Package ‘cardelino’

May 13, 2024

Type Package

Title Clone Identification from Single Cell Data

Version 1.7.0

Description Methods to infer clonal tree configuration for a population of cells using single-cell RNA-seq data (scRNA-seq), and possibly other data modalities. Methods are also provided to assign cells to inferred clones and explore differences in gene expression between clones. These methods can flexibly integrate information from imperfect clonal trees inferred based on bulk exome-seq data, and sparse variant alleles expressed in scRNA-seq data. A flexible beta-binomial error model that accounts for stochastic dropout events as well as systematic allelic imbalance is used.

License GPL-3

URL https://github.com/single-cell-genetics/cardelino

BugReports https://github.com/single-cell-genetics/cardelino/issues

Depends R (>= 4.2), stats

Imports combinat, GenomeInfoDb, GenomicRanges, ggplot2, ggtree, Matrix, matrixStats, methods, pheatmap, snpStats, S4Vectors, utils, VariantAnnotation, vcfR

Suggests BiocStyle, foreach, knitr, pcaMethods, rmarkdown, testthat, VGAM

Enhances doMC

VignetteBuilder knitr

biocViews SingleCell, RNASeq, Visualization, Transcriptomics, GeneExpression, Sequencing, Software, ExomeSeq

Encoding UTF-8

NeedsCompilation yes

RoxygenNote 7.2.1

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

git_branch devel

git_last_commit abf5cef

git_last_commit_date 2024-04-30

Repository Bioconductor 3.20

Date/Publication 2024-05-13

Author Jeffrey Pullin [aut],
Yuanhua Huang [aut],
Davis McCarthy [aut, cre]

Maintainer Davis McCarthy <dmccarthy@svi.edu.au>

Contents

`assign_cells_to_clones`

Assign cells to clones from cardelino results

Description

Assign cells to clones from cardelino results

Usage

```
assign_cells_to_clones(prob_mat, threshold = 0.5)
```

Arguments

`prob_mat`	numeric matrix (cells x clones) of clone posterior probabilities as output by `clone_id`
`threshold`	numeric(1), posterior probability threshold for cell-clone assignment: if posterior probability is above threshold, assign cell to clone, otherwise leave cell "unassigned"

Value

a `data.frame` with cell ID, assigned clone label and maximum posterior probability across clones.

Author(s)

Davis McCarthy

Examples

```
data(example_donor)
assignments <- clone_id(A_clone, D_clone, Config = tree$Z, inference = "EM")
df <- assign_cells_to_clones(assignments$prob)
head(df)
table(df$clone)
```

assign_scores — *Scoring the simulation in assignment of singlets and doublets*

Description

Scoring the simulation in assignment of singlets and doublets

Usage

```
assign_scores(prob, I_sim, cutoff = seq(0, 1, 0.001))
```

Arguments

prob	Probability matrix for each cell to each component
I_sim	The true identity of assignment from simulation
cutoff	A list of cutoffs from 0 to 1

Value

A list with components: df_sg, the recall/precision data.frame calculated by `multiPRC()`, AUC_sg, the AUC calculated by `multiPRC()`, df_db, the recall/precision data.frame calculated by `binaryPRC()` and AUC_db the AUC calculated by `binaryPRC()`. Note that `multiPRC()` is run on a multiclass version of the problem and `binaryPRC` is run on a binarised version of the problem.

A_clone — *A matrix of read numbers of alternative alleles for clone ID*

Description

This matrix contains read numbers of alternative alleles for 34 somatic variants across 428 cells, from one example scRNA-seq sample

Usage

```
example_donor
```

Format

a matrix of float

Value

NULL, but makes available a matrix

Author(s)

Yuanhua Huang, Davis McCarthy, 2018-06-25

Source

A fibroblast sample from HipSci project

A_germline — *A matrix of read numbers of alternative alleles*

Description

This matrix contains read numbers of alternative alleles for 34 germline variants (near the somatic variants) across 428 cells, from one example scRNA-seq sample

Usage

```
example_donor
```

Format

a matrix of float

Value

NULL, but makes available a matrix

Author(s)

Yuanhua Huang, Davis McCarthy, 2018-06-25

Source

A fibroblast sample from HipSci project

binaryPRC *Precision-recall curve for binary label prediction*

Description

Precision-recall curve for binary label prediction

Usage

```
binaryPRC(
  scores,
  labels,
  cutoff = NULL,
  cut_direction = ">=",
  add_cut1 = FALSE,
  empty_precision = 1
)
```

Arguments

scores	Prediction score for each sample
labels	True labels for each sample, e.g., from simulation
cutoff	A vector of cutoffs; if NULL use all unique scores
cut_direction	A string to compare with cutoff: >=, >, <=, <
add_cut1	Logical value; if True, manually add a cutoff of 1
empty_precision	Float value for default precision if no any recall

Value

A data.frame containing recall and precision values at various cutoffs.

