Package ‘infinityFlow’

July 7, 2025

Title Augmenting Massively Parallel Cytometry Experiments Using Multivariate Non-Linear Regressions

Version 1.19.0

Description Pipeline to analyze and merge data files produced by BioLegend's LEGEND-Screen or BD Human Cell Surface Marker Screening Panel (BD Lyoplates).

Depends R (>= 4.0.0), flowCore

License GPL-3

Encoding UTF-8

LazyData false

Imports stats, grDevices, utils, graphics, pbapply, matlab, png, raster, grid, uwot, gtools, Biobase, generics, parallel, methods, xgboost

Suggests knitr, rmarkdown, keras, tensorflow, glmnetUtils, e1071

VignetteBuilder knitr

RoxygenNote 7.3.0

biocViews Software, FlowCytometry, CellBasedAssays, SingleCell, Proteomics

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

git_branch devel

git_last_commit 166c477

git_last_commit_date 2025-04-15

Repository Bioconductor 3.22

Date/Publication 2025-07-07

Author Etienne Becht [cre, aut]

Maintainer Etienne Becht <`etienne.becht@protonmail.com`>

Contents

`fitter_glmnet` *Wrapper to glmnet. Defined separetely to avoid passing too many objects in parLapplyLB*

Description

Wrapper to glmnet. Defined separetely to avoid passing too many objects in parLapplyLB

Usage

```
fitter_glmnet(x = NULL, params = NULL)
```

Arguments

`x`	passed from fit_regressions
`params`	passed from fit_regressions

Value

A list with two elements: predictions and a fitted model

Examples

```
fitter_glmnet()
```

`fitter_linear` *Wrapper to linear model training. Defined separetely to avoid passing too many objects in parLapplyLB*

Description

Wrapper to linear model training. Defined separetely to avoid passing too many objects in parLapplyLB

Usage

```
fitter_linear(x = NULL, params = NULL)
```

Arguments

`x`	passed from fit_regressions
`params`	passed from fit_regressions

Value

A list with two elements: predictions and a fitted model

Examples

```
fitter_linear()
```

`fitter_nn` *Wrapper to Neural Network training. Defined separetely to avoid passing too many objects in parLapplyLB*

Description

Wrapper to Neural Network training. Defined separetely to avoid passing too many objects in parLapplyLB

Usage

```
fitter_nn(x, params)
```

Arguments

`x`	passed from fit_regressions. Defines model architecture
`params`	passed from fit_regressions

Value

A list with two elements: predictions and a fitted model

Examples

```
fitter_xgboost()
```

`fitter_svm` *Wrapper to SVM training. Defined separetely to avoid passing too many objects in parLapplyLB*

Description

Wrapper to SVM training. Defined separetely to avoid passing too many objects in parLapplyLB

Usage

```
fitter_svm(x = NULL, params = NULL)
```

Arguments

`x`	passed from fit_regressions
`params`	passed from fit_regressions

Value

A list with two elements: predictions and a fitted model

Examples

```
fitter_svm()
```

`fitter_xgboost` *Wrapper to XGBoost training. Defined separetely to avoid passing too many objects in parLapplyLB*

Description

Wrapper to XGBoost training. Defined separetely to avoid passing too many objects in parLapplyLB

Usage

```
fitter_xgboost(x = NULL, params = NULL)
```

Arguments

`x`	passed from fit_regressions
`params`	passed from fit_regressions

Value

A list with two elements: predictions and a fitted model

Examples

```
fitter_xgboost()
```

`infinity_flow` *Wrapper to the Infinity Flow pipeline*

Description

Wrapper to the Infinity Flow pipeline

Usage

```
infinity_flow(
  path_to_fcs,
  path_to_output,
  path_to_intermediary_results = tempdir(),
  backbone_selection_file = NULL,
  annotation = NULL,
  isotype = NULL,
  input_events_downsampling = Inf,
  prediction_events_downsampling = 1000,
  cores = 1L,
  your_random_seed = 123,
  verbose = TRUE,
  extra_args_read_FCS = list(emptyValue = FALSE, truncate_max_range = FALSE,
    ignore.text.offset = TRUE),
  regression_functions = list(XGBoost = fitter_xgboost),
  extra_args_regression_params = list(list(nrounds = 500, eta = 0.05)),
  extra_args_UMAP = list(n_neighbors = 15L, min_dist = 0.2, metric = "euclidean", verbose
    = verbose, n_epochs = 1000L, n_threads = cores, n_sgd_threads = cores),
  extra_args_export = list(FCS_export = c("split", "concatenated", "none")[1], CSV_export
    = FALSE),
  extra_args_correct_background = list(FCS_export = c("split", "concatenated",
    "none")[1], CSV_export = FALSE),
  extra_args_plotting = list(chop_quantiles = 0.005),
  neural_networks_seed = NULL
)
```

Arguments

`path_to_fcs`	Path to the input directory where input FCS files are stored (one file per well). Will look for FCS files recursively in that directory.
`path_to_output`	Path to the output directory where final results will be stored
`path_to_intermediary_results`	Path to results to store temporary data. If left blank, will default to a temporary directory. It may be useful to store the intermediary results to further explore the data, tweak the pipeline or to resume computations.
`backbone_selection_file`	If that argument is missing and R is run interactively, the user will be prompted to state whether each channel in the FCS file should be considered backbone measurement, exploratory measurement or ignored. Otherwise, the user should run `select_backbone_and_exploratory_markers` in an interactive R session, save its output using *write.csv(row.names=FALSE)* and set this *backbone_selection_file* parameter to the path of the saved output.
`annotation`	Named character vector. Elements should be the targets of the exploratory antibodies, names should be the name of the FCS file where that exploratory antibody was measured.
`isotype`	Named character vector. Elements should be the isotype used in each of the well and that (e.g. IgG2). The corresponding isotype should be present in *annotation* (e.g. Isotype_IgG2, with this capitalization exactly). Autofluorescence measurements should be listed here as "Blank"

