensembl gene id to the reactome pathways that contain this gene.

Usage

```
ensembl_to_reactome(this_ensembl, ensg_reactome_path_map)
```

Arguments

`this_ensembl`	The ensembl id of a gene.
`ensg_reactome_path_map`	A tibble with all ensembl id's and their reactome pathways. Can be loaded with `data("ensg_reactome_path_map")`.

Details

As input, an ensembl gene id is given as well as the tibble 'ensg_reactome_path_map'. It can be loaded with `data("ensg_reactome_path_map")`, and contains the ensembl gene id to reactome pathway mappings. The reactome pathways are from the lowest level of the hierarchy. This function returns the reactome pathways for the input gene.

Value

The pathways that contain this gene as a character vector.

Author(s)

Ariane L. Moore

Examples

```
data("ensg_reactome_path_map")
ensg_gene <- "ENSG00000134086"
ensembl_to_reactome(ensg_gene, ensg_reactome_path_map)
```

ensg_reactome_path_map *Ensembl gene id's and the Reactome pathways.*

Description

This is a tibble that contains mappings between ensembl gene id's and reactome pathways. The reactome pathways are from the lowest level in the hierarchy ("Lowest level pathway diagram / Subset of the pathway"), and were obtained by download from the Reactome website (https://reactome.org/download-data; "ENSEMBL to pathways"). The following commands were used: wget https://reactome.org/download/current/Ensembl2Reactome.txt; cat Ensembl2Reactome.txt | grep "Homo sapiens" > Ensembl2Reactome_homo_sapiens.txt

Usage

```
ensg_reactome_path_map
```

Format

A tibble with 46'141 rows and six variables:

ensembl_gene_id the Ensembl gene id as a character

reactome_pw_id the Reactome pathway stable identifier

url The url leading to the pathway graph

reactome_pw_name the name of the Reactome pathway

evidence_code the evidence code

species the species

Source

The tibble was created as follows: library(dplyr); ensg_path_map_raw <- read.csv("Ensembl2Reactome_homo_sapiens.txt", header = F, sep = "\t", comment.char = "", check.names = F, skip = 0); stopifnot(dim(ensg_path_map_raw)[1] == 46141); stopifnot(dim(ensg_path_map_raw)[2] == 6); colnames(ensg_path_map_raw) <- c("ensembl_gene_id", "reactome_pw_id", "url", "reactome_pw_name", "evidence_code", "species"); ensg_path_map_raw <- dplyr::as.tbl(ensg_path_map_raw); ensg_reactome_path_map <- filter(filter(ensg_path_map_raw, grepl("ENSG", ensg_path_map_raw$ensembl_gene_id)), species == "Homo sapiens")

`ensmusg_reactome_path_map` *Ensembl gene id's and the Reactome pathways - for mouse!*

Description

This is a tibble that contains mappings between mouse ensembl gene id's and reactome pathways. The reactome pathways are from the lowest level in the hierarchy ("Lowest level pathway diagram / Subset of the pathway"), and were obtained by download from the Reactome website (https://reactome.org/download-data; "ENSEMBL to pathways"). The following commands were used: wget https://reactome.org/download/current/Ensembl2Reactome.txt; cat Ensembl2Reactome.txt | grep "Mus musculus" > Ensembl2Reactome_mus_musculus.txt # and then Ensembl2Reactome_mus_musculus_woOmegaSy was created from this by just # replacing the greek 'omega'-symbol in pathway "R-MMU-9027604" with the word 'omega'.

Usage

```
ensmusg_reactome_path_map
```

Format

A tibble with 28,630 rows and six variables:

ensembl_gene_id the Ensembl gene id as a character

reactome_pw_id the Reactome pathway stable identifier

url The url leading to the pathway graph

reactome_pw_name the name of the Reactome pathway

evidence_code the evidence code

species the species

Source

The tibble was created as follows: library(dplyr); ensmusg_path_map_raw <- read.csv("Ensembl2Reactome_mus_musculus_ header = F, sep = "\t", comment.char = "", check.names = F, skip = 0); stopifnot(dim(ensmusg_path_map_raw)[1] == 28696); stopifnot(dim(ensmusg_path_map_raw)[2] == 6); colnames(ensmusg_path_map_raw) <- c("ensembl_gene_id", "reactome_pw_id", "url", "reactome_pw_name", "evidence_code", "species"); ensmusg_path_map_raw <- dplyr::as.tbl(ensmusg_path_map_raw); ensmusg_reactome_path_map <- filter(filter(ensmusg_path_map_raw, grepl("ENSMUSG", ensmusg_path_map_raw$ensembl_gene_id)), species == "Mus musculus") stopifnot(dim(ensmusg_reactome_path_map)[1] == 28630)

`extract_num_clones_tbl` *Extract number of clones.*

Description

Extract number of clones in each patient.

Usage

```
extract_num_clones_tbl(clone_tbl)
```

Arguments

`clone_tbl`	The tibble generated with `create_tbl_ent_clones`.

Details

Given a clone tibble as created with `create_tbl_ent_clones` this function extracts the information, how many clones there are in each patient. The counted clones will be those with at least one non-zero entry, i.e. at least one gene/pathway assigned to the clone.

Value

A named vector with the number of clones in each patient. The name of each element is the respective patient_id.

