Package ‘sccomp’

October 17, 2024

Title Robust Outlier-aware Estimation of Composition and Heterogeneity
for Single-cell Data

Version 1.8.0

Description A robust and outlier-aware method for testing differential tissue composition from single-cell data. This model can infer changes in tissue composition and heterogeneity, and can produce realistic data simulations based on any existing dataset. This model can also transfer knowledge from a large set of integrated datasets to increase accuracy further.

License GPL-3

Encoding UTF-8

LazyData true

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.1

Biarch true

Depends R (>= 4.2.0)

Imports methods, Rcpp (>= 0.12.0), RcppParallel (>= 5.0.1), rstantools
(>= 2.1.1), rstan (>= 2.26.0), SeuratObject,
SingleCellExperiment, parallel, dplyr, tidyr, purrr, magrittr,
rlang, tibble, boot, lifecycle, stats, tidyselect, utils,
ggplot2, ggrepel, patchwork, forcats, readr, scales, stringr,
glue

Suggests BiocStyle, testthat (>= 3.0.0), markdown, knitr, loo,
tidyseurat, tidySingleCellExperiment, prettydoc

Enhances furrr, extraDistr

LinkingTo BH (>= 1.66.0), Rcpp (>= 0.12.0), RcppEigen (>= 0.3.3.3.0),
RcppParallel (>= 5.0.1), rstan (>= 2.26.0), StanHeaders (>=
2.26.0)

SystemRequirements GNU make

VignetteBuilder knitr

RdMacros lifecycle

biocViews ImmunoOncology, Normalization, Sequencing, RNASeq, Software,
GeneExpression, Transcriptomics, SingleCell, Clustering

LazyDataCompression xz

Config/testthat/edition 3

URL https://github.com/stemangiola/sccomp

BugReports https://github.com/stemangiola/sccomp/issues

Additional_repositories https://mc-stan.org/r-packages/

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

git_branch RELEASE_3_19

git_last_commit dcbda58

git_last_commit_date 2024-04-30

Repository Bioconductor 3.19

Date/Publication 2024-10-16

Author Stefano Mangiola [aut, cre]

Maintainer Stefano Mangiola <mangiolastefano@gmail.com>

Contents

sccomp-package *The 'sccomp' package.*

Description

A DESCRIPTION OF THE PACKAGE

Author(s)

Maintainer: Stefano Mangiola <mangiolastefano@gmail.com>

References

Stan Development Team (2020). RStan: the R interface to Stan. R package version 2.21.2. https://mc-stan.org

See Also

Useful links:

- https://github.com/stemangiola/sccomp
- Report bugs at https://github.com/stemangiola/sccomp/issues

counts_obj *counts_obj*

Description

Example data set containing cell counts per cell cluster

Usage

```
data(counts_obj)
```

Format

A tidy data frame.

glm_dirichlet_multinomial

glm_dirichlet_multinomial

Description

This object is mostly for internal use and comparative purposes, if the dirichlet_multinomial is chosen as noise model.

Usage

```
data(glm_dirichlet_multinomial)
```

Format

A text file containing stan code for the Dirichlet model.

glm_dirichlet_multinomial_generate_quantities

glm_dirichlet_multinomial_generate_quantities

Description

This object is mostly for internal use and comparative purposes, if the dirichlet_multinomial is chosen as noise model.

Usage

```
data(glm_dirichlet_multinomial_generate_quantities)
```

Format

A text file containing stan code for the Dirichlet model.

`glm_dirichlet_multinomial_imputation` *glm_dirichlet_multinomial_imputation*

Description

This object is mostly for internal use and comparative purposes, if the dirichlet_multinomial is chosen as noise model.

Usage

```
data(glm_dirichlet_multinomial_imputation)
```

Format

A text file containing stan code for the Dirichlet model.

`glm_multi_beta` *glm_multi_beta*

Description

This object is mostly for internal use and comparative purposes, if the multi_beta is chosen as noise model.

Usage

```
data(glm_multi_beta)
```

Format

A text file containing stan code for the Beta only model.

`glm_multi_beta_generate_data` *glm_multi_beta_generate_data*

Description

This object is mostly for internal use and comparative purposes, if the multi_beta is chosen as noise model.