Examples

```
scores <- 1:10
labels <- c(0, 0, 0, 1, 0, 1, 0, 1, 1, 1)
binaryPRC(scores, labels)

# Extra arguments.
binaryPRC(scores, labels, cutoff = seq(1, 10, by = 2))
binaryPRC(scores, labels, cut_direction = ">")
binaryPRC(scores, labels, add_cut1 = TRUE)
```

binaryROC *ROC curve for binary label prediction*

Description

ROC curve for binary label prediction

Usage

```
binaryROC(
  scores,
  labels,
  cutoff = NULL,
  cut_direction = ">=",
  add_cut1 = TRUE,
  cutoff_point = 0.9
)
```

Arguments

scores	Prediction score for each sample
labels	True labels for each sample, e.g., from simulation
cutoff	A vector of cutoffs; if NULL use all unique scores
cut_direction	A string to compare with cutoff: >=, >, <=, <
add_cut1	Logical value; if True, manually add a cutoff of 1
cutoff_point	Numeric value; additional cutoff value

Value

A data.frame containing AUC and AUPRC at various cutoffs.

Examples

```
scores <- 1:10
labels <- c(0, 0, 0, 1, 0, 1, 0, 1, 1, 1)
binaryROC(scores, labels)

# Extra arguments.
binaryROC(scores, labels, cutoff = seq(1, 10, by = 2))
binaryROC(scores, labels, cut_direction = ">")
binaryROC(scores, labels, add_cut1 = TRUE)
```

Clone ID *Infer clonal identity of single cells*

Description

Infer clonal identity of single cells

Assign cells to clones using an EM algorithm

Assign cells to clones using a Gibbs sampling algorithm

Usage

```
clone_id(
  A,
  D,
  Config = NULL,
  n_clone = NULL,
  Psi = NULL,
  relax_Config = TRUE,
  relax_rate_fixed = NULL,
  inference = "sampling",
  n_chain = 1,
  n_proc = 1,
  verbose = TRUE,
  ...
)

clone_id_EM(
  A,
  D,
  Config,
  Psi = NULL,
  min_iter = 10,
  max_iter = 1000,
  logLik_threshold = 1e-05,
  verbose = TRUE
)

clone_id_Gibbs(
  A,
  D,
  Config,
  Psi = NULL,
  relax_Config = TRUE,
  relax_rate_fixed = NULL,
  relax_rate_prior = c(1, 9),
  keep_base_clone = TRUE,
```

```
  prior0 = c(0.2, 99.8),
  prior1 = c(0.45, 0.55),
  min_iter = 5000,
  max_iter = 20000,
  buin_frac = 0.5,
  wise = "variant",
  relabel = FALSE,
  verbose = TRUE
)
```

Arguments

A	variant x cell matrix of integers; number of alternative allele reads in variant i cell j
D	variant x cell matrix of integers; number of total reads covering variant i cell j
Config	variant x clone matrix of binary values. The clone-variant configuration, which encodes the phylogenetic tree structure. This is the output Z of Canopy
n_clone	integer(1), the number of clone to reconstruct. This is in use only if Config is NULL
Psi	A vector of float. The fractions of each clone, output P of Canopy
relax_Config	logical(1), If TRUE, relaxing the Clone Configuration by changing it from fixed value to act as a prior Config with a relax rate.
relax_rate_fixed	numeric(1), If the value is between 0 to 1, the relax rate will be set as a fix value during updating clone Config. If NULL, the relax rate will be learned automatically with relax_rate_prior.
inference	character(1), the method to use for inference, either "sampling" to use Gibbs sampling (default) or "EM" to use expectation-maximization (faster)
n_chain	integer(1), the number of chains to run, which will be averaged as an output result
n_proc	integer(1), the number of processors to use. This parallel computing can largely reduce time when using multiple chains
verbose	logical(1), should the function output verbose information as it runs?
...	arguments passed to clone_id_Gibbs or clone_id_EM (as appropriate)
min_iter	A integer. The minimum number of iterations in the Gibbs sampling. The real iteration may be longer until the convergence.
max_iter	A integer. The maximum number of iterations in the Gibbs sampling, even haven't passed the convergence diagnosis
logLik_threshold	A float. The threshold of logLikelihood increase for detecting convergence.
relax_rate_prior	numeric(2), the two parameters of beta prior distribution of the relax rate for relaxing the clone Configuration. This mode is used when relax_relax is NULL.

`keep_base_clone`
: bool(1), if TRUE, keep the base clone of Config to its input values when relax mode is used.

`prior0`
: numeric(2), alpha and beta parameters for the Beta prior distribution on the inferred false positive rate.

`prior1`
: numeric(2), alpha and beta parameters for the Beta prior distribution on the inferred (1 - false negative) rate.