Author(s)

Ariane L. Moore

Examples

```
clone_tbl <- dplyr::tibble(
    file_name=c(rep("fn1", 2), rep("fn2", 2)),
    patient_id=c(rep("pat1", 2), rep("pat2", 2)),
    altered_entity=c("pw1", "pw2", "pw1", "pw3"),
    clone1=c(0, 0, 0, 0),
    clone2=c(0, 1, 0, 1),
    clone3=c(1, 1, 0, 1),
    clone4=c(1, 0, 0, 0))
extract_num_clones_tbl(clone_tbl)
```

GeneAccord *Detection of clonally exclusive gene or pathway pairs in a cohort of cancer patients*

Description

Method to detect clonally exclusive gene or pathway pairs in a cohort of cancer patients

Usage

```
GeneAccord(clone_tbl, avg_rates_m, ecdf_list, alternative = "greater",
  genes_of_interest = "ALL", AND_OR = "OR")
```

Arguments

clone_tbl	The tibble containing the information of which gene/pathway is mutated in which clone from which patient and in which tree from the collection of trees. Can be generated with create_tbl_tree_collection for each patient separately and then appended.
avg_rates_m	The average rates of clonal exclusivity for each patient as computed with compute_rates_clon_excl. The name of each rate is the respective patient id. The rates are assumed to be the average over all tree inferences from a patient.
ecdf_list	The list of ECDF's of the test statistic under the null distribution. Can be generated with generate_ecdf_test_stat.
alternative	The character indicating whether pairs should only be tested if delta > 0 or if all pairs should be tested. Can be one of "greater" or "two.sided". Default: "greater".
genes_of_interest	A character vector of genes to test for clonal exclusivity. The genes have to be in the same identifier as the one in the tibble. Per default, all genes are tested. Default: "ALL".
AND_OR	If genes_of_interest is specified, this indicator tells whether to test only pairs within the genes_of_interest (AND), or whether all pairs involving at least one of these genes should be tested (OR). I.e. can be one of "AND", "OR". Default: "OR". If genes_of_interest is "ALL", then all gene pairs will be tested and this parameter is ignored.

Details

After running a tool such as Cloe that identifies clones in a tumor and infers the phylogenetic history, the user has for each tumor a list of alterations and their clone assignments. Since the tree inference includes uncertainty, it may be run several times. Given a tibble containing the information of which genes/pathways are mutated in which patient and clone and from which tree, this function systematically tests the data for significant clonal exclusivities. That is, it checks for each gene/pathway pair whether the number of clonal exclusivities is significantly different from what would be expected by chance. Such a tibble can be generated with create_tbl_tree_collection, and then adding

the additional column 'tree_id' to indicate which tree of the tree inference was used. For instance, if the tree inference tool was run several times using different seeds, the column 'tree_id' may contain the seed of the respective tree. Hence, the tibble is expected to have the columns 'file_name', 'patient_id', 'altered_entity', 'clone1', 'clone2', ... up to the maximal number of clones (Default: until 'clone7'), and 'tree_id'. Note that the labelling of the clones does not matter and only needs to stay fixed within each patient and tree inference. There is also the option to test two-sided, meaning that also pairs will be tested that tend to occur more often together in the same clones or separate in different clones. Hence it also allows to detect significant clonal co-occurrence. An additional option is to test only a specific subset of genes.

Value

A tibble containing the test result for each pair of mutated genes/pathways that was tested. More precisely, it contains the columns 'entity_A', 'entity_B', 'num_patients', 'pval', 'mle_delta', 'test_statistic', and 'qval'. Each row is then a gene or pathway pair which is specified with 'entity_A', and 'entity_B'. Note that the test is symmetric, hence switching the labels A and B does not change the results. The column 'num_patients' contains the information in how many patients both of the genes/pathways were mutated and hence how many patients' rates were used for the test. The 'pval' is the p-value of the clonal exclusivity test. The 'mlde_delta' is the maximum likelihood estimate of the delta for the elevated clonal exclusivity rate in the alternative model. The column 'test_statistic' is the likelihood ratio test statistic. The 'qval' is the adjusted p-value after multiple testing correction with Benjamini-Hochberg.

Author(s)

Ariane L. Moore, <ariane.moore@bsse.ethz.ch>

Examples

```
clone_tbl <- dplyr::tibble("file_name"=
   rep(c(rep(c("fn1", "fn2"), each=3)), 2),
   "patient_id"=rep(c(rep(c("pat1", "pat2"), each=3)), 2),
   "altered_entity"=c(rep(c("geneA", "geneB", "geneC"), 4)),
   "clone1"=c(0, 1, 0, 1, 0, 1, 0, 1, 1, 1, 0, 0),
   "clone2"=c(1, 0, 1, 0, 1, 1, 1, 0, 0, 1, 0, 1),
   "tree_id"=c(rep(5, 6), rep(10, 6)))
clone_tbl_pat1 <- dplyr::filter(clone_tbl, patient_id == "pat1")
clone_tbl_pat2 <- dplyr::filter(clone_tbl, patient_id == "pat2")
rates_exmpl_1 <- compute_rates_clon_excl(clone_tbl_pat1)
rates_exmpl_2 <- compute_rates_clon_excl(clone_tbl_pat2)
avg_rates_m <- apply(cbind(rates_exmpl_1, rates_exmpl_2), 2, mean)
names(avg_rates_m) <- c(names(rates_exmpl_1)[1],
names(rates_exmpl_2)[1])
values_clon_excl_num_trees_pat1 <- get_hist_clon_excl(clone_tbl_pat1)
values_clon_excl_num_trees_pat2 <- get_hist_clon_excl(clone_tbl_pat2)
list_of_num_trees_all_pats <-
    list(pat1=values_clon_excl_num_trees_pat1[[1]],
    pat2=values_clon_excl_num_trees_pat2[[1]])
list_of_clon_excl_all_pats <-
    list(pat1=values_clon_excl_num_trees_pat1[[2]],
    pat2=values_clon_excl_num_trees_pat2[[2]])
```

```
num_pat_pair_max <- 2
num_pairs_sim <- 10
ecdf_list <- generate_ecdf_test_stat(avg_rates_m,
  list_of_num_trees_all_pats,
  list_of_clon_excl_all_pats,
  num_pat_pair_max,
  num_pairs_sim)
alternative <- "greater"
GeneAccord(clone_tbl, avg_rates_m, ecdf_list, alternative)
alternative <- "two.sided"
GeneAccord(clone_tbl, avg_rates_m, ecdf_list, alternative)
genes_of_interest <- c("geneB", "geneC")
GeneAccord(clone_tbl, avg_rates_m, ecdf_list,
            alternative, genes_of_interest)
AND_OR <- "AND"
GeneAccord(clone_tbl, avg_rates_m, ecdf_list,
            alternative, genes_of_interest, AND_OR)
```