Usage

```
data(glm_multi_beta_generate_data)
```

Format

A text file containing stan code for the Beta only model.

`multipanel_theme` *multipanel_theme*

Description

this object includes the ggplot, theme

Usage

```
data(multipanel_theme)
```

Format

A ggplot theme

multi_beta_glm *multi_beta_glm main*

Description

This function runs the data modelling and statistical test for the hypothesis that a cell_type includes outlier biological replicate.

Usage

```
multi_beta_glm(
  .data,
  formula = ~1,
  .sample,
  check_outliers = FALSE,
  approximate_posterior_inference = TRUE,
  cores = detect_cores(),
  seed = sample(1e+05, 1)
)
```

Arguments

`.data`	A tibble including a cell_type name column | sample name column | read counts column | factor columns | Pvaue column | a significance column
`formula`	A formula. The sample formula used to perform the differential cell_type abundance analysis
`.sample`	A column name as symbol. The sample identifier
`check_outliers`	A boolean. Whether to check for outliers before the fit.
`approximate_posterior_inference`	A boolean. Whether the inference of the joint posterior distribution should be approximated with variational Bayes. It confers execution time advantage.
`cores`	An integer. How many cored to be used with parallel calculations.
`seed`	An integer. Used for development and testing purposes

Value

A nested tibble `tbl` with cell_type-wise information: `sample wise data` | plot | `ppc samples failed` | `exposure deleterious outliers`

plot.sccomp_tbl *plot*

Description

This function plots a summary of the results of the model.

Usage

```
## S3 method for class 'sccomp_tbl'
plot(x, ...)
```

Arguments

x	A tibble including a cell_group name column | sample name column | read counts column | factor columns | Pvalue column | a significance column
...	parameters like significance_threshold A real. FDR threshold for labelling significant cell-groups.

Value

A ggplot

Examples

```
data("counts_obj")

estimate =
  sccomp_estimate(
  counts_obj ,
   ~ type, ~1, sample, cell_group, count,
    cores = 1
  )

# estimate |> plot()
```

plot_summary *plot_summary*

Description

This function plots a summary of the results of the model.

Usage

```
plot_summary(.data, significance_threshold = 0.025)
```

Arguments

`.data` A tibble including a cell_group name column | sample name column | read counts column | factor columns | Pvalue column | a significance column

`significance_threshold`
A real. FDR threshold for labelling significant cell-groups.

Value

A `ggplot`

Examples

```
data("counts_obj")

estimate =
  sccomp_estimate(
  counts_obj ,
   ~ type, ~1, sample, cell_group, count,
    approximate_posterior_inference = "all",
    cores = 1
  )

# estimate |> plot_summary()
```

`sccomp_boxplot` *sccomp_boxplot*

Description

This function plots a boxplot of the results of the model.

Usage

```
sccomp_boxplot(.data, factor, significance_threshold = 0.025)
```

Arguments

`.data` A tibble including a cell_group name column | sample name column | read counts column | factor columns | Pvalue column | a significance column

`factor` A character string for a factor of interest included in the model

`significance_threshold`
A real. FDR threshold for labelling significant cell-groups.

Value

A `ggplot`

Examples

```
data("counts_obj")

estimate =
  sccomp_estimate(
  counts_obj ,
   ~ type, ~1, sample, cell_group, count,
    cores = 1
  ) |>
  sccomp_test()

# estimate |> sccomp_boxplot()
```

sccomp_estimate *Main Function for SCCOMP Estimate*

Description

The sccomp_estimate function performs linear modeling on a table of cell counts, which includes a cell-group identifier, sample identifier, integer count, and factors (continuous or discrete). The user can define a linear model with an input R formula, where the first factor is the factor of interest. Alternatively, sccomp accepts single-cell data containers (e.g., Seurat, SingleCellExperiment, cell metadata, or group-size) and derives the count data from cell metadata.