`buin_frac`
: numeric(1), the fraction of chain as burn-in period

`wise`
: A string, the wise of parameters for theta1: global, variant, element.

`relabel`
: bool(1), if TRUE, relabel the samples of both Config and prob during the Gibbs sampling.

Details

The two Bernoulli components correspond to false positive and false negative rates. The two binomial components correspond to the read distributions with and without the mutation present.

Value

If inference method is "EM", a list containing `theta`, a vector of two floats denoting the parameters of the two components of the base model, i.e., mean of Bernoulli or binomial model given variant exists or not, `prob`, the matrix of posterior probabilities of each cell belonging to each clone with fitted parameters, and `logLik`, the log likelihood of the final parameters.

If inference method is "sampling", a list containing: `theta0`, the mean of sampled false positive parameter values; `theta1` the mean of sampled (1 - false negative rate) parameter values; `theta0_all`, all sampled false positive parameter values; `theta1_all`, all sampled (1 - false negative rate) parameter values; `element`; `logLik_all`, log-likelihood for model for all sampled parameter sets; `prob_all`; `prob`, matrix with mean of sampled cell-clone assignment posterior probabilities (the key output of the model); `prob_variant`.

a list containing `theta`, a vector of two floats denoting the binomial rates given variant exists or not, `prob`, the matrix of posterior probabilities of each cell belonging to each clone with fitted parameters, and `logLik`, the log likelihood of the final parameters.

Author(s)

Yuanhua Huang and Davis McCarthy

Yuanhua Huang

Examples

```
data(example_donor)
assignments <- clone_id(A_clone, D_clone,
    Config = tree$Z,
    min_iter = 800, max_iter = 1200
)
prob_heatmap(assignments$prob)

assignments_EM <- clone_id(A_clone, D_clone,
```

```
    Config = tree$Z,
    inference = "EM"
)
prob_heatmap(assignments_EM$prob)
```

---

colMatch *Column match between two matrices by minimum mean absolute difference*

Description

Column match between two matrices by minimum mean absolute difference

Usage

```
colMatch(A, B, force = FALSE)
```

Arguments

A	The first matrix which will be matched
B	The second matrix, the return index will be used on
force	bool(1), If TRUE, force traversing all permutations of B to find the optimised match to A with computing cost of O(n!). Otherwise, use greedy search with computing cost of O(n^2).

Value

idx, the column index of B to be matched to A

Examples

```
matA <- matrix(sample(seq(12)), nrow = 3)
col_idx <- sample(4)
matB <- matA[, col_idx]
colMatch(matB, matA)
```

Config_all — *A list of tree configuration*

Description

This list of tree configuration between 3 clones to 10 clones, each element is a list with all possible tree matrix

Usage

```
config_all
```

Format

a list of list of matrix

Value

NULL, but makes available a list

Author(s)

Yuanhua Huang, Davis McCarthy, 2018-06-25

Source

PASTRI Python package

devianceIC — *Deviance Information Criterion for cardelino model*

Description

Deviance Information Criterion for cardelino model

Usage

```
devianceIC(logLik_all, logLik_post)
```

Arguments

logLik_all	A vector of numeric; the log likelihood of posterior sample, i.e., posterior samples of deviance
logLik_post	numeric(1); the log likelihood of mean posterior parameters, i.e., deviance of posterior means

Value

DIC, a float of deviance information criterion

Author(s)

Yuanhua Huang

---

donor_read_simulator *Reads simulator for donor identification*

Description

Reads simulator for donor identification

Usage

```
donor_read_simulator(
  GT,
  D_seed,
  sample_variants = FALSE,
  donor_size = NULL,
  beta_shapes = NULL,
  n_cell = 5000,
  doublet_rate = NULL
)
```

Arguments

GT	Variant-by-donor matrix for genotypes
D_seed	Variant-by-cell matrix for read coverage for generating depth, which be row sample and column sample both with replacement
sample_variants	logical(1), if TRUE, sample variants with replacement to the same size, otherwise not
donor_size	Vector of float for the fractions of each donor; default NULL means uniform
beta_shapes	A 3-by-2 matrix of beta parameters for genotypes: 0, 1, and 2; default NULL means matrix(c(0.2, 0.5, 99.8, 99.8, 0.5, 0.2), nrow = 3)
n_cell	An integer for number of total cells
doublet_rate	A float from 0 to 1 for doublet rate; default NULL means rate n_cell / 100000

Value

A list of various components of the simulated dataset.

