Package: censcyt (via r-universe)

October 20, 2026

Version 1.21.0

Title Differential abundance analysis with a right censored covariate in high-dimensional cytometry

Description Methods for differential abundance analysis in high-dimensional cytometry data when a covariate is subject to right censoring (e.g. survival time) based on multiple imputation and generalized linear mixed models.

URL https://github.com/retogerber/censcyt

BugReports https://github.com/retogerber/censcyt/issues

License MIT + file LICENSE

biocViews ImmunoOncology, FlowCytometry, Proteomics, SingleCell, CellBasedAssays, CellBiology, Clustering, FeatureExtraction, Software, Survival

Depends R (>= 4.0), diffcyt

Imports BiocParallel, broom.mixed, dirmult, dplyr, edgeR, fitdistrplus, lme4, magrittr, MASS, methods, mice, multcomp, purrr, rlang, S4Vectors, stats, stringr, SummarizedExperiment, survival, tibble, tidyr, utils

VignetteBuilder knitr

Suggests BiocStyle, knitr, rmarkdown, testthat, ggplot2

RoxygenNote 7.1.2

Config/pak/sysreqs cmake libfontconfig1-dev libfreetype6-dev libglpk-dev make libicu-dev libpng-dev libxml2-dev libssl-dev perl libx11-dev zlib1g-dev

Repository https://bioc.r-universe.dev

Date/Publication 2026-04-28 12:54:37 UTC

RemoteUrl https://github.com/bioc/censcyt

RemoteRef HEAD

RemoteSha 8bd67697e52ccdfa77ff9888c13c653b528e8d5b

Contents

censcyt 2
conditional_multiple_imputation 7
createFormula 9
simulate_multicluster 11
simulate_singlecluster 13
testDA_censoredGLMM 15

Index **19**

`censcyt` *Run* `censcyt` *pipeline*

Description

Wrapper function to run complete `censcyt` pipeline

Usage

```
censcyt(
  d_input,
  experiment_info = NULL,
  marker_info = NULL,
  design = NULL,
  formula = NULL,
  contrast,
  analysis_type = c("DA"),
  method_DA = c("censcyt-DA-censored-GLMM"),
  markers_to_test = NULL,
  clustering_to_use = NULL,
  cols_to_include = NULL,
  subsampling = FALSE,
  n_sub = NULL,
  seed_sub = NULL,
  transform = TRUE,
  cofactor = 5,
  cols_clustering = NULL,
  xdim = 10,
  ydim = 10,
  meta_clustering = FALSE,
  meta_k = 40,
  seed_clustering = NULL,
  min_cells = 3,
  min_samples = NULL,
  normalize = FALSE,
  norm_factors = "TMM",
  verbose = TRUE,
```

```
  mi_reps = 10,
 imputation_method = c("km", "km_exp", "km_wei", "km_os", "rs", "mrl", "cc", "pmm"),
  BPPARAM = BiocParallel::SerialParam()
)
```

Arguments

d_input	Input data. Must be either: (i) a flowSet-class or list of flowFrame-classs, DataFrames, data.frames, or matrices as input (one flowFrame or list item per sample) (see prepareData); or (ii) a CATALYST daFrame (containing cluster labels in rowData; see vignette for an example).
experiment_info	data.frame, DataFrame, or {tbl_df} of experiment information, for example sample IDs and group IDs. Must contain a column named sample_id. See prepareData. (Not required when providing a CATALYST daFrame for d_input.)
marker_info	data.frame, DataFrame, or tbl_df of marker information for each column of data. This should contain columns named marker_name and marker_class. The columns contain: (i) marker names (and any other column names); and (ii) a factor indicating the marker class for each column (with entries "type", "state", or "none"). See prepareData. (Not required when providing a CATALYST daFrame for d_input.)
design	Design matrix, created with createDesignMatrix. See createDesignMatrix.
formula	Model formula object, created with createFormula. See createFormula.
contrast	Contrast matrix, created with createContrast. See createContrast.
analysis_type	Type of differential analysis to perform: differential abundance (DA) of cell populations. The only option at the moment is "DA". See testDA_censoredGLMM.
method_DA	Method to use for calculating differential abundance (DA) tests. Currently the only option is testDA_censoredGLMM. Default = testDA_censoredGLMM.
markers_to_test	(Optional) Logical vector specifying which markers to test for differential expression (from the set of markers stored in the assays of d_medians; for method testDS_limma or testDS_LMM). Default = all 'cell state' markers, which are identified by the logical vector id_state_markers stored in the meta-data of d_medians. See testDS_limma or testDS_LMM.
clustering_to_use	(Optional) Column name indicating which set of cluster labels to use for differential testing, when input data are provided as a CATALYST daFrame object containing multiple sets of cluster labels. (In this case, the metadata of the daFrame object is assumed to contain a data frame named cluster_codes; clustering_to_use is the column name of the selected column in cluster_codes. If clustering_to_use is provided, an identifier clustering_name to identify this column will also be saved in the metadata of the output object.) Default = NULL, in which case cluster labels stored in column named cluster_id in the rowData of the daFrame object are used.