Usage

```
sccomp_estimate(
  .data,
  formula_composition = ~1,
  formula_variability = ~1,
  .sample,
  .cell_group,
  .count = NULL,
  cores = detectCores(),
  bimodal_mean_variability_association = FALSE,
  percent_false_positive = 5,
  variational_inference = TRUE,
  prior_mean = list(intercept = c(0, 1), coefficients = c(0, 1)),
 prior_overdispersion_mean_association = list(intercept = c(5, 2), slope = c(0, 0.6),
    standard_deviation = c(10, 20)),
  .sample_cell_group_pairs_to_exclude = NULL,
  verbose = TRUE,
  enable_loo = FALSE,
  noise_model = "multi_beta_binomial",
  exclude_priors = FALSE,
  use_data = TRUE,
  mcmc_seed = sample(1e+05, 1),
  max_sampling_iterations = 20000,
  pass_fit = TRUE,
  approximate_posterior_inference = NULL
)
```

Arguments

`.data` A tibble including cell_group name column, sample name column, read counts column (optional depending on the input class), and factor columns.

`formula_composition`
A formula describing the model for differential abundance.

`formula_variability`
A formula describing the model for differential variability.

`.sample` A column name as symbol for the sample identifier.

`.cell_group` A column name as symbol for the cell_group identifier.

`.count` A column name as symbol for the cell_group abundance (read count).

`cores` Number of cores to use for parallel calculations.

`bimodal_mean_variability_association`
Boolean for modeling mean-variability as bimodal.

`percent_false_positive`
Real number between 0 and 100 for outlier identification.

`variational_inference`
Boolean for using variational Bayes for posterior inference. It is faster and convenient. Setting this argument to FALSE runs the full Bayesian (Hamiltonian Monte Carlo) inference, slower but it is the gold standard.

`prior_mean` List with prior knowledge about mean distribution, they are the intercept and coefficient.

`prior_overdispersion_mean_association`
List with prior knowledge about mean/variability association.

`.sample_cell_group_pairs_to_exclude`
Column name with boolean for sample/cell-group pairs exclusion.

`verbose` Boolean to print progression.

`enable_loo` Boolean to enable model comparison using the LOO package.

`noise_model` Character string for the noise model (e.g., 'multi_beta_binomial').

`exclude_priors` Boolean to run a prior-free model.

`use_data` Boolean to run the model data-free.

`mcmc_seed` Integer for MCMC reproducibility.

`max_sampling_iterations`
Integer to limit maximum iterations for large datasets.

`pass_fit` Boolean to include the Stan fit as attribute in the output.

`approximate_posterior_inference`
DEPRECATED please use the `variational_inference` argument.

Value

A nested tibble `tbl`, with the following columns

- cell_group - column including the cell groups being tested
- parameter - The parameter being estimated, from the design matrix dscribed with the input formula_composition and formula_variability
- factor - The factor in the formula corresponding to the covariate, if exists (e.g. it does not exist in case og Intercept or contrasts, which usually are combination of parameters)
- c_lower - lower (2.5%) quantile of the posterior distribution for a composition (c) parameter.
- c_effect - mean of the posterior distribution for a composition (c) parameter.
- c_upper - upper (97.5%) quantile of the posterior distribution fo a composition (c) parameter.
- c_pH0 - Probability of the null hypothesis (no difference) for a composition (c). This is not a p-value.
- c_FDR - False-discovery rate of the null hypothesis (no difference) for a composition (c).
- c_n_eff - Effective sample size - the number of independent draws in the sample, the higher the better (mc-stan.org/docs/2_25/cmdstan-guide/stansummary.html).
- c_R_k_hat - R statistic, a measure of chain equilibrium, should be within 0.05 of 1.0 (mc-stan.org/docs/2_25/cmdstan-guide/stansummary.html).
- v_lower - Lower (2.5%) quantile of the posterior distribution for a variability (v) parameter
- v_effect - Mean of the posterior distribution for a variability (v) parameter
- v_upper - Upper (97.5%) quantile of the posterior distribution for a variability (v) parameter
- v_pH0 - Probability of the null hypothesis (no difference) for a variability (v). This is not a p-value.
- v_FDR - False-discovery rate of the null hypothesis (no difference), for a variability (v).
- v_n_eff - Effective sample size for a variability (v) parameter - the number of independent draws in the sample, the higher the better (mc-stan.org/docs/2_25/cmdstan-guide/stansummary.html).
- v_R_k_hat - R statistic for a variability (v) parameter, a measure of chain equilibrium, should be within 0.05 of 1.0 (mc-stan.org/docs/2_25/cmdstan-guide/stansummary.html).
- count_data Nested input count data.

