Package 'sccomp'

July 15, 2025

Title Tests differences in cell-type proportion for single-cell data, robust to outliers

Version 2.1.10

Description A robust and outlier-aware method for testing differences in cell-type proportion in 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

License_file LICENSE

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

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

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.2

Depends R (>= 4.3.0)

Imports instantiate (>= 0.2.3), stats, SingleCellExperiment, parallel, dplyr, tidyr, purrr, magrittr, rlang, tibble, boot, lifecycle, tidyselect, utils, ggplot2, ggrepel, patchwork, forcats, readr, scales, stringr, glue, crayon, cli, fansi

Suggests knitr, rmarkdown, BiocStyle, testthat (>= 3.0.0), markdown, loo, prettydoc, SeuratObject, tidyseurat, tidySingleCellExperiment, bayesplot, posterior

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

SystemRequirements CmdStan (https://mc-stan.org/users/interfaces/cmdstan)

biocViews Bayesian, Regression, DifferentialExpression, SingleCell, Metagenomics, FlowCytometry, Spatial

LazyData true

VignetteBuilder knitr

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

git_branch devel

git_last_commit 2ef5a77

git_last_commit_date 2025-07-09

Contents

---

clear_stan_model_cache
                          Clear Stan Model Cache

Description

This function removes the Stan model cache directory and all its contents. The cache is used by cmdstanr to store compiled Stan models. Clearing the cache can be useful when:

- You're experiencing issues with cached models

- You want to force recompilation of models

- You're switching between different versions of Stan

Usage

```
clear_stan_model_cache(cache_dir = sccomp_stan_models_cache_dir)
```

Arguments

cache_dir   A character string representing the path of the cache directory to delete. Defaults
                   to sccomp_stan_models_cache_dir, which is the default cache location used
                   by sccomp.

Details

The function uses unlink() with recursive = TRUE to remove the directory and all its contents. If
the cache directory doesn't exist, the function will simply print a message and continue. This is a
safe operation as cmdstanr will recreate the cache directory when needed.

Value

NULL invisibly. The function is called for its side effect of removing the cache directory.

References

S. Mangiola, A.J. Roth-Schulze, M. Trussart, E. Zozaya-Valdés, M. Ma, Z. Gao, A.F. Rubin, T.P.
Speed, H. Shim, & A.T. Papenfuss, sccomp: Robust differential composition and variability analy-
sis for single-cell data, Proc. Natl. Acad. Sci. U.S.A. 120 (33) e2203828120, https://doi.org/10.1073/pnas.2203828120
(2023).

See Also

- [cmdstan_path](#) for information about cmdstanr's cache management
- [unlink](#) for details about the underlying file removal function

Examples

```
# Clear the default sccomp cache directory
clear_stan_model_cache()

# Clear a specific cache directory
clear_stan_model_cache("path/to/custom/cache")

# Example of when you might want to clear the cache
## Not run:
# If you're experiencing issues with model compilation
clear_stan_model_cache()
# Then try running your model again

## End(Not run)
```

---

counts_obj   *counts_obj*

Description

A tidy example dataset containing cell counts per cell group (cluster), sample, and phenotype for
differential analysis. This dataset represents the counts of cells in various phenotypes and cell
groups across multiple samples.

Usage

```
data(counts_obj)
```

Format

A tidy data frame with the following columns:

- **sample**: Factor, representing the sample identifier.

- **type**: Factor, indicating the sample type (e.g., benign, cancerous).

- **phenotype**: Factor, representing the cell phenotype (e.g., B_cell, HSC, etc.).

- **count**: Integer, representing the number of cells for each cell group within each sample.

- **cell_group**: Factor, representing the cell group (e.g., BM, B1, Dm, etc.).

Value

A `tibble` representing cell counts per cluster, with columns for sample, type, phenotype, cell group, and counts.

---

handle_missing_values　　*Handle missing values in a data frame*

Description

Handle missing values in a data frame

Usage

```
handle_missing_values(data)
```

Arguments

data   A data frame to process

Value

A data frame with missing values handled:

- For categorical variables (factor/character): NA values are converted to a factor level "NA"

- For numeric variables: NA values are replaced with the mean of the column

---

multipanel_theme *multipanel_theme*

Description

A custom `ggplot2` theme used for creating publication-quality multi-panel plots. This theme modifies the appearance of plots by adjusting text sizes, spacing between panels, and axis formatting, ensuring better readability for complex figures.

Usage

```
data(multipanel_theme)
```

Format

A `ggplot2` theme with the following adjustments:

- **text**: Font size adjustments for plot titles, axis labels, and legend text.
- **panel.spacing**: Adjusts the spacing between panels in multi-panel plots.
- **axis.text**: Customises axis text appearance for better readability.

Value

A `ggplot2` theme object.

---

no_significance_df *no_significance_df*

Description

A small example dataset containing cell counts across samples, conditions, and cell groups. This dataset is used to demonstrate the use of sccomp functions in scenarios where there is no significant difference in cell composition between conditions.