`generate_ecdf_test_stat` *Generate the ECDF of the test statistic under the null distribution - taking the average rates of clonal exclusivity*

Description

Generate the ECDF of the test statistic under the null distribution - taking the average rates of clonal exclusivity, as well as sampling from the real data for each patient, in how many trees a pair occurs and is clonally excl.

Usage

```
generate_ecdf_test_stat(avg_rates_m, list_of_num_trees_all_pats,
  list_of_clon_excl_all_pats, num_pat_pair_max, num_pairs_sim,
  beta_distortion = 1000)
```

Arguments

`avg_rates_m`	The average rates of clonal exclusivity from all the patients in the cohort, and averaged over several trees from the collection of tree inferences.
`list_of_num_trees_all_pats`	A named list that contains an entry for each patient which is the vector with the values of the information from each pair in a patient of how often it was mutated across trees. The patient odering in the list has to be the same as in `avg_rates_m`.
`list_of_clon_excl_all_pats`	A named list with an entry for each patient that is a vector with the values of in how many trees a pair was clonally exclusive. The patient ordering in the list has to be the same as in `avg_rates_m`.

num_pat_pair_max
: The maximum number of patients a pair is mutated in.

num_pairs_sim
: The number of simulated gene/pathway pairs to be generated, i.e. the number of times the test statistic is computed. Recommended to choose a big number, e.g. 100000.

beta_distortion
: The value `M=alpha + beta` for the beta distribution, with which the average rates will be distorted. The bigger the M the higher the distribution is peaked around the actual rate. Therefore, the lesser the M, the more distorted the rates will be. Default: 1000.

Details

This function takes the computed average rates of clonal exclusivity from the data (m1, ... mN), which are specific to each patient and averaged over several trees from the collection of tree inferences. It also takes the histogram for each patient, of the values of how often a pair was clonally exclusive over the number of trees it was mutated in. It then simulates the test statistic under the null for each number of patients a pair is be mutated in from 2, 3, ... 'num_pat_pair_max'. Afterwards, it generates the empirical cumulative distribution function (ECDF) using the `ecdf` function of the stats package and returns the list with the ECDF's for the number of patients n=2, 3, ..., N. This step is necessary for each new data set before the clonal exclusivity test can be done. In the clonal exclusivity test, the observed test statistics are compared to the ECDF.

Value

The return value is a list with ECDF's. The first list entry is just set to NULL for technical reasons.

Author(s)

Ariane L. Moore

Examples

```
clone_tbl <- dplyr::tibble("file_name" =
   rep(c(rep(c("fn1", "fn2"), each=3)), 2),
   "patient_id"=rep(c(rep(c("pat1", "pat2"), each=3)), 2),
   "altered_entity"=c(rep(c("geneA", "geneB", "geneC"), 4)),
   "clone1"=c(0, 1, 0, 1, 0, 1, 0, 1, 1, 1, 0, 0),
   "clone2"=c(1, 0, 1, 0, 1, 1, 1, 0, 0, 1, 0, 1),
   "tree_id"=c(rep(5, 6), rep(10, 6)))
clone_tbl_pat1 <- dplyr::filter(clone_tbl, patient_id == "pat1")
clone_tbl_pat2 <- dplyr::filter(clone_tbl, patient_id == "pat2")
rates_exmpl_1 <- compute_rates_clon_excl(clone_tbl_pat1)
rates_exmpl_2 <- compute_rates_clon_excl(clone_tbl_pat2)
avg_rates_m <- apply(cbind(rates_exmpl_1, rates_exmpl_2), 2, mean)
names(avg_rates_m) <- c(names(rates_exmpl_1)[1], names(rates_exmpl_2)[1])
values_clon_excl_num_trees_pat1 <- get_hist_clon_excl(clone_tbl_pat1)
values_clon_excl_num_trees_pat2 <- get_hist_clon_excl(clone_tbl_pat2)
list_of_num_trees_all_pats <-
    list(pat1=values_clon_excl_num_trees_pat1[[1]],
    pat2=values_clon_excl_num_trees_pat2[[1]])
```

```
list_of_clon_excl_all_pats <-
    list(pat1=values_clon_excl_num_trees_pat1[[2]],
    pat2=values_clon_excl_num_trees_pat2[[2]])
num_pat_pair_max <- 2
num_pairs_sim <- 10
ecdf_list <- generate_ecdf_test_stat(avg_rates_m,
                list_of_num_trees_all_pats, list_of_clon_excl_all_pats,
                num_pat_pair_max, num_pairs_sim)
plot(ecdf_list[[2]])
```

`generate_test_stat_hist` *Generate the test statistic and p-values under the null distribution*

Description

Generate the values of the test statistic under the null, and also p-values of the clonal exclusivity test under the null. Taking the average rates of clonal exclusivity, as well as sampling from the real data for each patient, in how many trees a pair occurs and is clonally exclusive.