Examples

```
data("counts_obj")

estimate =
  sccomp_estimate(
  counts_obj ,
   ~ type,
   ~1,
   sample,
   cell_group,
   count,
    cores = 1
  )
```

sccomp_glm *DEPRECATED - sccomp_glm main*

Description

The function for linear modelling takes as input a table of cell counts with three columns containing a cell-group identifier, sample identifier, integer count and the factors (continuous or discrete). The user can define a linear model with an input R formula, where the first factor is the factor of interest. Alternatively, sccomp accepts single-cell data containers (Seurat, SingleCellExperiment44, cell metadata or group-size). In this case, sccomp derives the count data from cell metadata.

Usage

```
sccomp_glm(
  .data,
  formula_composition = ~1,
  formula_variability = ~1,
  .sample,
  .cell_group,
  .count = NULL,
  contrasts = NULL,
  prior_mean_variable_association = list(intercept = c(5, 2), slope = c(0, 0.6),
    standard_deviation = c(20, 40)),
  check_outliers = TRUE,
  bimodal_mean_variability_association = FALSE,
  enable_loo = FALSE,
  cores = detectCores(),
  percent_false_positive = 5,
  approximate_posterior_inference = "none",
  test_composition_above_logit_fold_change = 0.2,
  .sample_cell_group_pairs_to_exclude = NULL,
  verbose = FALSE,
  noise_model = "multi_beta_binomial",
  exclude_priors = FALSE,
  use_data = TRUE,
  mcmc_seed = sample(1e+05, 1),
  max_sampling_iterations = 20000,
  pass_fit = TRUE
)
```

Arguments

`.data` A tibble including a cell_group name column | sample name column | read counts column (optional depending on the input class) | factor columns.

`formula_composition`
A formula. The formula describing the model for differential abundance, for example ~treatment.

`formula_variability`
A formula. The formula describing the model for differential variability, for example ~treatment. In most cases, if differentially variability is of interest, the formula should only include the factor of interest as a large anount of data is needed to define variability depending to each factors.

`.sample` A column name as symbol. The sample identifier

`.cell_group` A column name as symbol. The cell_group identifier

`.count` A column name as symbol. The cell_group abundance (read count). Used only for data frame count output. The variable in this column should be of class integer.

`contrasts` A vector of character strings. For example if your formula is `~ 0 + treatment` and the factor treatment has values `yes` and `no`, your contrast could be constrasts = c("treatmentyes - treatmentno").

`prior_mean_variable_association`
A list of the form list(intercept = c(5, 2), slope = c(0, 0.6), standard_deviation = c(20, 40)). Where for intercept and slope parameters, we specify mean and standard deviation, while for standard deviation, we specify shape and rate. This is used to incorporate prior knowledge about the mean/variability association of cell-type proportions.

`check_outliers` A boolean. Whether to check for outliers before the fit.

`bimodal_mean_variability_association`
A boolean. Whether to model the mean-variability as bimodal, as often needed in the case of single-cell RNA sequencing data, and not usually for CyTOF and microbiome data. The plot summary_plot()$credible_intervals_2D can be used to assess whether the bimodality should be modelled.

`enable_loo` A boolean. Enable model comparison by the R package LOO. This is helpful when you want to compare the fit between two models, for example, analogously to ANOVA, between a one factor model versus a interceot-only model.

`cores` An integer. How many cored to be used with parallel calculations.

`percent_false_positive`
A real between 0 and 100 non included. This used to identify outliers with a specific false positive rate.

`approximate_posterior_inference`
A boolean. Whether the inference of the joint posterior distribution should be approximated with variational Bayes. It confers execution time advantage.

`test_composition_above_logit_fold_change`
A positive integer. It is the effect threshold used for the hypothesis test. A value of 0.2 correspond to a change in cell proportion of 10% for a cell type with baseline proportion of 50%. That is, a cell type goes from 45% to 50%. When the baseline proportion is closer to 0 or 1 this effect thrshold has consistent value in the logit uncontrained scale.