cols_to_include
: Logical vector indicating which columns to include from the input data. Default = all columns. See prepareData.

subsampling
: Whether to use random subsampling to select an equal number of cells from each sample. Default = FALSE. See prepareData.

n_sub
: Number of cells to select from each sample by random subsampling, if subsampling = TRUE. Default = number of cells in smallest sample. See prepareData.

seed_sub
: Random seed for subsampling. Set to an integer value to generate reproducible results. Default = NULL. See prepareData.

transform
: Whether to apply 'arcsinh' transform. This may be set to FALSE if the input data has already been transformed. Default = TRUE. See transformData.

cofactor
: Cofactor parameter for 'arcsinh' transform. Default = 5, which is appropriate for mass cytometry (CyTOF) data. For fluorescence flow cytometry, cofactor = 150 is recommended instead. See transformData.

cols_clustering
: Columns to use for clustering. Default = NULL, in which case markers identified as 'cell type' markers (with entries "type") in the vector marker_class in the column meta-data of d_se will be used. See generateClusters.

xdim
: Horizontal length of grid for self-organizing map for FlowSOM clustering (number of clusters = xdim * ydim). Default = 10 (i.e. 100 clusters). See generateClusters.

ydim
: Vertical length of grid for self-organizing map for FlowSOM clustering (number of clusters = xdim * ydim). Default = 10 (i.e. 100 clusters). See generateClusters.

meta_clustering
: Whether to include FlowSOM 'meta-clustering' step. Default = FALSE. See generateClusters.

meta_k
: Number of meta-clusters for FlowSOM, if meta-clustering = TRUE. Default = 40. See generateClusters.

seed_clustering
: Random seed for clustering. Set to an integer value to generate reproducible results. Default = NULL. See generateClusters.

min_cells
: Filtering parameter. Default = 3. Clusters are kept for differential testing if they have at least min_cells cells in at least min_samples samples. See testDA_censoredGLMM.

min_samples
: Filtering parameter. Default = number of samples / 2, which is appropriate for two-group comparisons (of equal size). Clusters are kept for differential testing if they have at least min_cells cells in at least min_samples samples. See testDA_censoredGLMM.

normalize
: Whether to include optional normalization factors to adjust for composition effects. Default = FALSE. See testDA_censoredGLMM.

norm_factors
: Normalization factors to use, if normalize = TRUE. Default = "TMM", in which case normalization factors are calculated automatically using the 'trimmed mean of M-values' (TMM) method from the edgeR package. Alternatively, a vector of values can be provided (the values should multiply to 1). See testDA_censoredGLMM.

verbose
: Whether to print status messages during each step of the pipeline. Default = TRUE.

mi_reps Number of imputations in multiple imputation. Default = 10. See testDA_censoredGLMM.

imputation_method
Method to be used in the imputation step. One of km,km_exp,km_wei,km_os, rs, mrl, cc, pmm. See testDA_censoredGLMM.

BPPARAM Specification of parallelization option as one of BiocParallelParam if BiocParallel is available otherwise no parallelization is used. e.g. MulticoreParam-class(workers=2) for parallelization with two cores. Default is SerialParam-class() (no parallelization).

Details

This wrapper function runs the complete diffcyt analysis pipeline where the only difference is the analysis step which uses the functions from censcyt (which is currently only testDA_censoredGLMM).

For more details about the functions for the individual steps, see diffcyt, the diffcyt vignette, the censcyt package vignette and the function help pages. The following is a slightly adapted summary from diffcyt:

Running the individual functions may provide additional flexibility, especially for complex analyses.