Usage

```
generate_test_stat_hist(avg_rates_m, list_of_num_trees_all_pats,
  list_of_clon_excl_all_pats, ecdf_list, num_pat_pair_max, num_pairs_sim,
  beta_distortion = 1000)
```

Arguments

`avg_rates_m`	The average rates of clonal exclusivity from all the patients in the cohort, and averaged over several trees from the collection of tree inferences.
`list_of_num_trees_all_pats`	A named list that contains an entry for each patient which is the vector with the values of the information from each pair in a patient of how often it was mutated across trees. The patient odering in the list has to be the same as in `avg_rates_m`.
`list_of_clon_excl_all_pats`	A named list with an entry for each patient that is a vector with the values of in how many trees a pair was clonally exclusive. The patient ordering in the list has to be the same as in `avg_rates_m`.
`ecdf_list`	The list with ECDF's as generated with `generate_ecdf_test_stat`.
`num_pat_pair_max`	The maximum number of patients a pair is mutated in.
`num_pairs_sim`	The number of simulated gene/pathway pairs to be generated, i.e. the number of times the test statistic is computed.

`beta_distortion`
: The value `M=alpha + beta` for the beta distribution, with which the average rates will be distorted. The bigger the M the higher the distribution is peaked around the actual rate. Therefore, the lesser the M, the more distorted the rates will be. Default: 1000.

Details

This function takes the computed average rates of clonal exclusivity from the data (m1, ... mN), which are specific to each patient and averaged over several trees from the collection of tree inferences. It also takes the histogram for each patient, of the values of how often a pair was clonally exclusive over the number of trees it was mutated in. It also takes the empirical cumulative distribution function (ECDF) which was generated with `generate_ecdf_test_stat`. It then computes the p-value of the simulated pairs under the null.

Value

The return value is a list of tibbles with a tibble for each number of patients, a pair can be mutated in. Each tibble contains the columns 'test_statistic', 'mle_delta', and then num_pat_pair columns of the rates of each patient 'pat1', 'pat2', ...; as well as num_pat_pair columns with the information about each patient, in how many trees the pair was occurring and in how many trees the pair was clonally exclusive. The tibble also contains a column 'pval' with the p-value of the simulated pair. The list of tibbles is of length minnum_pat_pair_max, length(avg_rates_m).

Author(s)

Ariane L. Moore

Examples

```
clone_tbl <- dplyr::tibble("file_name" =
       rep(c(rep(c("fn1", "fn2"), each=3)), 2),
       "patient_id"=rep(c(rep(c("pat1", "pat2"), each=3)), 2),
       "altered_entity"=c(rep(c("geneA", "geneB", "geneC"), 4)),
       "clone1"=c(0, 1, 0, 1, 0, 1, 0, 1, 1, 1, 0, 0),
       "clone2"=c(1, 0, 1, 0, 1, 1, 1, 0, 0, 1, 0, 1),
       "tree_id"=c(rep(5, 6), rep(10, 6)))
clone_tbl_pat1 <- dplyr::filter(clone_tbl, patient_id == "pat1")
clone_tbl_pat2 <- dplyr::filter(clone_tbl, patient_id == "pat2")
rates_exmpl_1 <- compute_rates_clon_excl(clone_tbl_pat1)
rates_exmpl_2 <- compute_rates_clon_excl(clone_tbl_pat2)
avg_rates_m <- apply(cbind(rates_exmpl_1, rates_exmpl_2), 2, mean)
names(avg_rates_m) <- c(names(rates_exmpl_1)[1], names(rates_exmpl_2)[1])
values_clon_excl_num_trees_pat1 <- get_hist_clon_excl(clone_tbl_pat1)
values_clon_excl_num_trees_pat2 <- get_hist_clon_excl(clone_tbl_pat2)
list_of_num_trees_all_pats <-
 list(pat1=values_clon_excl_num_trees_pat1[[1]],
      pat2=values_clon_excl_num_trees_pat2[[1]])
list_of_clon_excl_all_pats <-
  list(pat1=values_clon_excl_num_trees_pat1[[2]],
       pat2=values_clon_excl_num_trees_pat2[[2]])
```

```
num_pat_pair_max <- 2
num_pairs_sim <- 10
ecdf_list <- generate_ecdf_test_stat(avg_rates_m,
                           list_of_num_trees_all_pats,
                           list_of_clon_excl_all_pats,
                           num_pat_pair_max, num_pairs_sim)
sim_res <- generate_test_stat_hist(avg_rates_m,
                                   list_of_num_trees_all_pats,
                                   list_of_clon_excl_all_pats,
                                   ecdf_list,
                                   num_pat_pair_max,
                                   num_pairs_sim)
```

`get_hist_clon_excl` *Compute all values of how often gene pairs were clonally exclusive across all trees for a patient.*

Description

Compute all values of how often gene pairs were clonally exclusive/all trees for a patient.

Usage

```
get_hist_clon_excl(clone_tbl)
```

Arguments

`clone_tbl`	A tibble containing the columns 'altered_entity', and then a column for each clone in the tumor, e.g. 'clone1', 'clone2', 'clone3'. It also contains the column 'tree_id', which specifies which tree of the collection of tree inferences was used. This tibble can be generated e.g. from the `Cloe` output.

Details

It computes a histogram of the following two values: Amomg all gene/pathway pairs in a patient, the number of trees in which the both entities of a pair are assigned to a clone at all, and the number of trees in which the pair is clonally exclusive.

Value

A list with two vectors: The numbers of how often gene pairs were mutated across trees, and the numbers of how often they were clonally exclusive. The order of these two vectors is matching, i.e. the ith entry in each vector refers to the same gene pair.