D_clone — *A matrix of sequencing depths for clone ID*

Description

This matrix contains sequencing depths for 34 somatic variants across 428 cells, from one example scRNA-seq sample

Usage

```
example_donor
```

Format

a matrix of float

Value

NULL, but makes available a matrix

Author(s)

Yuanhua Huang, Davis McCarthy, 2018-06-25

Source

A fibroblast sample from HipSci project

D_germline — *A matrix of sequencing depths*

Description

This matrix contains sequencing depths for 34 germline variants (near the somatic variants) across 428 cells, from one example scRNA-seq sample

Usage

```
example_donor
```

Format

a matrix of float

Value

NULL, but makes available a matrix

Author(s)

Yuanhua Huang, Davis McCarthy, 2018-06-25

Source

A fibroblast sample from HipSci project

`D_input` *A matrix of sequencing depths*

Description

This matrix contains sequencing depths for 439 somatic variants across 151 cells, from one particular scRNA-seq sample, can be used to generate sequencing depths

Usage

```
simulation_input
```

Format

a matrix of float

Value

NULL, but makes available a matrix

Author(s)

Yuanhua Huang, Davis McCarthy, 2018-06-25

Source

A fibroblast sample from HipSci project

get_logLik — *Log likelihood of clone_id model It returns P(A, D | C, I, theta0, theta1)*

Description

Log likelihood of clone_id model It returns P(A, D | C, I, theta0, theta1)

Usage

```
get_logLik(A1, B1, Config, Assign, theta0, theta1)
```

Arguments

A1	variant x cell matrix of integers; number of alternative allele reads in variant i cell j
B1	variant x cell matrix of integers; number of reference allele reads in variant i cell j
Config	variant x clone matrix of float values. The clone-variant configuration probability, averaged by posterior samples
Assign	cells x clone matrix of float values. The cell-clone assignment probability, averaged by posterior samples
theta0	the binomial rate for alternative allele from config = 0
theta1	the binomial rate for alternative allele from config = 1

Value

logLik, a float of log likelihood

Author(s)

Yuanhua Huang

get_snp_matrices — *Get SNP data matrices from VCF object(s)*

Description

Get SNP data matrices from VCF object(s)

Usage

```
get_snp_matrices(vcf_cell, vcf_donor = NULL, verbose = TRUE, donors = NULL)
```

Arguments

vcf_cell	a CollapsedVCF object containing variant data for cells
vcf_donor	an optional CollapsedVCF object containing genotype data for donors
verbose	logical(1), should the function output verbose information as it runs?
donors	optional character vector providing a set of donors to use, by subsetting the donors present in the donor_vcf_file; if NULL (default) then all donors present in VCF will be used.

Value

a list containing A, a matrix of integers. Number of alteration reads in SNP i cell j. D, a matrix of integers. Number of reads depth in SNP i cell j. R, a matrix of integers. Number of reference reads in SNP i cell j. GT_cells, a matrix of integers for genotypes. The cell-SNP configuration. GT_donors, a matrix of integers for genotypes. The donor-SNP configuration.

Examples

```
vcf_cell <- read_vcf(system.file("extdata", "cells.donorid.vcf.gz",
                     package = "cardelino"))
vcf_donor <-  read_vcf(system.file("extdata", "donors.donorid.vcf.gz",
                       package = "cardelino"))
snp_data <- get_snp_matrices(vcf_cell, vcf_donor)
```

---

get_tree — *Get a clonal tree from a configuration matrix*

Description

Get a clonal tree from a configuration matrix

Usage

```
get_tree(Config, P = NULL, strictness = "lax")
```

Arguments

Config	variant x clone matrix of binary values. The clone-variant configuration, which encodes the phylogenetic tree structure. This is the output Z of Canopy
P	a one-column numeric matrix encoding the (observed or estimated) prevalence (or frequency) of each clone
strictness	character(1), a character string defining the strictness of the function if there are all-zero rows in the Config matrix. If "lax" then the function silently drops all-zero rows and proceeds. If "warn" then the function warns of dropping all-zero rows and proceeds. If "error" then the function throws an error is all-zero rows are detected.

Details

Output tree may be nonsensical if the input `Config` matrix does not define a coherent tree structure.

Value

An object of class "phylo" describing the tree structure. The output object also contains an element "sna" defining the clustering of variants onto the branches of the tree, and if `P` is non-null it also contains VAF (variant allele frequency), CCF (cell clone fraction) and clone prevalence values (computed from the supplied `P` argument).

Author(s)

Davis McCarthy

Examples

```
Configk3 <- matrix(c(
    rep(0, 15), rep(1, 8), rep(0, 7), rep(1, 5), rep(0, 3),
    rep(1, 7)
), ncol = 3)
tree_k3 <- get_tree(Config = Configk3, P = matrix(rep(1 / 3, 3), ncol = 1))
plot_tree(tree_k3)
```

`Geweke_Z` *Geweke diagnostic for MCMC sampling.*

Description

Geweke diagnostic for MCMC sampling.