The input data can be provided as a flowSet-class or a list of flowFrame-classs, DataFrames, data.frames, or matrices (one flowFrame or list item per sample). Alternatively, it is also possible to provide the input as a daFrame object from the CATALYST Bioconductor package (Chevrier, Crowell, Zanotelli et al., 2018). This can be useful when initial exploratory analyses and clustering have been performed using CATALYST; the daFrame object from CATALYST (containing cluster labels in the rowData) can then be provided directly to the censcyt functions for differential testing.

Minimum required arguments when not providing a flowSet-class or list of flowFrame-classs, DataFrames, data.frames, or matrices:

- d_input
- experiment_info
- marker_info
- either design or formula (depending on the differential testing method used)
- contrast
- analysis_type

Minimum required arguments when providing a CATALYST daFrame object:

- d_input
- either design or formula (depending on the differential testing method used)
- contrast
- analysis_type

Value

Returns a list containing the results object res, as well as the data objects d_se, d_counts, d_medians, d_medians_by_cluster_marker, and d_medians_by_sample_marker. (If a CATALYST daFrame object was used as input, the output list contains objects res, d_counts, and d_medians.)

Examples

```
# Function to create random data (one sample)
fcs_sim <- function(n = 2000, mean = 0, sd = 1, ncol = 10, cofactor = 5) {
  d <- matrix(sinh(rnorm(n*ncol, mean, sd)) * cofactor,ncol=ncol)
  for(i in seq_len(ncol)){
    d[seq(n/ncol*(i-1)+1,n/ncol*(i)),i] <- sinh(rnorm(n/ncol, mean+5, sd)) * cofactor
  }
  colnames(d) <- paste0("marker", sprintf("%02d", 1:ncol))
  d
}

# Create random data (without differential signal)
set.seed(123)
d_input <- lapply(1:50, function(i) fcs_sim())

# simulate survival time
d_surv <- simulate_singlecluster(50, formula(Y~Surv(X,I)))[c("X","I","TrVal")]

# Add differential abundance (DA) signal
for(i in 1:50){
  # number of cells in cluster 1
  n_da <- round(sqrt(2000*d_surv$TrVal[i]))*9
  # set to no expression
  tmpd <- matrix(sinh(rnorm(n_da*10, 0, 1)) * 5, ncol=10)
  # increase expresion for cluster 1
  tmpd[ ,1] <- sinh(rnorm(n_da, 5, 1)) * 5
  d_input[[i]][seq_len(n_da), ] <- tmpd
}

experiment_info <- data.frame(
  sample_id = factor(paste0("sample", 1:50)),
  survival_time = d_surv$X,
  event_indicator= d_surv$I,
  stringsAsFactors = FALSE
)

marker_info <- data.frame(
  channel_name = paste0("channel", sprintf("%03d", 1:10)),
  marker_name = paste0("marker", sprintf("%02d", 1:10)),
  marker_class = factor(c(rep("type", 10)),
                        levels = c("type", "state", "none")),
  stringsAsFactors = FALSE
)

# Create formula
da_formula <- createFormula(experiment_info, cols_fixed="survival_time",
                            cols_random = "sample_id",event_indicator = "event_indicator")
# Create contrast matrix
contrast <- diffcyt::createContrast(c(0, 1))

# Test for differential abundance (DA) of clusters
out_DA <- censcyt(d_input, experiment_info, marker_info,
```

```
                    formula = da_formula, contrast = contrast,
                    analysis_type = "DA", method_DA = "censcyt-DA-censored-GLMM",
                    seed_clustering = 123, verbose = FALSE, mi_reps = 3,
                    BPPARAM=BiocParallel::MulticoreParam(workers = 1),
                    imputation_method = "mrl",meta_clustering = TRUE, meta_k = 10)

# Display results for top DA clusters
diffcyt::topTable(out_DA, format_vals = TRUE)


# Plot heatmap for DA tests
diffcyt::plotHeatmap(out_DA, analysis_type = "DA")
```

`conditional_multiple_imputation` *Conditional multiple imputation*

Description

First two steps for multiple imputation for censored covariates. Returns regression fits in a list that can be combined using `pool()`.