Usage

```
data(no_significance_df)
```

Format

A tibble with the following columns:

- **sample**: Character. Identifier for each sample.
- **condition**: Character. Experimental condition or group (e.g., "X" or "Y").
- **cell_group**: Character. Cell group or cell type (e.g., "A", "B").
- **count**: Numeric. Count of cells in the given sample, condition, and cell group.

Value

A tibble with 34 rows and 4 columns: `sample`, `condition`, `cell_group`, and `count`.

Examples

```
data(no_significance_df)
head(no_significance_df)
```

---

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,
  significance_threshold = 0.05,
  test_composition_above_logit_fold_change = attr(.data,
    "test_composition_above_logit_fold_change"),
  significance_statistic = c("FDR", "pH0"),
  show_fdr_message = TRUE,
  ...
)
```

Arguments

x
: A tibble including a cell_group name column | sample name column | read counts column | factor columns | Pvalue column | a significance column

significance_threshold
: Numeric value specifying the significance threshold for highlighting differences. Default is 0.025.

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.

significance_statistic
: Character vector indicating which statistic to highlight. Default is "FDR".

show_fdr_message
: Logical. Whether to show the Bayesian FDR interpretation message on the plot. Default is TRUE.

...
: For internal use

Value

A ggplot

References

S. Mangiola, A.J. Roth-Schulze, M. Trussart, E. Zozaya-Valdés, M. Ma, Z. Gao, A.F. Rubin, T.P. Speed, H. Shim, & A.T. Papenfuss, sccomp: Robust differential composition and variability analysis for single-cell data, Proc. Natl. Acad. Sci. U.S.A. 120 (33) e2203828120, https://doi.org/10.1073/pnas.2203828120 (2023).

Examples

```
print("cmdstanr is needed to run this example.")
# Note: Before running the example, ensure that the 'cmdstanr' package is installed:
# install.packages("cmdstanr", repos = c("https://stan-dev.r-universe.dev/", getOption("repos")))


  if (instantiate::stan_cmdstan_exists()) {
    data("counts_obj")

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

    # estimate |> plot()
  }
```

---

plot_1D_intervals *Plot 1D Intervals for Cell-group Effects*

Description

This function creates a series of 1D interval plots for cell-group effects, highlighting significant differences based on a given significance threshold.

Usage

```
plot_1D_intervals(
  .data,
  significance_threshold = 0.05,
  test_composition_above_logit_fold_change = attr(.data,
    "test_composition_above_logit_fold_change"),
  show_fdr_message = TRUE,
  significance_statistic = c("FDR", "pH0")
)
```

Arguments

.data   Data frame containing the main data.

significance_threshold

                Numeric value specifying the significance threshold for highlighting differences.

```
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.

```
show_fdr_message
```
> Logical. Whether to show the Bayesian FDR interpretation message on the plot. Default is TRUE.

```
significance_statistic
```
> Character vector indicating which statistic to highlight. Default is "FDR".

Value

A combined plot of 1D interval plots.

Examples

```
print("cmdstanr is needed to run this example.")


  if (instantiate::stan_cmdstan_exists()) {
    data("counts_obj")

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

  # Example usage:
  my_plot = plot_1D_intervals(estimate)

  }
```

---

plot_2D_intervals   *Plot 2D Intervals for Mean-Variance Association*

Description

This function creates a 2D interval plot for mean-variance association, highlighting significant differences based on a given significance threshold.

Usage

```
plot_2D_intervals(
  .data,
  significance_threshold = 0.05,
  test_composition_above_logit_fold_change = attr(.data,
    "test_composition_above_logit_fold_change"),
  show_fdr_message = TRUE,
  significance_statistic = c("FDR", "pH0")
)
```

Arguments

.data   Data frame containing the main data.

significance_threshold

Numeric value specifying the significance threshold for highlighting differences. Default is 0.025.

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.

show_fdr_message

Logical. Whether to show the Bayesian FDR interpretation message on the plot. Default is TRUE.

significance_statistic

Character vector indicating which statistic to highlight. Default is "FDR".

Value

A ggplot object representing the 2D interval plot.

Examples

```
print("cmdstanr is needed to run this example.")


  if (instantiate::stan_cmdstan_exists()) {
    data("counts_obj")

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

  # Example usage:
  my_plot = plot_2D_intervals(estimate)
```

```
  }
```

---

plot_scatterplot   *Plot Scatterplot of Cell-group Proportion*

Description

This function creates a scatterplot of cell-group proportions, optionally highlighting significant differences based on a given significance threshold.

Usage

```
plot_scatterplot(
  .data,
  data_proportion,
  factor_of_interest,
  .cell_group,
  .sample,
  significance_threshold = 0.05,
  my_theme
)
```

Arguments

.data	Data frame containing the main data.
data_proportion	
	Data frame containing proportions of cell groups.
factor_of_interest	
	A factor indicating the biological condition of interest.
.cell_group	The cell group to be analysed.
.sample	The sample identifier.
significance_threshold	
	Numeric value specifying the significance threshold for highlighting differences. Default is 0.025.
my_theme	A ggplot2 theme object to be applied to the plot.

Value

A ggplot object representing the scatterplot.

Examples

```
# Example usage:
# plot_scatterplot(.data, data_proportion, "condition", "cell_group", "sample", 0.025, theme_minimal())
```

---

print.sccomp_tbl   *Print method for sccomp objects*

Description

Print method for sccomp objects.