`.sample_cell_group_pairs_to_exclude`
A column name that includes a boolean variable for the sample/cell-group pairs to be ignored in the fit. This argument is for pro-users.

`verbose` A boolean. Prints progression.

`noise_model` A character string. The two noise models available are multi_beta_binomial (default) and dirichlet_multinomial.

`exclude_priors` A boolean. Whether to run a prior-free model, for benchmarking purposes.

`use_data` A booelan. Whether to sun the model data free. This can be used for prior predictive check.

`mcmc_seed` An integer. Used for Markov-chain Monte Carlo reproducibility. By default a random number is sampled from 1 to 999999. This itself can be controlled by set.seed()

`max_sampling_iterations`
An integer. This limit the maximum number of iterations in case a large dataset is used, for limiting the computation time.

`pass_fit` A boolean. Whether to pass the Stan fit as attribute in the output. Because the Stan fit can be very large, setting this to FALSE can be used to lower the memory imprint to save the output.

Value

A nested tibble `tbl`, with the following columns

- cell_group - column including the cell groups being tested
- parameter - The parameter being estimated, from the design matrix dscribed with the input formula_composition and formula_variability
- factor - The factor in the formula corresponding to the covariate, if exists (e.g. it does not exist in case og Intercept or contrasts, which usually are combination of parameters)
- c_lower - lower (2.5%) quantile of the posterior distribution for a composition (c) parameter.
- c_effect - mean of the posterior distribution for a composition (c) parameter.
- c_upper - upper (97.5%) quantile of the posterior distribution fo a composition (c) parameter.
- c_pH0 - Probability of the null hypothesis (no difference) for a composition (c). This is not a p-value.
- c_FDR - False-discovery rate of the null hypothesis (no difference) for a composition (c).
- c_n_eff - Effective sample size - the number of independent draws in the sample, the higher the better (mc-stan.org/docs/2_25/cmdstan-guide/stansummary.html).
- c_R_k_hat - R statistic, a measure of chain equilibrium, should be within 0.05 of 1.0 (mc-stan.org/docs/2_25/cmdstan-guide/stansummary.html).
- v_lower - Lower (2.5%) quantile of the posterior distribution for a variability (v) parameter
- v_effect - Mean of the posterior distribution for a variability (v) parameter
- v_upper - Upper (97.5%) quantile of the posterior distribution for a variability (v) parameter
- v_pH0 - Probability of the null hypothesis (no difference) for a variability (v). This is not a p-value.
- v_FDR - False-discovery rate of the null hypothesis (no difference), for a variability (v).
- v_n_eff - Effective sample size for a variability (v) parameter - the number of independent draws in the sample, the higher the better (mc-stan.org/docs/2_25/cmdstan-guide/stansummary.html).
- v_R_k_hat - R statistic for a variability (v) parameter, a measure of chain equilibrium, should be within 0.05 of 1.0 (mc-stan.org/docs/2_25/cmdstan-guide/stansummary.html).
- count_data Nested input count data.

Examples

```
data("counts_obj")

estimate =
  sccomp_glm(
  counts_obj ,
   ~ type,
   ~1,
   sample,
   cell_group,
   count,
    check_outliers = FALSE,
    cores = 1
  )
```

sccomp_predict *sccomp_predict*

Description

This function replicates counts from a real-world dataset.

Usage

```
sccomp_predict(
  fit,
  formula_composition = NULL,
  new_data = NULL,
  number_of_draws = 500,
  mcmc_seed = sample(1e+05, 1)
)
```

Arguments

`fit` The result of sccomp_estimate.

`formula_composition` A formula. The formula describing the model for differential abundance, for example ~treatment. This formula can be a sub-formula of your estimated model; in this case all other factor will be factored out.

`new_data` A sample-wise data frame including the column that represent the factors in your formula. If you want to predict proportions for 10 samples, there should be 10 rows. T

`number_of_draws` An integer. How may copies of the data you want to draw from the model joint posterior distribution.