Usage

```
conditional_multiple_imputation(
  data,
  formula,
  regression_type = c("lm", "glm", "glmer"),
  mi_reps = 10,
 imputation_method = c("km", "km_exp", "km_wei", "km_os", "rs", "mrl", "cc", "pmm"),
  weights = NULL,
  contrasts = NULL,
  family = "binomial",
  id = NULL,
  verbose = FALSE,
  n_obs_min = 2
)
```

Arguments

`data`	'data.frame'
`formula`	the formula for fitting the regression model with a special syntax for the censored covariate : e.g. 'y~Surv(x,I)' means 'y~x' with 'x' being censored and 'I' the event indicator (0=censored,1=observed).
`regression_type`	function. The regression type to be used, lm for linear regression, glm for general linear regression, glmer for generalized linear mixed-effects models. Default: lm

`mi_reps`	number of repetitions for multiple imputation. Default: 10
`imputation_method`	which method should be used in the imputation step. One of 'km','km_exp','km_wei','km_os', 'rs', 'mrl', 'cc', 'pmm'. See details. default = 'km'.
`weights`	Weights to be used in fitting the regression model. Default = NULL
`contrasts`	Contrast vector to be used in testing the regression model. Default = NULL
`family`	The family to be used in the regression model. Default = "binomial". Omitted if linear model is used.
`id`	name of column containing id of sample
`verbose`	Logical.
`n_obs_min`	minimum number of observed events needed. default = 2. if lower than this value will throw an error.

Details

Possible methods in 'imputation_method' are:

'km' Kaplan Meier imputation is similar to 'rs' (Risk set imputation) but the random draw is according to the survival function of the respective risk set.

'km_exp' The same as 'km' but if the largest value is censored the tail of the survival function is modeled as an exponential distribution where the rate parameter is obtained by fixing the distribution to the last observed value. See (Moeschberger and Klein, 1985).

'km_wei' The same as 'km' but if the largest value is censored the tail of the survival function is modeled as an weibull distribution where the parameters are obtained by MLE fitting on the whole data. See (Moeschberger and Klein, 1985).

'km_os' The same as 'km' but if the largest value is censored the tail of the survival function is modeled by order statistics. See (Moeschberger and Klein, 1985).

'rs' Risk Set imputation replaces the censored values with a random draw from the risk set of the respective censored value.

'mrl' Mean Residual Life (Conditional single imputation from Atem et al. 2017) is a multiple imputation procedure that bootstraps the data and imputes the censored values by replacing them with their respective mean residual life.

'cc' complete case (listwise deletion) analysis removes incomlete samples.

'pmm' predictive mean matching treats censored values as missing and uses predictive mean matching method from `mice`.

Value

A list with five elements:

'data' The input data frame

'betasMean' the mean regression coefficients

'betasVar' the variances of the mean regression coefficients

'metadata' a list of three elements:

'mi_reps' number of repetitions in multiple imputation

'betas' all regression coefficients

'vars' the variances of the regression coefficients

'fits' list with all regression fits

References

A Comparison of Several Methods of Estimating the Survival Function When There is Extreme Right Censoring (M. L. Moeschberger and John P. Klein, 1985)

Examples

```
# define association
lm_formula <- formula(Y ~ Surv(X,I) + Z)
# simulate data
data <- simulate_singlecluster(100, lm_formula, type = "lm", n_levels_fixeff=2)
# run fitting
cmi_out <- conditional_multiple_imputation(data,lm_formula)
# pool fits
comb_out <- mice::pool(cmi_out$fits)
# result
pvals <- summary(comb_out)$p.value
```

---

createFormula *Create model formula and corresponding data frame of variables*

Description

Create model formula and corresponding data frame of variables for model fitting

Usage

```
createFormula(
  experiment_info,
  cols_fixed = NULL,
  cols_random = NULL,
  event_indicator = NULL
)
```

Arguments

`experiment_info`

`data.frame`, `DataFrame`, or `tbl_df` of experiment information (which was also previously provided to `prepareData`). This should be a data frame containing all factors and covariates of interest; e.g. group IDs, block IDs, batch IDs, and continuous covariates.

`cols_fixed` Argument specifying columns of `experiment_info` to include as fixed effect terms in the model formula. This can be provided as a character vector of column names, a numeric vector of column indices, or a logical vector.