The print method for sccomp objects provides a summary of the model specifications, data dimensions, and convergence diagnostics.

The output is formatted to be easy to read and understand.

Usage

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

Arguments

x   A sccomp object

...   Additional arguments passed to print

Value

The printed object

---

sccomp_boxplot   *sccomp_boxplot*

Description

Creates a boxplot visualization of the model results from sccomp. This function plots the estimated cell proportions across samples, highlighting significant changes in cell composition according to a specified factor.

Usage

```
sccomp_boxplot(
  .data,
  factor,
  significance_threshold = 0.05,
  test_composition_above_logit_fold_change = attr(.data,
    "test_composition_above_logit_fold_change"),
  remove_unwanted_effects = FALSE
)
```

Arguments

`.data`	A tibble containing the results from sccomp_estimate and sccomp_test, including the columns: cell_group name, sample name, read counts, factor(s), p-values, and significance indicators.
`factor`	A character string specifying the factor of interest included in the model for stratifying the boxplot.
`significance_threshold`	
	A numeric value indicating the False Discovery Rate (FDR) threshold for labeling significant cell-groups. Defaults to 0.05.
`test_composition_above_logit_fold_change`	
	A positive numeric value representing the effect size threshold used in the hypothesis test. A value of 0.2 corresponds to a change in cell proportion of approximately 10% for a cell type with a baseline proportion of 50% (e.g., from 45% to 55%). This threshold is consistent on the logit-unconstrained scale, even when the baseline proportion is close to 0 or 1.
`remove_unwanted_effects`	
	A logical value indicating whether to remove unwanted variation from the data before plotting. Defaults to FALSE.

Value

A ggplot object representing the boxplot of cell proportions across samples, stratified by the specified factor.

References

S. Mangiola, A.J. Roth-Schulze, M. Trussart, E. Zozaya-Valdés, M. Ma, Z. Gao, A.F. Rubin, T.P. Speed, H. Shim, & A.T. Papenfuss, sccomp: Robust differential composition and variability analysis for single-cell data, Proc. Natl. Acad. Sci. U.S.A. 120 (33) e2203828120, https://doi.org/10.1073/pnas.2203828120 (2023).

Examples

```
print("cmdstanr is needed to run this example.")
# Note: Before running the example, ensure that the 'cmdstanr' package is installed:
# install.packages("cmdstanr", repos = c("https://stan-dev.r-universe.dev/", getOption("repos")))


if (instantiate::stan_cmdstan_exists()) {
    data("counts_obj")

    estimate <- sccomp_estimate(
      counts_obj,
      formula_composition = ~ type,
      formula_variability = ~ 1,
      sample = "sample",
      cell_group = "cell_group",
      abundance = "count",
      cores = 1
    ) |>
    sccomp_test()

    # Plot the boxplot of estimated cell proportions
```

```
    sccomp_boxplot(
        .data = estimate,
        factor = "type",
        significance_threshold = 0.05
    )
}
```

---

sccomp_calculate_residuals

Calculate Residuals Between Observed and Predicted Proportions

---

Description

sccomp_calculate_residuals computes the residuals between observed cell group proportions and the predicted proportions from a fitted sccomp model. This function is useful for assessing model fit and identifying cell groups or samples where the model may not adequately capture the observed data. The residuals are calculated as the difference between the observed proportions and the predicted mean proportions from the model.

Usage

```
sccomp_calculate_residuals(.data)
```

Arguments

.data   A tibble of class sccomp_tbl, which is the result of sccomp_estimate(). This tibble contains the fitted model and associated data necessary for calculating residuals.

Details

The function performs the following steps:

1. Extracts the predicted mean proportions for each cell group and sample using sccomp_predict().
2. Calculates the observed proportions from the original count data.
3. Computes residuals by subtracting the predicted proportions from the observed proportions.
4. Returns a tibble containing the sample, cell group, residuals, and exposure (total counts per sample).

Value

A tibble (tbl) with the following columns:

- **sample** - A character column representing the sample identifiers.
- **cell_group** - A character column representing the cell group identifiers.
- **residuals** - A numeric column representing the residuals, calculated as the difference between observed and predicted proportions.
- **exposure** - A numeric column representing the total counts (sum of counts across cell groups) for each sample.

References

S. Mangiola, A.J. Roth-Schulze, M. Trussart, E. Zozaya-Valdés, M. Ma, Z. Gao, A.F. Rubin, T.P. Speed, H. Shim, & A.T. Papenfuss, sccomp: Robust differential composition and variability analysis for single-cell data, Proc. Natl. Acad. Sci. U.S.A. 120 (33) e2203828120, https://doi.org/10.1073/pnas.2203828120 (2023).