`input_events_downsampling`	How many event should be kept per input FCS file. Default to no downsampling. In any case, half of the events will be used to train regression models and half to test the performance. Predictions will be made only on events from the test set, and downsampled according to prediction_events_downsampling.
`prediction_events_downsampling`	How many event should be kept per input FCS file to output prediction for. Default to 1000.
`cores`	Number of cores to use for parallel computing. Defaults to 1 (no parallel computing)
`your_random_seed`	Deprecated: was used to set a seed for computationally reproducible results but is not allowed by Bioconductor. Please set a random seed yourself using set.seed(somenumber) if you desire computionally-reproducible results.
`verbose`	Whether to print information about progress
`extra_args_read_FCS`	list of named arguments to pass to flowCore:read.FCS. Defaults to list(emptyValue=FALSE,truncate_ which in our experience avoided issues with data loading.
`regression_functions`	named list of fitter_* functions (see ls("package:infinityFlow") for the complete list). The names should be desired names for the different models. Each object of the list will correspond to a machine learning model to train. Defaults to list(XGBoost = fitter_xgboost).
`extra_args_regression_params`	list of lists the same length as the regression_functions argument. Each element should be a named list, that will be passed as named arguments to the corresponding fitter_ function. Defaults to list(list(nrounds = 500, eta = 0.05)).
`extra_args_UMAP`	list of named arguments to pass to uwot:umap. Defaults to list(n_neighbors=15L,min_dist=0.2,metric
`extra_args_export`	Whether raw imputed data should be exported. Possible values are list(FCS_export = "split") to export one FCS file per input well, list(FCS_export = "concatenated") to export a single concatenated FCS file containing all the dataset, list(FCS_export = "csv") for a single CSV file containing all the dataset. You can export multiple modalities by using for instance extra_args_export = list(FCS_export = c("split", "concatenated", "csv"))
`extra_args_correct_background`	Whether background-corrected imputed data should be exported. Possible values are list(FCS_export = "split") to export one FCS file per input well, list(FCS_export = "concatenated") to export a single concatenated FCS file containing all the dataset, list(FCS_export = "csv") for a single CSV file containing all the dataset. You can export multiple modalities by using for instance extra_args_export = list(FCS_export = c("split", "concatenated", "csv"))
`extra_args_plotting`	list of named arguments to pass to plot_results. Defaults to list(chop_quantiles=0.005) which removes the top 0.05% and bottom 0.05% of the scale for each marker when mapping color palettes to intensities.
`neural_networks_seed`	Seed for computationally reproducible results when using neural networks (in additional to the other sources of stochasticity - sampling - that are made reproducible by the your_random_seed argument.

Value

Raw and background-corrected imputed expression data for every Infinity antibody

---

`select_backbone_and_exploratory_markers` *For each parameter in the FCS files, interactively prompts whether it is part of the Backbone, the Infinity (exploratory) markers or should be ignored.*

Description

This function will load the first of the input FCS files and extract the measured parameters as well as their labels. For each of these, it will ask the user whether it is part of the backbone measurements (which will be used as a predictor variable in regressions models), Infinity (exploratory) measurements (usually PE-conjugated or APC-conjugated, used as dependent/target variable in regressions) or discarded (e.g. for parameter such as Time, Sample IDs, Event number IDs, ...).

Usage

```
select_backbone_and_exploratory_markers(files)
```

Arguments

`files`	character vector of paths to FCS files

Value

A data.frame

Examples

```
data(steady_state_lung)
dir <- tempdir()
fcs_tmp <- file.path(dir, "tmp.fcs")
library(flowCore)
write.FCS(steady_state_lung[[1]], file <- fcs_tmp)
if(interactive()){
    select_backbone_and_exploratory_markers(fcs_tmp)
}
```

`steady_state_lung` *Subset of a massively parallel cytometry experiment of mouse lung single cells*

Description

Subset of a massively parallel cytometry experiment of mouse lung single cells

Usage

```
data(steady_state_lung)
```

Format

a flowSet containing 10 flowFrames (thus corresponding to 10 FCS files)

Source

`https://flowrepository.org/id/FR-FCM-Z2LP`

`steady_state_lung_annotation` *Target and isotypes annotation for the data object infinityFlow::steady_state_lnug*

Description

Target and isotypes annotation for the data object infinityFlow::steady_state_lnug

Usage

```
data(steady_state_lung_annotation)
```

Format

a data.frame specifying the Infinity antibody targets and isotypes for each flowFrame of the steady_state_lung flowSet

---

`steady_state_lung_backbone_specification`

Backbone and Infinity antibodies specification for the data object infinityFlow::steady_state_lnug

---

Description

Backbone and Infinity antibodies specification for the data object infinityFlow::steady_state_lnug

Usage

```
data(steady_state_lung_backbone_specification)
```

Format

a data.frame specifying the Infinity antibody targets and isotypes for each flowFrame of the steady_state_lung flowSet

Index