Usage

```
Geweke_Z(X, first = 0.1, last = 0.5)
```

Arguments

`X`	A matrix of MCMC samples for N samples per K variables
`first`	A float between 0 and 1. The initial region of MCMC chain.
`last`	A float between 0 and 1. The final region of MCMC chain.

Value

`Z`, a vector of absolute value of Z scores for each variable. When |Z| <= 2, the sampling could be taken as converged.

Author(s)

Yuanhua Huang

heatmap.theme — *The theme of heatmaps for prob_heatmap and sites_heatmap*

Description

The theme of heatmaps for prob_heatmap and sites_heatmap

Usage

```
heatmap.theme(legend.position = "bottom", size = 12)
```

Arguments

legend.position	character, describes where to place legend on plot (passed to theme_gray)
size	numeric, base font size for plot (passed to theme_gray)

Value

a ggplot theme based on theme_gray

heat_matrix — *Plot heatmap from a matrix*

Description

Plot heatmap from a matrix

Usage

```
heat_matrix(mat, base_size = 12, digits = 2, show_value = FALSE)
```

Arguments

mat	A matrix to show, column by x-axis and row by y-axis
base_size	Numeric value for the base size in theme_bw
digits	Integer value for the number of digits to show
show_value	Logical value for showing the value for each element or not

Value

A ggplot heatmap visualization of the passed matrix.

Examples

```
mat <- matrix(rnorm(9), ncol = 3, nrow = 3) + diag(rnorm(3, 2, 0.1))
rownames(mat) <- paste0("sample_", letters[1:3])
colnames(mat) <- paste0("var_", 1:3)
heat_matrix(mat)

# Additional arguments.
heat_matrix(mat, base_size = 6)
heat_matrix(mat, show_value = TRUE)
heat_matrix(mat, show_value = TRUE, digits = 4)
```

---

load_cellSNP_vcf *Load sparse matrices A and D from cellSNP VCF file with filtering SNPs*

Description

Load sparse matrices A and D from cellSNP VCF file with filtering SNPs

Usage

```
load_cellSNP_vcf(
  vcf_file,
  min_count = 0,
  min_MAF = 0,
  max_other_allele = NULL,
  rowname_format = "full",
  keep_GL = FALSE
)
```

Arguments

vcf_file	character(1), path to VCF file generated from cellSNP
min_count	minimum count across all cells, e.g., 20
min_MAF	minimum minor allele fraction, e.g., 0.1
max_other_allele	maximum ratio of other alleles comparing to REF and ALT alleles; for cellSNP vcf, we recommend 0.05
rowname_format	the format of rowname: NULL is the default from vcfR, short is CHROM_POS, and full is CHROM_POS_REF_ALT
keep_GL	logical(1), if TRUE, check if GL (genotype probability) exists it will be returned

Value

A list with elements the matrices A and D and GL, the genotype probability. If keep_GL is false the GL element will be an empty list.

Examples

```
vcf_file <- system.file("extdata", "cellSNP.cells.vcf.gz",
    package = "cardelino"
)
input_data <- load_cellSNP_vcf(vcf_file)
```

---

load_GT_vcf *Load genotype VCF into numeric values: 0, 1, or 2*

Description

Note, the genotype VCF can be very big for whole genome. It would be more efficient to only keep the wanted variants and samples. bcftools does such jobs nicely.

Usage

```
load_GT_vcf(vcf_file, rowname_format = "full", na.rm = TRUE, keep_GP = TRUE)
```

Arguments

vcf_file	character(1), path to VCF file for donor genotypes
rowname_format	the format of rowname: NULL is the default from vcfR, short is CHROM_POS, and full is CHROM_POS_REF_ALT
na.rm	logical(1), if TRUE, remove the variants with NA values
keep_GP	logical(1), if TRUE, check if GP (genotype probability) exists it will be returned

Value

A list representing the loaded genotype information with two components: GT, the usual numeric representation of genotype and GP the genotype probabilities. Note that if keep_GP is false the GP component will be NULL.

Examples

```
vcf_file <- system.file("extdata", "cellSNP.cells.vcf.gz",
    package = "cardelino"
)
GT_dat <- load_GT_vcf(vcf_file, na.rm = FALSE)
```

mixBinom *EM algorithm for estimating binomial mixture model*

Description

EM algorithm for estimating binomial mixture model

Usage

```
mixBinom(
  k,
  n,
  n_components = 2,
  p_init = NULL,
  learn_p = TRUE,
  min_iter = 10,
  max_iter = 1000,
  logLik_threshold = 1e-05
)
```

Arguments

k	A vector of integers. number of success
n	A vector of integers. number of trials
n_components	A number. number of components
p_init	A vector of floats with length n_components, the initial value of p
learn_p	bool(1) or a vector of bool, whether learn each p
min_iter	integer(1). number of minimum iterations
max_iter	integer(1). number of maximum iterations
logLik_threshold	A float. The threshold of logLikelihood increase for detecting convergence

Value

a list containing p, a vector of floats between 0 and 1 giving the estimated success probability for each component, psi, estimated fraction of each component in the mixture, and prob, the matrix of fitted probabilities of each observation belonging to each component.