Examples

```
  if (instantiate::stan_cmdstan_exists() && .Platform$OS.type == "unix") {
# Load example data
data("counts_obj")

# Fit the sccomp model
estimates <- sccomp_estimate(
  counts_obj,
  formula_composition = ~ type,
  formula_variability = ~1,
  sample = "sample",
  cell_group = "cell_group",
  abundance = "count",
  approximate_posterior_inference = "all",
  cores = 1
)

# Calculate residuals
residuals <- sccomp_calculate_residuals(estimates)

# View the residuals
print(residuals)
}
```

| sccomp_estimate | *Main Function for SCCOMP Estimate* |

Description

The sccomp_estimate function performs linear modeling on a table of cell counts or proportions, which includes a cell-group identifier, sample identifier, abundance (counts or proportions), and factors (continuous or discrete). The user can define a linear model using an 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,
  abundance = NULL,
```

```
     cores = detectCores(),
     bimodal_mean_variability_association = FALSE,
     percent_false_positive = 5,
     inference_method = "pathfinder",
     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,
     output_directory = "sccomp_draws_files",
     verbose = TRUE,
     enable_loo = FALSE,
     noise_model = "multi_beta_binomial",
     exclude_priors = FALSE,
     use_data = TRUE,
     mcmc_seed = sample_seed(),
     max_sampling_iterations = 20000,
     pass_fit = TRUE,
     sig_figs = 9,
     ...,
     .count = NULL,
     approximate_posterior_inference = NULL,
     variational_inference = NULL,
     .sample = NULL,
     .cell_group = NULL,
     .abundance = NULL
   )
```

Arguments

`.data`	A tibble including cell_group name column, sample name column, abundance column (counts or proportions), 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 a character string for the sample identifier. Replaces the deprecated `.sample`.
`cell_group`	A column name as a character string for the cell-group identifier. Replaces the deprecated `.cell_group`.
`abundance`	A column name as a character string for the cell-group abundance, which can be counts (> 0) or proportions (between 0 and 1, summing to 1 across `cell_group`). Replaces the deprecated `.abundance` and `.count`.
`cores`	Number of cores to use for parallel calculations.
`bimodal_mean_variability_association`	
	Logical, whether to model mean-variability as bimodal.
`percent_false_positive`	
	A real number between 0 and 100 for outlier identification.
`inference_method`	
	Character string specifying the inference method to use ('pathfinder', 'hmc', or 'variational'). Replaces the deprecated `approximate_posterior_inference` and `variational_inference`.

prior_mean   A list specifying prior knowledge about the mean distribution, including inter-
                    cept and coefficients.

prior_overdispersion_mean_association
                    A list specifying prior knowledge about mean/variability association.

.sample_cell_group_pairs_to_exclude
                    A column name indicating sample/cell-group pairs to exclude.

output_directory
                    A character string specifying the output directory for Stan draws.

verbose   Logical, whether to print progression details.

enable_loo   Logical, whether to enable model comparison using the LOO package.

noise_model   A character string specifying the noise model (e.g., 'multi_beta_binomial').

exclude_priors   Logical, whether to run a prior-free model.

use_data   Logical, whether to run the model data-free.

mcmc_seed   An integer seed for MCMC reproducibility.

max_sampling_iterations
                    Integer to limit the maximum number of iterations for large datasets.

pass_fit   Logical, whether to include the Stan fit as an attribute in the output.

sig_figs   Number of significant figures to use for Stan model output. Default is 9.

...   Additional arguments passed to the `cmdstanr::sample` function.

.count   **DEPRECATED**. Use abundance instead.

approximate_posterior_inference
                    DEPRECATED. Use inference_method instead.

variational_inference
                    DEPRECATED. Use inference_method instead.

.sample   **DEPRECATED**. Use sample instead.

.cell_group   **DEPRECATED**. Use cell_group instead.

.abundance   **DEPRECATED**. Use abundance instead.

Value

A tibble (`tbl`), with the following columns:

- cell_group - The cell groups being tested.

- parameter - The parameter being estimated from the design matrix described by the input
  formula_composition and formula_variability.

- factor - The covariate factor in the formula, if applicable (e.g., not present for Intercept or
  contrasts).

- 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 for a composition (c) parameter.

- c_pH0 - Probability of the c_effect being smaller or bigger than the `test_composition_above_logit_fold_chang`
  argument.

- c_FDR - False discovery rate of the c_effect being smaller or bigger than the `test_composition_above_logit_fo`
  argument. False discovery rate for Bayesian models is calculated differently from frequentists
  models, as detailed in Mangiola et al, PNAS 2023.

- c_n_eff - Effective sample size, the number of independent draws in the sample. The higher, the better.
- c_R_k_hat - R statistic, a measure of chain equilibrium, should be within 0.05 of 1.0.
- 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 v_effect being smaller or bigger than the `test_composition_above_logit_fold_chan`
  argument.
- v_FDR - False discovery rate of the v_effect being smaller or bigger than the `test_composition_above_logit_fo`
  argument. False discovery rate for Bayesian models is calculated differently from frequentists
  models, as detailed in Mangiola et al, PNAS 2023.
- v_n_eff - Effective sample size for a variability (v) parameter.
- v_R_k_hat - R statistic for a variability (v) parameter, a measure of chain equilibrium.

The function also attaches several attributes to the result:

- count_data - The original count data used in the analysis, stored as an attribute for efficient
  access.
- model_input - The model input data used for fitting.
- formula_composition - The formula used for composition modeling.
- formula_variability - The formula used for variability modeling.
- fit - The Stan fit object (if pass_fit = TRUE).

References

S. Mangiola, A.J. Roth-Schulze, M. Trussart, E. Zozaya-Valdés, M. Ma, Z. Gao, A.F. Rubin, T.P.
Speed, H. Shim, & A.T. Papenfuss, sccomp: Robust differential composition and variability analysis for single-cell data, Proc. Natl. Acad. Sci. U.S.A. 120 (33) e2203828120, https://doi.org/10.1073/pnas.2203828120
(2023).

Examples