`cols_random` Argument specifying columns of `experiment_info` to include as random intercept terms in the model formula. This can be provided as a character vector of column names, a numeric vector of column indices, or a logical vector. Default = none.

`event_indicator` Argument specifying columns of `experiment_info` to include as event indicator for the censored covariate in the model formula. The censored covariate is assumed to be the first element of argument `cols_fixed`. This can be provided as a character vector of column names, a numeric vector of column indices, or a logical vector. Default = none.

Details

Creates a model formula and corresponding data frame of variables specifying the models to be fitted. Extends `createFormula` from `diffcyt`.

The output is a list containing the model formula and corresponding data frame of variables (one column per formula term). These can then be provided to differential testing functions that require a model formula, together with the main data object and contrast matrix.

The `experiment_info` input (which was also previously provided to `prepareData`) should be a data frame containing all factors and covariates of interest. For example, depending on the experimental design, this may include the following columns:

- group IDs (e.g. groups for differential testing)
- block IDs (e.g. patient IDs in a paired design; these may be included as either fixed effect or random effects)
- batch IDs (batch effects)
- continuous covariates
- sample IDs (e.g. to include random intercept terms for each sample, to account for overdispersion typically seen in high-dimensional cytometry data; this is known as an 'observation-level random effect' (OLRE); see see Nowicka et al., 2017, *F1000Research* for more details)

The arguments `cols_fixed` and `cols_random` specify the columns in `experiment_info` to include as fixed effect terms and random intercept terms respectively. These can be provided as character vectors of column names, numeric vectors of column indices, or logical vectors. The names for each formula term are taken from the column names of `experiment_info`. The argument `event_indicator` specifies the column in `experiment_info` as the event indicator ('0' represents censored and '1' represents observed) of the first element in `cols_fixed`.

Value

`formula`: Returns a list with three elements:

- `formula`: model formula
- `data`: data frame of variables corresponding to the model formula
- `random_terms`: TRUE if model formula contains any random effect terms

Examples

```
# model formula with censored variable
experiment_info <- data.frame(
  survival_time = rexp(8),
  sample_id = factor(paste0("sample", 1:8)),
  group_id = factor(rep(paste0("group", 1:2), each = 4)),
  observed = factor(rep(c(0,1),4)),
  patient_id = factor(rep(paste0("patient", 1:4), 2)),
  stringsAsFactors = FALSE
)
createFormula(experiment_info, cols_fixed = c("survival_time","group_id"),
  cols_random = c("sample_id", "patient_id"), event_indicator="observed")
```

---

simulate_multicluster *Simulate multicluster counts with time dependent association from a Dirichlet-Multinomial distribution*

Description

Simulate multicluster counts with time dependent association from a Dirichlet-Multinomial distribution

Usage

```
simulate_multicluster(
  counts = NULL,
  nr_diff = 2,
  nr_samples = NULL,
  alphas = NULL,
  theta = NULL,
  sizes = NULL,
  covariate = NULL,
  slope = NULL,
  group = NULL,
  group_slope = NULL,
  diff_cluster = FALSE,
  enforce_sum_alpha = FALSE,
  return_summarized_experiment = FALSE
)
```

Arguments

counts	the reference counts data set, either a matrix with rows as cluster and colums as samples or a SummarizedExperiment-class object as generated from calcCounts.
nr_diff	number of clusters where an association should be introduced. Has to be an even number.

`nr_samples` number of samples in output data. If NULL will set to same as input data.

`alphas` alpha parameter of Dirichlet-Multinomial distribution. If 'NULL' will be estimated from 'counts'.

`theta` correlation parameter. If 'NULL' will be estimated from 'counts'.

`sizes` total sizes for each sample

`covariate` covariates, one for each sample. Default Null means random draws from an exponential distribution with rate = 1.

`slope` negative double. Coefficients corresponding to the covariate for the DA clusters. One for each pair of DA clusters. To ensure correctness of the final distribution use only negative values. Alternatively can be a list of length 'nr_diff'/2, where each elements indicates the proportion of the cluster size at the maximum covariate relative to the mean. E.g. 0.1 means that the cluster proportion at the maximum covariate is 0.1 times smaller than the mean.

`group` either Null (no group effect), double between 0 and 1 (proportion of samples with group effect), integer (total number of samples with group effect), vector of 0 and 1 (indicating which samples have a group effect) or TRUE (effect with even group size).