Author(s)

Ariane L. Moore

Examples

```
clone_tbl <- dplyr::tibble(
    altered_entity=c(paste("gene", seq_len(10), sep="")),
    clone1=c(rep(0,10)),
    clone2=c(sample(c(0,1), 10, replace=TRUE)),
    clone3=c(sample(c(0,1), 10, replace=TRUE)),
    clone4=c(sample(c(0,1), 10, replace=TRUE)),
    tree_id=c(rep(5, 5), rep(10, 5)) )
get_hist_clon_excl(clone_tbl)
```

`get_hist_clon_excl_this_pat_this_pair` *Check for a pair how often it was mutated in the current patient across trees, and how often also clonally exclusive.*

Description

Check for a pair how often it was mutated in the current patient across trees, and how often also clonally exclusive.

Usage

```
get_hist_clon_excl_this_pat_this_pair(entA, entB, clone_tbl)
```

Arguments

`entA`	One gene/pathway of the pair
`entB`	The other gene/pathway of the pair
`clone_tbl`	A tibble containing the columns 'altered_entity', and then a column for each clone in the tumor, e.g. 'clone1', 'clone2', 'clone3'. It also contains the column 'tree_id', which specifies which tree of the collection of tree inferences was used. This tibble can be generated e.g. from the `Cloe` output.

Value

A vector with the values of in how many trees the pair was mutated, and in how many of those it was clonally exclusive.

Author(s)

Ariane L. Moore

Examples

```
clone_tbl <- dplyr::tibble(
    altered_entity=c(paste("gene", seq_len(10), sep="")),
    clone1=c(rep(0,10)),
    clone2=c(sample(c(0,1), 10, replace=TRUE)),
    clone3=c(sample(c(0,1), 10, replace=TRUE)),
    clone4=c(sample(c(0,1), 10, replace=TRUE)),
    tree_id=c(rep(5, 5), rep(10, 5)) )
get_hist_clon_excl_this_pat_this_pair("gene1", "gene2", clone_tbl)
```

---

get_rate_diff_branch_ent_pair

Compute rate of being in different branches/clones.

Description

Compute the rate of mutated gene/pathway pairs being in different branches.

Usage

```
get_rate_diff_branch_ent_pair(clone_tbl)
```

Arguments

clone_tbl	A tibble containing the columns 'altered_entity', and then a column for each clone

Details

Given the output of a tool that identifies clones within tumors and their phylogenetic history, this function computes the rate of mutated gene/pathway pairs being in different branches. That is, it will calculate the number of times mutated gene/pathway pairs are in different branches/clones divided by the total number of all mutated gene/pathway pairs.

Value

The rate of occurrence of mutated gene/pathway pairs being in different clones.

Author(s)

Ariane L. Moore in the tumor, e.g. 'clone1', 'clone2', 'clone3'. This tibble can be generated e.g. from the `Cloe` output.

Examples

```
clone_tbl <- dplyr::tibble(
    altered_entity=c(paste("gene", seq_len(10), sep="")),
    clone1=c(rep(0,10)),
    clone2=c(sample(c(0,1), 10, replace=TRUE)),
    clone3=c(sample(c(0,1), 10, replace=TRUE)),
    clone4=c(sample(c(0,1), 10, replace=TRUE)))
get_rate_diff_branch_ent_pair(clone_tbl)
```

`heatmap_clones_gene_pat` *Heatmaps of gene pairs of interest*

Description

This function plots the heatmaps of final gene clone matrices.

Usage

```
heatmap_clones_gene_pat(pairs_of_interest, clone_tbl, all_genes_tbl,
  first_clone_is_N = FALSE, output_pdf = "direct")
```

Arguments

`pairs_of_interest`
: The tibble containing the pairs of genes/pathways that should be visualized in the heatmap. This may be, e.g. the gene pairs were mle_delta > 0, qval < 0.1, and num_patients > 1. It contains the columns 'entity_A', and 'entity_B', and can be generated with GeneAccord. For the plot, the function will attempt to map the gene ID's from ensembl ID to gene name. However, if the input genes are not ensembl IDs, it does not matter.

`clone_tbl`
: The tibble containing the information of which gene/pathway is mutated in which clone from which patient. Here, it is assumed that only one tree from the collection of trees was chosen per patient.

`all_genes_tbl`
: A tibble with all genes ensembl id's and hgnc symbols. Can be created with `create_ensembl_gene_tbl_hg`.

`first_clone_is_N`
: Logical indicating whether the first clone column is actually representing the normal or germline, and is not a tumor clone. In that case, it will have the name 'N', and all other columns will be one clone number smaller, e.g. 'clone2' is then actually 'clone1' etc. Default: FALSE.

`output_pdf`
: The name of the pdf to be generated. Or if output_pdf is "direct", then the plot is generated directly and not to a pdf. Default: "direct"

Details

After running the GeneAccord, one may want to visualize the gene clone heatmap for significant gene pairs.

Value

None, the function plots a gene-to-clone assignment heatmap.

Author(s)

Ariane L. Moore

Examples

```
pairs_of_interest <- dplyr::tibble(entity_A="SETD2",
                           entity_B="BAP1")
clone_tbl <- dplyr::tibble(
               file_name=c("X.csv", "X.csv", "Y.csv", "Y.csv"),
               patient_id=c("X", "X", "Y", "Y"),
               altered_entity=c("SETD2", "BAP1", "SETD2", "BAP1"),
               clone1=c(0, 1, 1, 0),
               clone2=c(1, 0, 0, 1))
## Not run: all_genes_tbl <- create_ensembl_gene_tbl_hg()
all_genes_tbl_example <- dplyr::tibble(
                 ensembl_gene_id=c("ENSG00000181555",
                 "ENSG00000163930"),
                 hgnc_symbol=c("SETD2", "BAP1"))
heatmap_clones_gene_pat(pairs_of_interest, clone_tbl,
all_genes_tbl_example)
```

---

hgnc_to_ensembl *Get the ensembl gene id for a hgnc gene symbol.*

Description

Map a given hgnc gene symbol to the ensembl gene id.