```
print("cmdstanr is needed to run this example.")
# Note: Before running the example, ensure that the 'cmdstanr' package is installed:
# install.packages("cmdstanr", repos = c("https://stan-dev.r-universe.dev/", getOption("repos")))


  if (instantiate::stan_cmdstan_exists()) {
    data("counts_obj")

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

    # Note!
    # If counts are available, do not use proportion.
```

```
    # Using proportion ignores the high uncertainty of low counts

    estimate_proportion <- sccomp_estimate(
        counts_obj,
        ~ type,
        ~1,
        "sample",
        "cell_group",
        "proportion",
        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_seed(),
  summary_instead_of_draws = TRUE,
  robust = FALSE
)
```

Arguments

fit   The result of sccomp_estimate.

formula_composition

                     A formula. The formula describing the model for differential abundance, for ex-
                     ample ~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()

summary_instead_of_draws

> Return the summary values (i.e. mean and quantiles) of the predicted proportions, or return single draws. Single draws can be helful to better analyse the uncertainty of the prediction.

robust
> A logical. If TRUE, use robust statistics (median and median absolute deviation) instead of classical statistics (mean and standard deviation) for the summary calculations.

Value

A tibble (`tbl`) with the following columns:

- **cell_group** - A character column representing the cell group being tested.

- **sample** - A factor column representing the sample name for which the predictions are made.

- **proportion_mean** - A numeric column representing the predicted mean (or median when robust=TRUE) proportions from the model.

- **proportion_lower** - A numeric column representing the lower bound (2.5%) of the 95% credible interval for the predicted proportions.

- **proportion_upper** - A numeric column representing the upper bound (97.5%) of the 95% credible interval for the predicted proportions.

References

S. Mangiola, A.J. Roth-Schulze, M. Trussart, E. Zozaya-Valdés, M. Ma, Z. Gao, A.F. Rubin, T.P. Speed, H. Shim, & A.T. Papenfuss, sccomp: Robust differential composition and variability analysis for single-cell data, Proc. Natl. Acad. Sci. U.S.A. 120 (33) e2203828120, https://doi.org/10.1073/pnas.2203828120 (2023).

Examples

```
print("cmdstanr is needed to run this example.")
# Note: Before running the example, ensure that the 'cmdstanr' package is installed:
# install.packages("cmdstanr", repos = c("https://stan-dev.r-universe.dev/", getOption("repos")))


  if (instantiate::stan_cmdstan_exists() && .Platform$OS.type == "unix") {
    data("counts_obj")

    sccomp_estimate(
      counts_obj,
      ~ type, ~1, "sample", "cell_group", "count",
      cores = 1
    ) |>
    sccomp_predict()
  }
```

sccomp_proportional_fold_change

Calculate Proportional Fold Change from sccomp Estimated Effects

Description

This function calculates the proportional fold change between two conditions using the estimated effects from a sccomp model. The fold changes are derived from the model's posterior predictions rather than raw counts, providing a more robust estimate that accounts for the model's uncertainty and covariate effects.

Note! This statistic is descriptive and should not be used to define significance - use sccomp_test() for that. While fold changes in proportions are easier to interpret than changes in logit space, they are not linear (the same proportional change has different meaning for rare vs abundant cell types). In contrast, the logit scale used internally by sccomp provides linear effects that are more appropriate for statistical inference.

Usage

```
sccomp_proportional_fold_change(.data, formula_composition, from, to)
```

Arguments

.data	A sccomp estimate object (of class 'sccomp_tbl') obtained from running sccomp_estimate(). This object contains the fitted model and estimated effects.
formula_composition	
	The formula specifying which model effects to use for calculating fold changes. This should match or be a subset of the formula used in the original sccomp_estimate() call.
from	Character string specifying the reference/control condition (e.g., "benign").
to	Character string specifying the comparison condition (e.g., "cancer").

Value

A tibble with the following columns:

- cell_group - The cell group identifier
- proportion_fold_change - The estimated fold change in proportions between conditions. Positive values indicate increases, negative values indicate decreases.
- average_uncertainty - The average uncertainty in the fold change estimate, derived from the credible intervals
- statement - A text description of the fold change, including the direction and the estimated proportions

Examples

```
print("cmdstanr is needed to run this example.")
# Note: Before running the example, ensure that the 'cmdstanr' package is installed:
# install.packages("cmdstanr", repos = c("https://stan-dev.r-universe.dev/", getOption("repos")))
```

```
if (instantiate::stan_cmdstan_exists()) {
  # Load example data
  data("counts_obj")

  # First estimate the composition effects
  estimate <- sccomp_estimate(
      counts_obj,
      ~ type,
      ~1,
      "sample",
      "cell_group",
      "count",
      cores = 1
  )