`group_slope` regression coefficient of second covariate 'group'. If Null will choose a value automatically. Alternatively can be a list of length 'nr_diff'/2, where each elements indicates the proportion of the cluster size at the maximum covariate relative to the mean. E.g. 0.1 means that the cluster proportion at the maximum covariate is 0.1 times smaller than the mean.

`diff_cluster` Logical. Should the clusters be choosen random (TRUE) or according to a minimal distance of of mean cluster sizes (FALSE). Alternatively a list of length 'nr_diff' with each element a vector of length 2 indicating the paired clusters can be given. Default is FALSE.

`enforce_sum_alpha`
Logical. Should the total sum of alphas be kept constant to ensure randomness of non association clusters. The drawback is that one of the two paired clusters with an association will not follow a GLMM (binomial link function) exactly any more. Default is TRUE.

`return_summarized_experiment`
logical. Should the counts returned as a `SummarizedExperiment-class` object. Default is FALSE.

Value

returns a list with elements counts (either matrix or SummarizedExperiment object, depending on input), row_data (data per cluster: regression coefficients used), col_data (data per sample: covariates), alphas (matrix of alpha parameters used), theta (theta parameter), var_counts (covariance matrix of a DM distribution with the given alphas and sizes).

Examples

```
# without data reference:
alphas <- runif(20,10,100)
```

```
sizes <- runif(10,1e4,1e5)
output <- simulate_multicluster(alphas=alphas,sizes=sizes)
# counts:
counts <- output$counts

# with data reference:
# first simulate reference data set (normally this would be a real data set):
data <- t(dirmult::simPop(n=runif(10,1e4,1e5),theta=0.001)$data)
# then generate new data set based on original one but if DA clusters
output <- simulate_multicluster(data)

# specify number of differential clusters (has to be an even number):
output <- simulate_multicluster(alphas=alphas,sizes=sizes,nr_diff = 4)

# specify which clusters should be differential:
output <- simulate_multicluster(alphas=alphas,
                                sizes=sizes,
                                nr_diff = 4,
                                diff_cluster = list(c(2,9),c(6,7)))

# with second covariate (group):
output <- simulate_multicluster(alphas=alphas,sizes=sizes, group = TRUE)

# with second covariate (group), specify group proportion:
output <- simulate_multicluster(alphas=alphas,sizes=sizes, group = 0.5)

# with second covariate (group), specify id of group memberships for one group:
output <- simulate_multicluster(alphas=alphas,sizes=sizes, group = 3:7)
```

`simulate_singlecluster` *Simulation of data with a censored covariate*

Description

Function to simulate an association between a censored covariate and a predictor.

Usage

```
simulate_singlecluster(
  n,
  formula,
  type = c("lm", "glm", "glmer"),
  b = NULL,
  n_levels_fixeff = NULL,
  n_levels_raneff = NULL,
  weibull_params = list(X = list(shape = 0.5, scale = 0.25), C = list(shape = 1, scale
    = 0.25)),
  censoring_dependent_on_covariate = FALSE,
```

```
  weibull_params_covariate_dependent_censoring = list(shape = 1, scale = 0.1),
  error_variance = 0,
  variance_raneff = 0.5,
  transform_fn = "identity",
  verbose = FALSE
)
```

Arguments

n	number of samples
formula	the formula to specify the structure in the data. The censored variable should be written in the following format: 'Surv(X,I)', where 'X' is the observed value, and 'I' is the event indicator (1 if observed, 0 if censored). A full example is: 'Y ~ Surv(X,I) + Covariate + (1|Random_effect)'.
type	which regression type is used, one of 'lm', 'glm', 'glmer'. For the generalized linear models the response is binomial with a logistic link function. default = 'lm'.
b	the regression coefficients, either **NUll** will us 0 for the intercept and 1 for the remaining coefficients **a vector with regression coefficients** the length has to be (1 (intercept) + number of covariates (including the censored covariate))
n_levels_fixeff	The number of levels to use for each covariate, e.g. for two covariates: c(10,100). If NULL sets all to 2 (two groups).
n_levels_raneff	The number of levels to use for each random effect. If NULL sets to 'n' (observation level random effects).
weibull_params	The parameters for the distribution of the censored variable and the censoring time. Should be a list of lists, where the elements of the outer lists are 'X' the true value and 'C' the censoring time. The inner lists should have two keywords, 'shape' and 'scale', for the parameters of the Weibull distribution (See Weibull).
censoring_dependent_on_covariate	Logical. If censoring should depend on a covariate. The respective covariate needs to have only two levels ('n_level_fixeff'=2). Will use first covariate in formula.
weibull_params_covariate_dependent_censoring	list with two elements, shape and scale, representing the parameters of a weibull distribution for the second level of a covariate if 'censoring_dependent_on_covariate'=TRUE.
error_variance	positive double. Variance of additional gaussian noise to add in the linear sum of the predictors. For linear regression this is the only error added. Otherwise it should be set to zero. default = 0.
variance_raneff	positive double vector of the length of 'n_levels_raneff'. The variance of the gaussian distributed random effect covariates. default = 0.5.