Usage

```
hgnc_to_ensembl(this_hgnc, all_genes_tbl)
```

Arguments

this_hgnc	The hgnc gene symbol of a gene.
all_genes_tbl	A tibble with all genes ensembl id's and hgnc gene symbols.

Details

For a hgnc gene symbol and a tibble with all genes as input, this function returns the matching ensembl gene id. The tibble with all genes can be generated with create_ensembl_gene_tbl_hg.

Value

The matching ensembl gene id. In case several ensembl gene id's were found, they are all returned with ";" as a separator.

Author(s)

Ariane L. Moore

Examples

```
## Not run:
all_genes_tbl <- create_ensembl_gene_tbl_hg()
hgnc_to_ensembl("VHL", all_genes_tbl)
hgnc_to_ensembl("PBRM1", all_genes_tbl)

## End(Not run)
```

---

is_diff_branch_ent_pair

Check whether pair is in different branches/clones.

Description

Check whether a given pair of mutated genes/pathways is in different branches/clones.

Usage

```
is_diff_branch_ent_pair(ent1, ent2, clone_tbl)
```

Arguments

ent1	One mutated gene/pathway from the pair.
ent2	The other mutated gene/pathway from the pair.
clone_tbl	A tibble containing the columns 'altered_entity', and then a column for each clone in the tumor, e.g. 'clone1', 'clone2', 'clone3'. This tibble can be generated e.g. from the Cloe output.

Details

Given two mutated genes or pathways and the clone tibble as described in get_rate_diff_branch_ent_pair, this function returns TRUE or FALSE for whether the pair is mutated in different branches/clones.

Value

TRUE or FALSE for whether or not the pair is mutated in different clones/in different branches of the tree.

Author(s)

Ariane L. Moore

Examples

```
clone_tbl <- dplyr::tibble(
    altered_entity=c(paste("gene", seq_len(10), sep="")),
    clone1=c(rep(0,10)),
    clone2=c(sample(c(0,1), 10, replace=TRUE)),
    clone3=c(sample(c(0,1), 10, replace=TRUE)),
    clone4=c(sample(c(0,1), 10, replace=TRUE)))
is_diff_branch_ent_pair("gene1", "gene2", clone_tbl)
```

---

list_of_clon_excl_all_pats

The list with the histogram of how often pairs are clonally exclusive across the collection of trees

---

Description

This is a named list whose entries for each patient are the histograms of how often pairs are clonally exclusive in all trees of a patient.

Usage

```
list_of_clon_excl_all_pats
```

Format

A list whose entries are named after the patient, and they contain vectors with the numbers of how often the pairs in this patient are clonally exclusive accross trees.

Source

The histogram can be generated for each patient separately with `get_hist_clon_excl` as demonstrated in the vignette.

`list_of_num_trees_all_pats`

The list with the histogram of how often pairs are occurring across the collection of trees

---

Description

This is a named list whose entries for each patient are the histograms of how often pairs occurr in all trees of a patient.

Usage

```
list_of_num_trees_all_pats
```

Format

A list whose entries are named after the patient, and they contain vectors with the numbers of how often the pairs in this patient occur accross trees.

Source

The histogram can be generated for each patient separately with `get_hist_clon_excl` as demonstrated in the vignette.

`map_pairs_to_hgnc_symbols`

Map the ensembl gene ids to hgnc symbols from a tibble

Description

Map the ensembl gene ids to hgnc symbols from a tibble with pairs.

Usage

```
map_pairs_to_hgnc_symbols(pairs_of_interest_tbl, all_genes_tbl)
```

Arguments

`pairs_of_interest_tbl`	A tibble containing pairs of mutated genes/pathways. More precisely, it contains the columns 'entity_A' and 'entity_B'.
`all_genes_tbl`	A tibble with all genes ensembl id's and hgnc symbols. Can be created with `create_ensembl_gene_tbl_hg`.

Details

After having extracted the pairs of interest, it is of interest to know the genes hgnc symbols of the pairs. Here, it is assumed that the current gene identifier of the pairs are ensembl gene ids. They will be mapped to the corresponding hgnc symbols.

Value

A tibble similar to the input `pairs_of_interest_tbl` but with two additional columns, namely 'hgnc_gene_A', and 'hgnc_gene_B'. The column 'hgnc_gene_A' contains the hgnc gene symbol of 'entity_A', and the column 'hgnc_gene_B' the one of 'entity_B'.

Author(s)

Ariane L. Moore

Examples

```
## Not run:
pairs_of_interest <- dplyr::tibble(
    entity_A=c("ENSG00000181143", "ENSG00000163939"),
    entity_B=c("ENSG00000141510", "ENSG00000163930"))
all_genes_tbl <- create_ensembl_gene_tbl_hg()
map_pairs_to_hgnc_symbols(pairs_of_interest, all_genes_tbl)

## End(Not run)
```

---

merge_clones_identical_ents

Merge identical entities in clone tibble from one patient

Description

Merge clone profile of identical entities in clone tibble from one patient

Usage

```
merge_clones_identical_ents(clone_tbl)
```

Arguments

`clone_tbl`	The clone tibble as generated with `create_tbl_ent_clones` from one patient.

Details

Given a clone tibble as created with `create_tbl_ent_clones` from one patient and where the entities were possibly mapped from genes to pathways, this function checks whether there were several entities mapped to the same new entity. If so, the clone profile will be merged. This can be the case, for instance, of two mutated genes are in the same pathway(s).