`mcmc_seed` An integer. Used for Markov-chain Monte Carlo reproducibility. By default a random number is sampled from 1 to 999999. This itself can be controlled by set.seed()

Value

A nested tibble `tbl` with cell_group-wise statistics

Examples

```
data("counts_obj")

if(.Platform$OS.type == "unix")
  sccomp_estimate(
  counts_obj ,
   ~ type, ~1,  sample, cell_group, count,
    cores = 1
  ) |>

  sccomp_predict()
```

`sccomp_remove_outliers`

sccomp_remove_outliers main

Description

The function for linear modelling takes as input a table of cell counts with three columns containing a cell-group identifier, sample identifier, integer count and the factors (continuous or discrete). The user can define a linear model with an input R formula, where the first factor is the factor of interest. Alternatively, sccomp accepts single-cell data containers (Seurat, SingleCellExperiment44, cell metadata or group-size). In this case, sccomp derives the count data from cell metadata.

Usage

```
sccomp_remove_outliers(
  .estimate,
  percent_false_positive = 5,
  cores = detectCores(),
  variational_inference = TRUE,
  verbose = TRUE,
  mcmc_seed = sample(1e+05, 1),
  max_sampling_iterations = 20000,
  enable_loo = FALSE,
  approximate_posterior_inference = NULL
)
```

Arguments

`.estimate` A tibble including a cell_group name column | sample name column | read counts column (optional depending on the input class) | factor columns.

`percent_false_positive`
: A real between 0 and 100 non included. This used to identify outliers with a specific false positive rate.

`cores`
: An integer. How many cored to be used with parallel calculations.

`variational_inference`
: Boolean for using variational Bayes for posterior inference. It is faster and convenient. Setting this argument to FALSE runs the full Bayesian (Hamiltonian Monte Carlo) inference, slower but it is the gold standard.

`verbose`
: A boolean. Prints progression.

`mcmc_seed`
: An integer. Used for Markov-chain Monte Carlo reproducibility. By default a random number is sampled from 1 to 999999. This itself can be controlled by set.seed()

`max_sampling_iterations`
: An integer. This limit the maximum number of iterations in case a large dataset is used, for limiting the computation time.

`enable_loo`
: A boolean. Enable model comparison by the R package LOO. This is helpful when you want to compare the fit between two models, for example, analogously to ANOVA, between a one factor model versus a interceot-only model.

`approximate_posterior_inference`
: DEPRECATED please use the `variational_inference` argument.

Value

A nested tibble `tbl`, with the following columns

- cell_group - column including the cell groups being tested
- parameter - The parameter being estimated, from the design matrix dscribed with the input formula_composition and formula_variability
- factor - The factor in the formula corresponding to the covariate, if exists (e.g. it does not exist in case og Intercept or contrasts, which usually are combination of parameters)
- c_lower - lower (2.5%) quantile of the posterior distribution for a composition (c) parameter.
- c_effect - mean of the posterior distribution for a composition (c) parameter.
- c_upper - upper (97.5%) quantile of the posterior distribution fo a composition (c) parameter.
- c_n_eff - Effective sample size - the number of independent draws in the sample, the higher the better (mc-stan.org/docs/2_25/cmdstan-guide/stansummary.html).
- c_R_k_hat - R statistic, a measure of chain equilibrium, should be within 0.05 of 1.0 (mc-stan.org/docs/2_25/cmdstan-guide/stansummary.html).
- v_lower - Lower (2.5%) quantile of the posterior distribution for a variability (v) parameter
- v_effect - Mean of the posterior distribution for a variability (v) parameter
- v_upper - Upper (97.5%) quantile of the posterior distribution for a variability (v) parameter
- v_n_eff - Effective sample size for a variability (v) parameter - the number of independent draws in the sample, the higher the better (mc-stan.org/docs/2_25/cmdstan-guide/stansummary.html).
- v_R_k_hat - R statistic for a variability (v) parameter, a measure of chain equilibrium, should be within 0.05 of 1.0 (mc-stan.org/docs/2_25/cmdstan-guide/stansummary.html).
- count_data Nested input count data.