transform_fn	function to transform censored covariate or one of 'identity' (no transformation), 'boxcox' (box-cox transformation), 'boxcox_positive' (box-cox transformation and translation to all positive values), 'log_positive' (log transformation and translation to all positive values). The transformation is applied before the response is modeled. default = 'identity'.
verbose	verbose

Value

tibble

Examples

```
# single differential cluster
 glmer_formula <- formula(Y ~ Surv(X,I) + Z + (1|R))
 simulate_singlecluster(100, glmer_formula, type = "glmer")
```

---

testDA_censoredGLMM *Test for differential abundance: method 'censcyt-DA-censored-GLMM'*

Description

Calculate tests for differential abundance of cell populations using method 'censcyt-DA-censored-GLMM'

Usage

```
testDA_censoredGLMM(
  d_counts,
  formula,
  contrast,
  mi_reps = 10,
 imputation_method = c("km", "km_exp", "km_wei", "km_os", "rs", "mrl", "cc", "pmm"),
  min_cells = 3,
  min_samples = NULL,
  normalize = FALSE,
  norm_factors = "TMM",
  BPPARAM = BiocParallel::SerialParam(),
  verbose = FALSE
)
```

Arguments

d_counts	SummarizedExperiment object containing cluster cell counts, from calcCounts.
formula	Model formula object, see testDA_GLMM and for more details createFormula. Be aware of the special format required for the censored covariate: instead of just the covariate name (e.g. 'X') the columnname of the data being an event indicator (e.g. 'I', with 'I' = 1 if 'X' is observed and 'I' = 0 if 'X' is censored,) needs to specified as well. The notation in the formula is then 'Surv(X,I)'.
contrast	Contrast matrix, created with createContrast. See createContrast for details.
mi_reps	number of imputations in multiple imputation. default = 10.
imputation_method	which method should be used in the imputation step. One of 'km','km_exp','km_wei','km_os', 'rs', 'mrl', 'cc', 'pmm'. See details. default = 'km'.
min_cells	Filtering parameter. Default = 3. Clusters are kept for differential testing if they have at least min_cells cells in at least min_samples samples.
min_samples	Filtering parameter. Default = number of samples / 2, which is appropriate for two-group comparisons (of equal size). Clusters are kept for differential testing if they have at least min_cells cells in at least min_samples samples.
normalize	Whether to include optional normalization factors to adjust for composition effects (see details). Default = FALSE.
norm_factors	Normalization factors to use, if normalize = TRUE. Default = "TMM", in which case normalization factors are calculated automatically using the 'trimmed mean of M-values' (TMM) method from the edgeR package. Alternatively, a vector of values can be provided (the values should multiply to 1).
BPPARAM	specify parallelization option as one of BiocParallelParam if 'BiocParallel' is available otherwise no parallelization. e.g. MulticoreParam-class(workers=2) for parallelization with two cores. Default is SerialParam-class() (no parallelization).
verbose	Logical.

Details

Calculates tests for differential abundance of clusters, using generalized linear mixed models (GLMMs) where a covariate is subject to right censoring.

The same underlying testing as described in testDA_GLMM is applied here. The main difference is that multiple imputation is used to handle a censored covariate. In short, multiple imputation consists of three steps: imputation, analysis and pooling. In the imputation step multiple complete data sets are generated by imputation. The imputed data is then analysed in the second step and the results are combined in the third step. See also pool. The imputation in the first step is specific for censored data in contrast to the 'normal' use of multiple imputation where data is missing. Alternatively the samples with censored data can be removed (complete case analysis) or the censored values can be treated as missing (predictive mean matching).