  # Calculate proportional fold changes from the estimated effects
  estimate |>
  sccomp_proportional_fold_change(
    formula_composition = ~  type,
    from = "benign",
    to = "cancer"
  )
}
```

sccomp_remove_outliers

sccomp_remove_outliers main

Description

The sccomp_remove_outliers function takes as input a table of cell counts with columns for cell-group identifier, sample identifier, integer count, and factors (continuous or discrete). The user can define a linear model using 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_remove_outliers(
  .estimate,
  percent_false_positive = 5,
  cores = detectCores(),
  inference_method = attr(.estimate, "inference_method"),
  output_directory = "sccomp_draws_files",
  verbose = TRUE,
  mcmc_seed = sample_seed(),
  max_sampling_iterations = 20000,
  enable_loo = FALSE,
  sig_figs = 9,
  approximate_posterior_inference = NULL,
  variational_inference = NULL,
```

```
  ...
)
```

Arguments

`.estimate`	A tibble including a cell_group name column, sample name column, read counts column (optional depending on the input class), and factor columns.
`percent_false_positive`	
	A real number between 0 and 100 (not inclusive), used to identify outliers with a specific false positive rate.
`cores`	Integer, the number of cores to be used for parallel calculations.
`inference_method`	
	Character string specifying the inference method to use ('pathfinder', 'hmc', or 'variational').
`output_directory`	
	A character string specifying the output directory for Stan draws.
`verbose`	Logical, whether to print progression details.
`mcmc_seed`	Integer, used for Markov-chain Monte Carlo reproducibility. By default, a random number is sampled from 1 to 999999.
`max_sampling_iterations`	
	Integer, limits the maximum number of iterations in case a large dataset is used, to limit computation time.
`enable_loo`	Logical, whether to enable model comparison using the R package LOO. This is useful for comparing fits between models, similar to ANOVA.
`sig_figs`	Number of significant figures to use for Stan model output. Default is 9.
`approximate_posterior_inference`	
	DEPRECATED, use the `variational_inference` argument.
`variational_inference`	
	DEPRECATED Logical, whether to use variational Bayes for posterior inference. It is faster and convenient. Setting this argument to `FALSE` runs full Bayesian (Hamiltonian Monte Carlo) inference, which is slower but the gold standard.
`...`	Additional arguments passed to the `cmdstanr::sample` function.

Value

A tibble (`tbl`), with the following columns:

- cell_group - The cell groups being tested.
- parameter - The parameter being estimated from the design matrix described by the input formula_composition and formula_variability.
- factor - The covariate factor in the formula, if applicable (e.g., not present for Intercept or contrasts).
- 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 for a composition (c) parameter.
- c_pH0 - Probability of the c_effect being smaller or bigger than the `test_composition_above_logit_fold_chang` argument.

- c_FDR - False discovery rate of the c_effect being smaller or bigger than the `test_composition_above_logit_fo` argument. False discovery rate for Bayesian models is calculated differently from frequentists models, as detailed in Mangiola et al, PNAS 2023.

- c_n_eff - Effective sample size, the number of independent draws in the sample. The higher, the better.

- c_R_k_hat - R statistic, a measure of chain equilibrium, should be within 0.05 of 1.0.

- 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 v_effect being smaller or bigger than the `test_composition_above_logit_fold_chang` argument.

- v_FDR - False discovery rate of the v_effect being smaller or bigger than the `test_composition_above_logit_fo` argument. False discovery rate for Bayesian models is calculated differently from frequentists models, as detailed in Mangiola et al, PNAS 2023.

- v_n_eff - Effective sample size for a variability (v) parameter.

- v_R_k_hat - R statistic for a variability (v) parameter, a measure of chain equilibrium.

The function also attaches several attributes to the result:

- count_data - The original count data used in the analysis, stored as an attribute for efficient access.

- model_input - The model input data used for fitting.

- formula_composition - The formula used for composition modeling.

- formula_variability - The formula used for variability modeling.

- fit - The Stan fit object (if pass_fit = TRUE).

References

S. Mangiola, A.J. Roth-Schulze, M. Trussart, E. Zozaya-Valdés, M. Ma, Z. Gao, A.F. Rubin, T.P. Speed, H. Shim, & A.T. Papenfuss, sccomp: Robust differential composition and variability analysis for single-cell data, Proc. Natl. Acad. Sci. U.S.A. 120 (33) e2203828120, https://doi.org/10.1073/pnas.2203828120 (2023).