Examples

```
n1 <- array(sample(1:30, 50, replace = TRUE))
n2 <- array(sample(1:30, 200, replace = TRUE))
k1 <- apply(n1, 1, rbinom, n = 1, p = 0.5)
k2 <- apply(n2, 1, rbinom, n = 1, p = 0.01)
RV <- mixBinom(c(k1, k2), c(n1, n2))
```

mtx_to_df — *Convert a matrix to data frame*

Description

Convert a matrix to data frame

Usage

```
mtx_to_df(X)
```

Arguments

X	A matrix of values

Value

A data.frame version of the passed matrix.

Examples

```
mtx_to_df(matrix(seq(12), nrow = 3))
```

multiPRC — *Precision-recall curve for multi-class prediction*

Description

Precision-recall curve for multi-class prediction

Usage

```
multiPRC(
  prob_mat,
  simu_mat,
  marginal_mode = "best",
  cutoff = NULL,
  multiLabel.rm = TRUE,
  add_cut1 = FALSE
)
```

Arguments

prob_mat	Probability matrix for each cell to each component
simu_mat	The true identity of assignment from simulation
marginal_mode	A string for the mode to marginalize the column: best, second, or delta
cutoff	A list of cutoff; if NULL use all unique scores
multiLabel.rm	Logical value; if True, remove the samples with multiple labels
add_cut1	Logical value; if True, manually add a cutoff of 1

Value

A list with two components: df, a data.frame containing precision and recall values at various cutoffs and AUC, the overall AUC.

---

plot_config_diffs *Define a publication-style plot theme*

Description

Define a publication-style plot theme

Usage

```
plot_config_diffs(Config1, Config2, show_variant_names = FALSE)
```

Arguments

Config1	variant by clone matrix defining the first clonal structure
Config2	variant by clone matrix defining the second clonal structure
show_variant_names	logical(1), should the variant names (rownames of Config matrices) be shown on the plot? Default is FALSE.

Value

a ggplot heatmap style plot showing the differences between the two Config matrices, specifically the differences Config1 - Config2.

Examples

```
Config1 <- matrix(c(
    rep(0, 15), rep(1, 8), rep(0, 7),
    rep(1, 5), rep(0, 3), rep(1, 7)
), ncol = 3)
Config2 <- matrix(c(
    rep(0, 15), rep(1, 8), rep(1, 7),
    rep(0, 5), rep(1, 3), rep(1, 7)
```

```
), ncol = 3)
rownames(Config1) <- rownames(Config2) <- paste0("var", 1:nrow(Config1))
colnames(Config1) <- colnames(Config2) <- paste0("clone", 1:ncol(Config1))
plot_config_diffs(Config1, Config2)
```

`plot_tree` *Plot a phylogenetic tree*

Description

Plot a phylogenetic tree

Usage

```
plot_tree(tree, orient = "h")
```

Arguments

`tree`	A phylgenetic tee object of class "phylo"
`orient`	A string for the orientation of the tree: "v" (vertical) or "h" (horizontal)

Details

This function plots a phylogenetic tree from an object of class "phylo", as produced, for example, by the Canopy package.

Value

a ggtree object

Author(s)

Davis McCarthy and Yuanhua Huang

References

This function makes use of the `ggtree` package:

Guangchuang Yu, David Smith, Huachen Zhu, Yi Guan, Tommy Tsan-Yuk Lam. ggtree: an R package for visualization and annotation of phylogenetic trees with their covariates and other associated data. Methods in Ecology and Evolution 2017, 8(1):28-36, doi:10.1111/2041-210X.12628

Examples

```
data(example_donor)
plot_tree(tree, orient = "v")
```

predMixBinom — *Predicted probability from learned binomial mixture model*

Description

Predicted probability from learned binomial mixture model

Usage

```
predMixBinom(k, n, p, psi)
```

Arguments

k	A vector of integers. number of success
n	A vector of integers. number of trials
p	a vector of binomial success probabilities
psi	A float between 0 and 1. fraction of each component

Value

A list with two components: prob, a matrix representing the probability of each of the passed values coming from each component of the mixture and logLik, the total log-likelihood of the new samples.