Possible imputation methods in argument 'imputation_method' are:

'km' Kaplan Meier imputation is similar to 'rs' (Risk set imputation) but the random draw is according to the survival function of the respective risk set. The largest value is treated as observed to obtain a complete survival function. (Taylor et al. 2002)

'km_exp' The same as 'km' but if the largest value is censored the tail of the survival function is modeled as an exponential distribution where the rate parameter is obtained by fixing the distribution to the last observed value. See (Moeschberger and Klein, 1985).

'km_wei' The same as 'km' but if the largest value is censored the tail of the survival function is modeled as an weibull distribution where the parameters are obtained by MLE fitting on the whole data. See (Moeschberger and Klein, 1985).

'km_os' The same as 'km' but if the largest value is censored the tail of the survival function is modeled by order statistics. See (Moeschberger and Klein, 1985).

'rs' Risk Set imputation replaces the censored values with a random draw from the risk set of the respective censored value. (Taylor et al. 2002)

'mrl' Mean Residual Life (Conditional multiple imputation, See Atem et al. 2017) is a multiple imputation procedure that bootstraps the data and imputes the censored values by replacing them with their respective mean residual life.

'cc' complete case (listwise deletion) analysis removes incomlete samples.

'pmm' predictive mean matching treats censored values as missing and uses predictive mean matching from `mice`.

Value

Returns a new `SummarizedExperiment` object, with differential test results stored in the `rowData` slot. Results include raw p-values (`p_val`) and adjusted p-values (`p_adj`), which can be used to rank clusters by evidence for differential abundance. The results can be accessed with the `rowData` accessor function.

References

A Comparison of Several Methods of Estimating the Survival Function When There is Extreme Right Censoring (M. L. Moeschberger and John P. Klein, 1985)

Improved conditional imputation for linear regression with a randomly censored predictor (Atem et al. 2017)

Survival estimation and testing via multiple imputation (Taylor et al. 2002)

Examples

```
# create small data set with 2 differential clusters with 10 samples.
d_counts <- simulate_multicluster(alphas = runif(10,1e4,1e5),
                                  sizes = runif(10,1e4,1e5),
                                  nr_diff = 2,
                                  group=2,
                                  return_summarized_experiment = TRUE)$counts

# extract covariates data.frame
experiment_info <- SummarizedExperiment::colData(d_counts)
# add censoring
```

```
experiment_info$status <- sample(c(0,1),size=10,replace = TRUE,prob = c(0.3,0.7))
experiment_info$covariate[experiment_info$status == 0] <-
  runif(10-sum(experiment_info$status),
        min=0,
        max=experiment_info$covariate[experiment_info$status == 0])

# create model formula object
da_formula <- createFormula(experiment_info,
                            cols_fixed = c("covariate", "group_covariate"),
                            cols_random = "sample",event_indicator = "status")

# create contrast matrix
contrast <- diffcyt::createContrast(c(0, 1, 0))

# run testing with imputation method 'km'
outs <- testDA_censoredGLMM(d_counts = d_counts, formula = da_formula,
                            contrast = contrast, mi_reps = 2, imputation_method = "km")
diffcyt::topTable(outs)
# differential clusters:
which(!is.na(SummarizedExperiment::rowData(d_counts)$paired))
```

Index

BiocParallelParam, *5*, *16*

calcCounts, *11*, *16*
censcyt, 2
censcyt-package *(*censcyt*)*, 2
conditional_multiple_imputation, 7
createContrast, *3*, *16*
createDesignMatrix, *3*
createFormula, *3*, 9, *10*, *16*

DataFrame, *3*, *5*
diffcyt, *5*

generateClusters, *4*

mice, *8*, *17*

pool, *7*, *16*
prepareData, *3, 4*, *9, 10*

rowData, *17*

simulate_multicluster, 11
simulate_singlecluster, 13
SummarizedExperiment, *16, 17*

tbl_df, *3*
testDA_censoredGLMM, *3–5*, 15
testDA_GLMM, *16*
testDS_limma, *3*
testDS_LMM, *3*
tibble, *15*
transformData, *4*

Weibull, *14*