Examples

```
print("cmdstanr is needed to run this example.")
# Note: Before running the example, ensure that the 'cmdstanr' package is installed:
# install.packages("cmdstanr", repos = c("https://stan-dev.r-universe.dev/", getOption("repos")))


  if (instantiate::stan_cmdstan_exists()) {
    data("counts_obj")

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

```
  ) |>
  sccomp_remove_outliers(cores = 1)
}
```

```
sccomp_remove_unwanted_effects
```
Remove unwanted effects from a sccomp_tbl object

Description

This method removes unwanted effects from a dataset using the model estimates. For example, if you fit your data with the formula ~ `factor_1` + `factor_2` and use the formula ~ `factor_1` to remove unwanted variation, the `factor_2` effect will be factored out.

Usage

```
sccomp_remove_unwanted_effects(
  .data,
  formula_composition_keep = NULL,
  formula_composition = NULL,
  cores = detectCores()
)
```

Arguments

`.data`	A `sccomp_tbl` object. The result of `sccomp_estimate`.
`formula_composition_keep`	
	A formula. The formula describing the model for differential abundance, for example `~type`. In this case, only the effect of the `type` factor will be preserved, while all other factors will be factored out.
`formula_composition`	
	DEPRECATED. Use `formula_composition_keep` instead.
`cores`	Integer, the number of cores to be used for parallel calculations.

Value

A tibble (`tbl`) with the following columns:

- **sample** - A character column representing the sample name for which data was adjusted.
- **cell_group** - A character column representing the cell group being tested.
- **adjusted_proportion** - A numeric column representing the adjusted proportion after removing unwanted variation.
- **adjusted_counts** - A numeric column representing the adjusted counts after removing unwanted variation.
- **logit_residuals** - A numeric column representing the logit residuals calculated after adjustment.

References

S. Mangiola, A.J. Roth-Schulze, M. Trussart, E. Zozaya-Valdés, M. Ma, Z. Gao, A.F. Rubin, T.P. Speed, H. Shim, & A.T. Papenfuss, sccomp: Robust differential composition and variability analysis for single-cell data, Proc. Natl. Acad. Sci. U.S.A. 120 (33) e2203828120, https://doi.org/10.1073/pnas.2203828120 (2023).

Examples

```
print("cmdstanr is needed to run this example.")
# Note: Before running the example, ensure that the 'cmdstanr' package is installed:
# install.packages("cmdstanr", repos = c("https://stan-dev.r-universe.dev/", getOption("repos")))


  if (instantiate::stan_cmdstan_exists()) {
    data("counts_obj")

    estimates = sccomp_estimate(
      counts_obj,
      ~ type, ~1, "sample", "cell_group", "count",
      cores = 1
    ) |>
    sccomp_remove_unwanted_effects()
  }
```

---

sccomp_remove_unwanted_variation

DEPRECATED: Remove Unwanted Variation from sccomp Estimates

Description

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

Usage

```
sccomp_remove_unwanted_variation(
  .data,
  formula_composition_keep = NULL,
  formula_composition = NULL,
  formula_variability = NULL,
  cores = detectCores()
)
```

Arguments

.data   A tibble. The result of sccomp_estimate.

formula_composition_keep

> A formula. The formula describing the model for differential abundance, for example ~type. In this case, only the effect of the `type` factor will be preserved, while all other factors will be factored out.

formula_composition

> DEPRECATED. Use `formula_composition_keep` instead.

formula_variability

> DEPRECATED. Use `formula_variability_keep` instead.

cores   Integer, the number of cores to be used for parallel calculations.

Value

A tibble (`tbl`) with the following columns:

- **sample** - A character column representing the sample name for which data was adjusted.

- **cell_group** - A character column representing the cell group being tested.

- **adjusted_proportion** - A numeric column representing the adjusted proportion after removing unwanted variation.

- **adjusted_counts** - A numeric column representing the adjusted counts after removing unwanted variation.

- **logit_residuals** - A numeric column representing the logit residuals calculated after adjustment.

References

S. Mangiola, A.J. Roth-Schulze, M. Trussart, E. Zozaya-Valdés, M. Ma, Z. Gao, A.F. Rubin, T.P. Speed, H. Shim, & A.T. Papenfuss, sccomp: Robust differential composition and variability analysis for single-cell data, Proc. Natl. Acad. Sci. U.S.A. 120 (33) e2203828120, https://doi.org/10.1073/pnas.2203828120 (2023).

Examples

```
print("cmdstanr is needed to run this example.")
# Note: Before running the example, ensure that the 'cmdstanr' package is installed:
# install.packages("cmdstanr", repos = c("https://stan-dev.r-universe.dev/", getOption("repos")))


  if (instantiate::stan_cmdstan_exists()) {
    data("counts_obj")

    estimates = sccomp_estimate(
      counts_obj,
      ~ type, ~1, "sample", "cell_group", "count",
      cores = 1
    ) |>
    sccomp_remove_unwanted_variation()
  }
```

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_seed()
)
```

Arguments

fit   The result of sccomp_estimate.

formula_composition

A formula. The formula describing the model for differential abundance, for ex-
ample ~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 amount 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 tibble `tbl` with cell_group-wise statistics

A tibble (`tbl`), with the following columns:

- **cell_group** - A character column representing the cell group being tested.

- **sample** - A factor column representing the sample name from which data was generated.

- **generated_proportions** - A numeric column representing the proportions generated from the model.

- **generated_counts** - An integer column representing the counts generated from the model.

- **replicate** - An integer column representing the replicate number, where each row corresponds to a different replicate of the data.

References

S. Mangiola, A.J. Roth-Schulze, M. Trussart, E. Zozaya-Valdés, M. Ma, Z. Gao, A.F. Rubin, T.P. Speed, H. Shim, & A.T. Papenfuss, sccomp: Robust differential composition and variability analysis for single-cell data, Proc. Natl. Acad. Sci. U.S.A. 120 (33) e2203828120, https://doi.org/10.1073/pnas.2203828120 (2023).