Examples

```
data("counts_obj")

estimate =
  sccomp_estimate(
  counts_obj ,
   ~ type,
   ~1,
   sample,
   cell_group,
   count,
    cores = 1
  ) |>
  sccomp_remove_outliers(cores = 1)
```

---

sccomp_remove_unwanted_variation

sccomp_remove_unwanted_variation

Description

This function uses the model to remove unwanted variation from a dataset using the estimated of the model. For example if you fit your data with this formula ~ `factor_1` + `factor_2` and use this formula to remove unwanted variation ~ `factor_1`, the `factor_2` will be factored out.

Usage

```
sccomp_remove_unwanted_variation(
  .data,
  formula_composition = ~1,
  formula_variability = NULL
)
```

Arguments

`.data` A tibble. The result of sccomp_estimate.

`formula_composition`
A formula. The formula describing the model for differential abundance, for example ~treatment. This formula can be a sub-formula of your estimated model; in this case all other factor will be factored out.

`formula_variability`
A formula. The formula describing the model for differential variability, for example ~treatment. In most cases, if differentially variability is of interest, the formula should only include the factor of interest as a large anount of data is needed to define variability depending to each factors. This formula can be a sub-formula of your estimated model; in this case all other factor will be factored out.

Value

A nested tibble `tbl` with cell_group-wise statistics

Examples

```
data("counts_obj")

  estimates = sccomp_estimate(
  counts_obj ,
   ~ type, ~1,  sample, cell_group, count,
    cores = 1
  )

  sccomp_remove_unwanted_variation(estimates)
```

`sccomp_replicate` *sccomp_replicate*

Description

This function replicates counts from a real-world dataset.

Usage

```
sccomp_replicate(
  fit,
  formula_composition = NULL,
  formula_variability = NULL,
  number_of_draws = 1,
  mcmc_seed = sample(1e+05, 1)
)
```

Arguments

`fit` The result of sccomp_estimate.

`formula_composition`
A formula. The formula describing the model for differential abundance, for example ~treatment. This formula can be a sub-formula of your estimated model; in this case all other factor will be factored out.

`formula_variability`
A formula. The formula describing the model for differential variability, for example ~treatment. In most cases, if differentially variability is of interest, the formula should only include the factor of interest as a large anount of data is needed to define variability depending to each factors. This formula can be a sub-formula of your estimated model; in this case all other factor will be factored out.

`number_of_draws`
An integer. How may copies of the data you want to draw from the model joint posterior distribution.

`mcmc_seed` An integer. Used for Markov-chain Monte Carlo reproducibility. By default a random number is sampled from 1 to 999999. This itself can be controlled by set.seed()

Value

A nested tibble `tbl` with cell_group-wise statistics

Examples

```
data("counts_obj")

if(.Platform$OS.type == "unix")
  sccomp_estimate(
  counts_obj ,
   ~ type, ~1,  sample, cell_group, count,
    cores = 1
  ) |>

  sccomp_replicate()
```

`sccomp_test` *sccomp_test*

Description

This function test contrasts from a sccomp result.

Usage

```
sccomp_test(
  .data,
  contrasts = NULL,
  percent_false_positive = 5,
  test_composition_above_logit_fold_change = 0.2,
  pass_fit = TRUE
)
```

Arguments

`.data` A tibble. The result of sccomp_estimate.

`contrasts` A vector of character strings. For example if your formula is `~ 0 + treatment` and the factor treatment has values `yes` and `no`, your contrast could be "contrasts = c(treatmentyes - treatmentno)".

`percent_false_positive`
A real between 0 and 100 non included. This used to identify outliers with a specific false positive rate.

`test_composition_above_logit_fold_change`
A positive integer. It is the effect threshold used for the hypothesis test. A value of 0.2 correspond to a change in cell proportion of 10% for a cell type with baseline proportion of 50%. That is, a cell type goes from 45% to 50%. When the baseline proportion is closer to 0 or 1 this effect thrshold has consistent value in the logit uncontrained scale.

`pass_fit` A boolean. Whether to pass the Stan fit as attribute in the output. Because the Stan fit can be very large, setting this to FALSE can be used to lower the memory imprint to save the output.

Value

A nested tibble `tbl` with cell_group-wise statistics

Examples

```
data("counts_obj")

  estimates =
  sccomp_estimate(
  counts_obj ,
   ~ 0 + type, ~1,  sample, cell_group, count,
    cores = 1
  ) |>

  sccomp_test("typecancer - typebenign")
```

`sce_obj` *sce_obj*

Description

Example SingleCellExperiment data set. SingleCellExperiment data objects can be directly used with sccomp_glm function.