Examples

```
n1 <- array(sample(1:30, 50, replace = TRUE))
n2 <- array(sample(1:30, 200, replace = TRUE))
k1 <- apply(n1, 1, rbinom, n = 1, p = 0.5)
k2 <- apply(n2, 1, rbinom, n = 1, p = 0.01)
RV <- mixBinom(c(k1, k2), c(n1, n2))
RV_pred <- predMixBinom(3, 10, RV$p, RV$psi)
```

prob_heatmap — *Plot a heatmap for probability of clone assignment*

Description

Plot a heatmap for probability of clone assignment

Usage

```
prob_heatmap(prob_mat, threshold = 0.5, mode = "best", cell_idx = NULL)
```

Arguments

prob_mat	A matrix (M x K), the probability of cell j to clone k
threshold	A float value, the threshold for assignable cells
mode	A string, the method for defining scores for filtering cells: best and delta. best: highest probability of a cell to K clones, delta: the difference between the best and second.
cell_idx	A vector the indices of the input cells. If NULL, order by the probability of each clone

Value

a ggplot object

Examples

```
data(example_donor)
assignments <- clone_id(A_clone, D_clone, Config = tree$Z, inference = "EM")
fig <- prob_heatmap(assignments$prob)
```

---

pub.theme *Define a publication-style plot theme*

Description

Define a publication-style plot theme

Usage

```
pub.theme(size = 12)
```

Arguments

size	numeric, base font size for adapted ggplot2 theme

Details

This theme modifies the theme_classic theme in ggplot2.

Value

a ggplot theme based on theme_classic

Examples

```
library(ggplot2)
x <- sample(10)
y <- x + runif(10) - 0.5
df <- data.frame(x = x, y = y)
fig <- ggplot(df, aes(x = x, y = y)) +
    geom_point() +
    pub.theme()
```

---

read_vcf *Read a VCF file into R session*

Description

Read a VCF file into R session

Usage

```
read_vcf(
  vcf_file,
  genome = "GRCh37",
  seq_levels_style = "Ensembl",
  verbose = TRUE
)
```

Arguments

vcf_file	character(1), path to VCF file to read into R session as a CollapsedVCF object
genome	character(1), string indicating the genome build used in the VCF file(s) (default: "GRCh37")
seq_levels_style	character(1), string passed to seqlevelsStyle the style to use for chromosome/contig names (default: "Ensembl")
verbose	logical(1), should messages be printed as function runs?

Value

a vcf object

Examples

```
vcf <- read_vcf(system.file("extdata", "cells.donorid.vcf.gz",
                package = "cardelino"))
```

rowArgmax — *Column index of the maximum value for each row in a matrix*

Description

Column index of the maximum value for each row in a matrix

Usage

```
rowArgmax(X)
```

Arguments

X	A matrix of floats.

Value

a vector of the index of column for each row. Note, when multiple columns have the same value, only the earliest column will be returned.

Examples

```
matA <- matrix(sample(seq(12)), nrow = 3)
rowArgmax(matA)
```

rowMax — *Maximum value for each row in a matrix*

Description

Maximum value for each row in a matrix

Usage

```
rowMax(X, mode = "best")
```

Arguments

X	A matrix of floats.
mode	A string, the method for defining scores for filtering cells: best, second and delta. `best`: highest value for each row, similarly for the `second`. `delta` is the difference between the best and the second.

Value

a vector of the collapsed value for each row, depending on the mode used.

Examples

```
matA <- matrix(sample(seq(12)), nrow = 3)
rowMax(matA)
```

`sample_seq_depth` *Update matrix D with manually selected missing rate*

Description

Given missing rate, the NA will be generated first. For none NA element, sequencing depth with uniformly sampled from D, row wisely. Namely, the depth is variant specific.

Usage

```
sample_seq_depth(D, n_cells = NULL, n_sites = NULL, missing_rate = NULL)
```

Arguments

`D`	A matrix (N variants x M cells), the original sequencing coverage, NA means missing
`n_cells`	A integer, the number of the cells to generate
`n_sites`	A integer, the number of variants to generate
`missing_rate`	A float value, if NULL, use the same missing rate as D

Value

a n_sites by n_cells matrix sampled from input D.

Examples

```
data(simulation_input)
D1 <- sample_seq_depth(D_input,
    n_cells = 500, n_sites = 50,
    missing_rate = 0.85
)
```

sample_tree_SNV *Down sample number of SNVs in the tree*

Description

Down sample number of SNVs in the tree

Usage

```
sample_tree_SNV(tree, n_SNV = NULL)
```

Arguments

tree	A tree object from Canopy
n_SNV	A integer, the number of SNVs to keep in the output tree

Value

a phylo tree with down sampled variants

Examples

```
data(simulation_input)
tree_lite <- sample_tree_SNV(tree_4clone, n_SNV = 10)
```

sim_read_count *Synthetic reads generator for genetic variants*

Description

There are following steps to generate the simulated reads counts for variants in single cells: 1) given the clonal genotype and the clonal prevalence, the genotypes (i.e, the clone) of cells will be generated following a multinomial distribution. Note, one cell may contain variants from two clones when it is a doublet. 2) given the distribution of reads coverage, e.g., a matrix of read coverage from real data, (variant specific), the total reads of each variant will be generated by random sampling. Note, the missing rate is governed by this matrix. 3) the allelic frequency of each variant will be generated by following a beta distribution with parameters of mean and variance. 4) Given the genotype of a cell, if the mutation exists in a cell, the alteration read counts will be generated by a binomial distribution, parameterized the allelic frequency, sampled from step 3. 5) Given the genotype of a cell, if the mutation does not exist in a cell, the alteration read counts will be generated by a binomial distribution, parameterized by the technical error rate.