Examples

```
print("cmdstanr is needed to run this example.")
# Note: Before running the example, ensure that the 'cmdstanr' package is installed:
# install.packages("cmdstanr", repos = c("https://stan-dev.r-universe.dev/", getOption("repos")))


  if (instantiate::stan_cmdstan_exists() && .Platform$OS.type == "unix") {
    data("counts_obj")

    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 = 0.1,
  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 "constrasts = 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 tibble (`tbl`), with the following columns:

- cell_group - The cell groups being tested.

- parameter - The parameter being estimated from the design matrix described by the input formula_composition and formula_variability.

- factor - The covariate factor in the formula, if applicable (e.g., not present for Intercept or contrasts).

- 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 for a composition (c) parameter.

- c_pH0 - Probability of the c_effect being smaller or bigger than the `test_composition_above_logit_fold_change` argument.

- 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.

- c_R_k_hat - R statistic, a measure of chain equilibrium, should be within 0.05 of 1.0.

- 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).

- 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.

- v_R_k_hat - R statistic for a variability (v) parameter.

- count_data - Nested input count data.

References

S. Mangiola, A.J. Roth-Schulze, M. Trussart, E. Zozaya-Valdés, M. Ma, Z. Gao, A.F. Rubin, T.P. Speed, H. Shim, & A.T. Papenfuss, sccomp: Robust differential composition and variability analysis for single-cell data, Proc. Natl. Acad. Sci. U.S.A. 120 (33) e2203828120, https://doi.org/10.1073/pnas.2203828120 (2023).

Examples

```
print("cmdstanr is needed to run this example.")
# Note: Before running the example, ensure that the 'cmdstanr' package is installed:
# install.packages("cmdstanr", repos = c("https://stan-dev.r-universe.dev/", getOption("repos")))


  if (instantiate::stan_cmdstan_exists()) {
    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` object containing gene expression data for 106,297 cells across two assays: counts and logcounts. The object includes metadata and assay data for RNA expression, which can be used directly in differential analysis functions like `sccomp_glm`.

Usage

```
data(sce_obj)
```

Format

A `SingleCellExperiment` object with the following structure:

- **assays**: Two assays: counts (raw RNA counts) and logcounts (log-transformed counts).
- **rowData**: No additional row-level metadata is present.
- **colData**: Metadata for each cell, including six fields: sample, type, nFeature_RNA, ident, and others.
- **dim**: 1 feature and 106,297 cells.
- **colnames**: Cell identifiers for all 106,297 cells.

Value

A `SingleCellExperiment` object containing single-cell RNA expression data.

---

seurat_obj   *seurat_obj*

Description

Example Seurat object containing gene expression data for 106,297 cells across a single assay. The object includes RNA counts and data layers, but no variable features are defined. This dataset can be directly used with functions like sccomp_glm for differential abundance analysis.

Usage

```
data(seurat_obj)
```

Format

A Seurat object with the following structure:

- **assays**: Contains gene expression data. The active assay is RNA, with 1 feature and no variable features.
- **layers**: Two layers: counts and data, representing raw and processed RNA expression values, respectively.
- **samples**: 106,297 samples (cells) within the RNA assay.

Value

A Seurat object containing single-cell RNA expression data.

---

simulate_data   *simulate_data*

Description

This function simulates data from a fitted 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_seed(),
  cores = detectCores(),
  sig_figs = 9
)
```

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 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
`.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()#' @param cores Integer, the number of cores to be used for parallel calculations.
`cores`	Integer, the number of cores to be used for parallel calculations.
`sig_figs`	Number of significant figures to use for Stan model output. Default is 9.

Value

A tibble (`tbl`) with the following columns:

- **sample** - A character column representing the sample name.
- **type** - A factor column representing the type of the sample.
- **phenotype** - A factor column representing the phenotype in the data.
- **count** - An integer column representing the original cell counts.
- **cell_group** - A character column representing the cell group identifier.
- **b_0** - A numeric column representing the first coefficient used for simulation.
- **b_1** - A numeric column representing the second coefficient used for simulation.
- **generated_proportions** - A numeric column representing the generated proportions from the simulation.
- **generated_counts** - An integer column representing the generated cell counts from the simulation.
- **replicate** - An integer column representing the replicate number for each draw from the posterior distribution.

References

S. Mangiola, A.J. Roth-Schulze, M. Trussart, E. Zozaya-Valdés, M. Ma, Z. Gao, A.F. Rubin, T.P. Speed, H. Shim, & A.T. Papenfuss, sccomp: Robust differential composition and variability analysis for single-cell data, Proc. Natl. Acad. Sci. U.S.A. 120 (33) e2203828120, https://doi.org/10.1073/pnas.2203828120 (2023).

Examples

```
print("cmdstanr is needed to run this example.")
# Note: Before running the example, ensure that the 'cmdstanr' package is installed:
# install.packages("cmdstanr", repos = c("https://stan-dev.r-universe.dev/", getOption("repos")))


  if (instantiate::stan_cmdstan_exists()) {
    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))
  }
```

Index