Usage

```
data(sce_obj)
```

Format

A SingeCellExperiment object. SingeCellExperiment data objects can be directly used with sccomp_glm function.

`seurat_obj` *seurat_obj*

Description

Example Seurat data set. Seurat data objects can be directly used with sccomp_glm function.

Usage

```
data(seurat_obj)
```

Format

A Seurat object

`simulate_data` *simulate_data*

Description

This function simulates counts from a linear model.

Usage

```
simulate_data(
  .data,
  .estimate_object,
  formula_composition,
  formula_variability = NULL,
  .sample = NULL,
  .cell_group = NULL,
  .coefficients = NULL,
  variability_multiplier = 5,
  number_of_draws = 1,
  mcmc_seed = sample(1e+05, 1)
)
```

Arguments

`.data`	A tibble including a cell_group name column | sample name column | read counts column | factor columns | Pvalue column | a significance column
`.estimate_object`	The result of sccomp_estimate execution. This is used for sampling from real-data properties.

`formula_composition`
A formula. The sample formula used to perform the differential cell_group abundance analysis

`formula_variability`
A formula. The formula describing the model for differential variability, for example ~treatment. In most cases, if differentially variability is of interest, the formula should only include the factor of interest as a large anount of data is needed to define variability depending to each factors.

`.sample` A column name as symbol. The sample identifier

`.cell_group` A column name as symbol. The cell_group identifier

`.coefficients` The column names for coefficients, for example, c(b_0, b_1)

`variability_multiplier`
A real scalar. This can be used for artificially increasing the variability of the simulation for benchmarking purposes.

`number_of_draws`
An integer. How may copies of the data you want to draw from the model joint posterior distribution.

`mcmc_seed` An integer. Used for Markov-chain Monte Carlo reproducibility. By default a random number is sampled from 1 to 999999. This itself can be controlled by set.seed()

Value

A nested tibble `tbl` with cell_group-wise statistics

Examples

```
data("counts_obj")
library(dplyr)

estimate =
 sccomp_estimate(
 counts_obj ,
  ~ type, ~1,  sample, cell_group, count,
   cores = 1
 )

# Set coefficients for cell_groups. In this case all coefficients are 0 for simplicity.
counts_obj = counts_obj |> mutate(b_0 = 0, b_1 = 0)
# Simulate data
simulate_data(counts_obj, estimate, ~type, ~1, sample, cell_group, c(b_0, b_1))
```

`test_contrasts` *test_contrasts*

Description

This function test ocntrasts from a sccomp result.

Usage

```
test_contrasts(
  .data,
  contrasts = NULL,
  percent_false_positive = 5,
  test_composition_above_logit_fold_change = 0.2,
  pass_fit = TRUE
)
```

Arguments

`.data` A tibble. The result of sccomp_glm.

`contrasts` A vector of character strings. For example if your formula is `~ 0 + treatment` and the factor treatment has values `yes` and `no`, your contrast could be "contrasts = c(treatmentyes - treatmentno)".

`percent_false_positive`
A real between 0 and 100 non included. This used to identify outliers with a specific false positive rate.

`test_composition_above_logit_fold_change`
A positive integer. It is the effect threshold used for the hypothesis test. A value of 0.2 correspond to a change in cell proportion of 10% for a cell type with baseline proportion of 50%. That is, a cell type goes from 45% to 50%. When the baseline proportion is closer to 0 or 1 this effect thrshold has consistent value in the logit uncontrained scale.

`pass_fit` A boolean. Whether to pass the Stan fit as attribute in the output. Because the Stan fit can be very large, setting this to FALSE can be used to lower the memory imprint to save the output.

Value

A nested tibble `tbl` with cell_group-wise statistics

Examples

```
data("counts_obj")

  estimates =
  sccomp_glm(
  counts_obj ,
   ~ 0 + type, ~1,  sample, cell_group, count,
    check_outliers = FALSE,
    cores = 1
  ) |>

  test_contrasts("typecancer - typebenign")
```

Index