Usage

```
sim_read_count(
  Config,
  D,
  Psi = NULL,
  means = c(0.002, 0.45),
  vars = c(100, 1),
  wise0 = "element",
  wise1 = "variant",
  cell_num = 300,
  permute_D = FALSE,
  sample_cell = TRUE,
  doublet = 0
)
```

Arguments

`Config`	A matrix of binary values. The clone-variant configuration, which encodes the phylogenetic tree structure, and the genotype of each clone
`D`	A matrix of integers. Sequencing depth for N variants across x cells (ideally >100 cells). NA means 0 here.
`Psi`	A vector of float. The fractions of each clone. If NULL, set a uniform distribution.
`means`	A vector of two floats. The mean theta_1 (false positive rate) and the mean theta_2 (true positive rate).
`vars`	A vector of two floats. The variance of theta_1 and theta_2.
`wise0`	A string, the beta-binomial parameter specificity for theta0: global, variant, element.
`wise1`	A string, the beta-binomial parameter specificity for theta1: global, variant, element.
`cell_num`	A integer. The number of cells to generate.
`permute_D`	A Boolean value. If True permute variants in D.
`sample_cell`	A Boolean value. If True and M > ncol(D), sample cells.
`doublet`	A float between 0 and 1, the rate of doublets

Value

a list containing `A_sim`, a matrix for alteration reads, `A_sim`, a matrix for total reads, `I_sim`, a matrix for clonal label, `H_sim`, a matrix for genotype, `theta0`, a matrix of expected false positive rate, `theta1`, a matrix of expected true positive rate, `theta0_binom`, theta0 as binomial parameter, `theta1_binom`, theta0 as binomial parameter, and `is_doublet`, a vector of Boolean value if a cell is a doublet

Examples

```
data(simulation_input)
D2 <- sample_seq_depth(D_input, n_cells = 500, n_sites = nrow(tree_4clone$Z))
simu <- sim_read_count(tree_4clone$Z, D2, Psi = NULL, cell_num = 500)
```

`tree` *A tree object*

Description

This tree object contains clonal tree information, inferred from bulk exome-seq data

Usage

```
example_donor
```

Format

a tree object

Value

NULL, but makes available a tree object

Author(s)

Yuanhua Huang, Davis McCarthy, 2018-06-25

Source

A fibroblast sample from HipSci project

`tree_3clone` *A tree object*

Description

This tree object with 3 clones contains clonal tree information, inferred from bulk exome-seq data

Usage

```
simulation_input
```

Format

a tree object

Value

NULL, but makes available a tree object

Author(s)

Yuanhua Huang, Davis McCarthy, 2018-06-25

Source

A fibroblast sample from HipSci project

---

tree_4clone *A tree object*

Description

This tree object with 4 clones contains clonal tree information, inferred from bulk exome-seq data

Usage

```
simulation_input
```

Format

a tree object

Value

NULL, but makes available a tree object

Author(s)

Yuanhua Huang, Davis McCarthy, 2018-06-25

Source

A fibroblast sample from HipSci project

tree_5clone — *A tree object*

Description

This tree object with 5 clones contains clonal tree information, inferred from bulk exome-seq data

Usage

```
simulation_input
```

Format

a tree object

Value

NULL, but makes available a tree object

Author(s)

Yuanhua Huang, Davis McCarthy, 2018-06-25

Source

A fibroblast sample from HipSci project

vc_heatmap — *Plot a variant-cell heatmap for cell clonal assignment*

Description

Plot a variant-cell heatmap for cell clonal assignment

Usage

```
vc_heatmap(mat, prob, Config, show_legend = FALSE)
```

Arguments

mat	A matrix for heatmap: N variants x M cells. row and column will be sorted automatically.
prob	A matrix of probability of clonal assignment: M cells x K clones
Config	A binary matrix of clonal Configuration: N variants x K clones
show_legend	A bool value: if TRUE, show the legend

Value

a pheatmap object

a ggplot object

References

This function makes use of the pheatmap packages

Examples

```
data(example_donor)
assignments <- clone_id(A_clone, D_clone, Config = tree$Z)
fig <- vc_heatmap(assignments$prob_variant, assignments$prob, tree$Z)
```

Index