Value

The same tibble but in case there were several identical genes/pathways in the same patient with different clone profiles, their profile will be merged together. This can happen if, e.g. two genes with different clone profiles are in the same pathway. When mapping them to the pathways, there will be two identical 'altered_entities' with different clone profiles. These profiles would be merged by this function because the pathway is affected in the union of clones were the two genes were mutated.

Author(s)

Ariane L. Moore

Examples

```
clone_tbl <- dplyr::tibble(
    file_name=c(rep("fn1", 4)),
    patient_id=c(rep("pat1", 4)),
    altered_entity=c("pw1", "pw1",
                     "pw2", "pw3"),
    clone1=c(1, 0, 1, 0),
    clone2=c(0, 1, 0, 1),
    clone3=c(1, 1, 0, 1),
    clone4=c(0, 1, 0, 0))
merge_clones_identical_ents(clone_tbl)
```

---

pairs_in_patients_hist

Pairs in how many patients histogram

Description

Check in how many patients pairs are mutated in

Usage

```
pairs_in_patients_hist(clone_tbl)
```

Arguments

clone_tbl	The tibble containing the information of which gene/pathway is mutated in which clone from which patient and in which tree from the collection of trees. Can be generated with create_tbl_tree_collection for each patient separately and then appended.

Details

After having created the tibble with all gene-to-clone assignments from all patients and the whole collection of trees, we're interested in how many patients tha pairs are mutated in. This function creats a histogram that shows in how many patients the pairs are mutated in.

Value

The tibble that summarizes the number of pairs that occur in how many patients.

Author(s)

Ariane L. Moore

Examples

```
clone_tbl <- dplyr::tibble("file_name" =
    rep(c(rep(c("fn1", "fn2"), each=3)), 2),
    "patient_id"=rep(c(rep(c("pat1", "pat2"), each=3)), 2),
    "altered_entity"=c(rep(c("geneA", "geneB", "geneC"), 4)),
    "clone1"=c(0, 1, 0, 1, 0, 1, 0, 1, 1, 1, 0, 0),
    "clone2"=c(1, 0, 1, 0, 1, 1, 1, 0, 0, 1, 0, 1),
    "tree_id"=c(rep(5, 6), rep(10, 6)))
pairs_in_patients_hist(clone_tbl)
```

---

plot_ecdf_test_stat *Plot empirical cumulative distribution functions of the test statistic under the null.*

Description

This function plots the ECDFs of the test statistic under the null hypothesis.

Usage

```
plot_ecdf_test_stat(ecdf_list, plot_idx = c(2, 3), num_panel_rows = 1,
  output_pdf = "direct")
```

Arguments

ecdf_list	The list of ECDF's as generated with generate_ecdf_test_stat.
plot_idx	The index of which of the list entries of the ecdf_list to plot. Default: c(2,3).
num_panel_rows	The ECDF's will be plotted altogether, hence par(mfrow=c(x,y)) is used. Here, x is the number of panel rows, which has to be set with this parameter, and y will be taken as ceil(#ECDF's/x). E.g., if you have 20 ECDF's in total, you can set num_panel_rows=4, and then your 20 ECDF's will be plotted in panels with four rows, and five columns. Default=1.
output_pdf	The name of the pdf to be generated. Or if output_pdf is "direct", then the plot is generated directly and not to a pdf. Default: "direct".

Details

The ECDF's of the test statistic under the null for a data set can be generated with generate_ecdf_test_stat. Afterwards, they can be visualized with this function. It is assumed that the first ECDF in the ecdf_list is the ECDF for the case where pairs are mutated in two patients.

Value

None, the function plots ecdf curves.

Author(s)

Ariane L. Moore

Examples

```
avg_rates_m <- c(pat1=0.1, pat2=0.034, pat3=0.21, pat4=0.063)
list_of_num_trees_all_pats <- list(pat1=c(20, 20, 19),
                                   pat2=c(20, 18, 20),
                                   pat3=c(19, 20, 20),
                                   pat4=c(20, 20, 20))
list_of_clon_excl_all_pats <- list(pat1=c(5, 0, 1),
                                   pat2=c(10, 2, 0),
                                   pat3=c(18, 12, 0),
                                   pat4=c(0, 2, 0))
num_pat_pair_max <- 2
num_pairs_sim <- 10
ecdf_list <- generate_ecdf_test_stat(avg_rates_m,
                                     list_of_num_trees_all_pats,
                                     list_of_clon_excl_all_pats,
                                     num_pat_pair_max,
                                     num_pairs_sim)
plot_ecdf_test_stat(ecdf_list, plot_idx=2)
```

---

plot_rates_clon_excl *Barplot of rates of clonal exclusivity and number of clones.*

Description

This function plots the average rates of clonal exclusivity for each patient.

Usage

```
plot_rates_clon_excl(avg_rates_m, clone_tbl, output_pdf = "direct")
```

Arguments

avg_rates_m	A named vector with the average rates of clonal exclusivity for each patient. The name of each element is the patient id to be used in the barplot.
clone_tbl	The tibble containing the gene-to-clone assignments from all patients and all trees from the collection of trees.
output_pdf	The name of the pdf to be generated. Or if output_pdf is "direct", then the plot is generated directly and not to a pdf. Default: "direct"

Details

In addition to the average rate of clonal exclusivity, it also visualizes the average number of clones of each patient.

Value

None, the function plots the average rates of clonal exclusivity.

Author(s)

Ariane L. Moore

Examples

```
clone_tbl <- dplyr::tibble(
            "file_name"=rep(c(rep(c("fn1", "fn2"), each=3)), 2),
            "patient_id"=rep(c(rep(c("pat1", "pat2"), each=3)), 2),
            "altered_entity"=c(rep(c("geneA", "geneB", "geneC"), 4)),
            "clone1"=c(0, 1, 0, 1, 0, 1, 0, 1, 1, 1, 0, 0),
            "clone2"=c(1, 0, 1, 0, 1, 1, 1, 0, 0, 1, 0, 1),
            "tree_id"=c(rep(1, 6), rep(2, 6)))
avg_rates_m <- c(pat1=0.014, pat2=0.3)
plot_rates_clon_excl(avg_rates_m, clone_tbl)
```

---

take_pairs_and_get_patients

Get the patients in which pairs are mutated

Description

Take the final pairs and return the patients id's, in which they are mutated, and the patients id's in which they are clonally exclusive.

Usage

```
take_pairs_and_get_patients(clone_tbl_all_trees, pairs_of_interest_tbl)
```

Arguments

`clone_tbl_all_trees`
: The tibble containing the information of which gene/pathway is mutated in which clone from which patient across a collection of trees. Can be generated with `create_tbl_tree_collection`, repeatedly for each patient, and then combining them.

`pairs_of_interest_tbl`
: A tibble containing pairs of mutated genes/pathways. More precisely, it contains the columns 'entity_A' and 'entity_B'.

Details

This function takes the final pairs of interest, and returns a tibble with the information for each gene pair, in which patient the pair was mutated, and in which of these patients the pair was clonally exclusive in the majority of the trees in the tree inference collection.

Value

A tibble similar to the input `pairs_of_interest_tbl` but with two additional columns, namely 'mutated_in', and 'clonally_exclusive_in'. The column 'mutated_in' contains the patient id's that the pair is mutated in separated by a semicolon. The column 'clonally_exclusive_in' contains the semicolon separated patient id's of the ones in which the pairs was also clonally exclusive in the majority of the trees in the collection of tree inferences.

Author(s)

Ariane L. Moore

Examples

```
clone_tbl <- dplyr::tibble(file_name=rep(c(rep(c("fn1", "fn2"),
    each=3)), 2),
    patient_id=rep(c(rep(c("pat1", "pat2"),
    each=3)), 2),
    altered_entity=c(rep(c("geneA", "geneB", "geneC"),
    4)),
    clone1=c(0, 1, 0, 1, 0, 1, 0,
    1, 1, 1, 0, 0),
    clone2=c(1, 0, 1, 0, 1, 1, 1,
    0, 0, 1, 0, 1),
    tree_id=c(rep(5, 6), rep(10, 6)))
pairs_of_interest <- dplyr::tibble(entity_A=c("geneA", "geneB"),
    entity_B=c("geneB", "geneC"))
take_pairs_and_get_patients(clone_tbl, pairs_of_interest)
```

---

vis_pval_distr_num_pat

Plot histogram and empirical cumulative distribution function of p-values.

Description

This function visualizes the distribution of p-values.

Usage

```
vis_pval_distr_num_pat(res_sim, output_pdf = "direct")
```

Arguments

res_sim	tibble containing the simulated pairs of genes/pathways. It contains the columns 'num_patients', and 'pval', and can be generated with generate_test_stat_hist and then concatenating the tibbles.
output_pdf	The name of the pdf to be generated. Or if output_pdf is "direct", then the plot is generated directly and not to a pdf. Default: "direct"

Details

It is especially useful, when exploring the results with simulated data under the null hypothesis, i.e. when delta is zero. In that scenario, the p-values are expected to be uniformly distributed. This function can take the p-values from generate_test_stat_hist where the concatenated tibble contains different values for 'num_pat_pair', i.e. the number of patients the simulated pairs are mutated in. The input tibble is expected to have the two columns 'pval', and 'num_patients'. Left panel: histogram of all p-values from the whole tibble. Right panel: ecdf of the p-values with different colors for different numbers of patients that the pairs were mutated in.

Value

None, the function plots a p-value histogram.

Author(s)

Ariane L. Moore

Examples

```
res_sim <- dplyr::tibble(num_patients=c(rep(2,100),
                         rep(3,100), rep(4,100)),
                         pval=c(runif(300)))
vis_pval_distr_num_pat(res_sim)
```

---

write_res_pairs_to_disk

Write resulting significant pairs to disk

Description

Write the resulting significant pairs tibble to disk as a tab-separated file.

Usage

```
write_res_pairs_to_disk(sig_pairs, avg_rates_m, tsv_file, num_digits = 2)
```

Arguments

sig_pairs	The tibble containing the significant pairs of mutated genes/pathways.
avg_rates_m	The average rates of clonal exclusivity for each patient. The name of each rate is the respective patient identifier.
tsv_file	The path to the tsv-file to which the results should be written.
num_digits	The number of digits after the comma of the average rate m, the p-value and the q-value (adjusted p-value). Default: 2.

Details

After having extracted the significant pairs. The tibble can be saved as a tab-separated file. It is assumed that the input tibble has the columns 'hgnc_gene_A', 'hgnc_gene_B', 'pval', 'qval', 'mutated_in', 'clonally_exclusive_in'.

Value

The tibble that is written to disk. It has the columns 'Gene A', 'Gene B', 'P-value', 'Adjusted p-value', 'Mutated in (rate)', 'Clonally exclusive in'.

Author(s)

Ariane L. Moore

Examples

```
sig_pairs <- dplyr::tibble(hgnc_gene_A=c("VHL", "BAP1"),
    hgnc_gene_B=c("PTEN", "KIT"),
    pval=c(0.001, 0.002),
    qval=c(0.01, 0.02),
    mutated_in=c("pat1; pat2", "pat1; pat2"),
    clonally_exclusive_in=c("pat1; pat2",
    "pat2"))
avg_rates_m <- c(pat1=0.0034, pat2=0.0021)
write_res_pairs_to_disk(sig_pairs, avg_rates_m, "test.tsv")
file.remove("test.tsv")
```

Index