Package: polyICSFlow (via r-universe)

June 11, 2026

Type Package

Title Identifying the Frequency of Polyfunctional Antigen-Specific T cells in ICS Flow Cytometry Data

Version 0.99.3

Imports flowCore, flowWorkspace, openCyto, FlowSOM, ComplexHeatmap, SummarizedExperiment, cowplot, dplyr, forcats, ggtext, rlang, scales, stringr, tidyselect, tidyr, magrittr, ggplot2, methods

Description polyICSFlow systematically identifies all cytokine combinations detected in an intracellular cytokine staining (ICS) flow cytometry assay and quantifies polyfunctional antigen-specific responses to single or multiple antigens. The package also offers versatile plotting options for efficient data exploration and visualization of cytokine co-expression profiles.

URL https://github.com/SoegaardLab/polyICSFlow

BugReports https://github.com/SoegaardLab/polyICSFlow/issues

biocViews FlowCytometry, Immunology, Software, SingleCell

License GPL (>=2)

Encoding UTF-8

LazyData false

RoxygenNote 7.3.3

Suggests knitr, rmarkdown, testthat (>= 3.0.0), patchwork, ggh4x, CytoExploreR

Config/testthat/edition 3

VignetteBuilder knitr

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

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

Date/Publication 2026-06-11 20:00:30 UTC

RemoteUrl https://github.com/BiocStaging/polyICSFlow

RemoteRef HEAD

RemoteSha 036ed01f8651ae6c37bb63dbdc88d2cafe6cee74

Contents

`assignMarkerCombinations`
Assing each cell to a combination of gated markers.

Description

Assing each cell to a combination of gated markers.

Usage

```
assignMarkerCombinations(data_markerPos, mode = c("simple", "exhaustive"))
```

Arguments

`data_markerPos`	A data.frame where each row is a cell and each column is a gated marker describing whether a cell is included in the gate, as constructed by the `getMarkerPositivity` function.
`mode`	Character string specifying the level of detail in the polyfunctionality assignment. If `simple`, the output includes only `MarkerComb` (exact marker combinations), `MarkerCount` (number of functions), and `Functionality` (mono/polyfunctional). If `exhaustive`, the output additionally contains all "atleast_n" columns, which are logical describing whether cells have at least 1, 2, 3, ... n markers in any combination. Default = `simple`.

Value

The input data.frame with three extra columns describing

- `MarkerComb`: A factor describing which marker combination a specific cell has. Describes the exact combinations, which are mutually exclusive (e.g., 4 markers $\rightarrow$ 2 = 16 combinations).

- `MarkerCount`: An integer describing the number of functions of the cell (count of positive markers).
- `Functionality`: A factor grouping `MarkerCount)` into three categories; No cytokines (`MarkerCount == 0`), Monofunctional (`MarkerCount == 1`), or Polyfunctional (`MarkerCount > 1`).

See Also

`getMarkerPositivity`, `calcPolyfunctionality`

Examples

```
# Define path to data
path_data <- system.file("extdata", package = "polyICSFlow")

# Read fcs files as flowSet
fs <- flowCore::read.flowSet(path = path_data,
                             pattern = ".fcs",
                             truncate_max_range = FALSE,
                             transformation = FALSE)

# Create GatingSet
gs <- flowWorkspace::GatingSet(fs)

# Apply GatingTemplate
gt <- openCyto::gatingTemplate(file.path(path_data, "gatingTemp_cytokines.csv"))
openCyto::gt_gating(x = gt, y = gs)

# Get marker positivity per cell
df_markerPos <- getMarkerPositivity(x = gs,
                                    gate_names = c("IFN+","TNFa+","IL2+","CD107a+"))

# Assign marker combinations to each cell based on marker positivity
# Simple mode
df_markerComb <- assignMarkerCombinations(data_markerPos = df_markerPos,
                                          mode = "simple")
# Exhaustive mode
df_markerComb <- assignMarkerCombinations(data_markerPos = df_markerPos,
                                          mode = "exhaustive")
```

`calcPolyfunctionality` *Calculate the frequency of polyfunctional cells*

Description

Calculate the frequency of cells positive for each marker combination within a defined population and subtract the corresponding background. Negative values are set to zero.

Usage

```
calcPolyfunctionality(
  x,
  md_cols,
  pop_col = "cell_type",
  resolution = "MarkerComb",
  condition_col = "Stimulation",
  background_val = "unstimulated"
)
```

Arguments

`x`	A dataframe generated by the `assignMarkerCombinations` function, containing marker combination data along with associated metadata.
`md_cols`	A character vector specifying the names of metadata columns in `x`. These columns provide contextual information (e.g., participant ID, sample ID, time point).
`pop_col`	A string giving the name of the column in `x` that contains the population of interest. The function calculates the percentage of positive cells for each marker combination within this population. Can be obtained from any source, e.g., annotated metacluster IDs from FlowSOM clustering or cell labels from manual gating (see `GetFlowJoLabels`).
`resolution`	A string specifying the column in `x` that defines polyfunctionality, e.g., `"MarkerComb"`, `"MarkerCount"`, or `"Functionality"`. Default is `"MarkerComb"`. Can also refer to any of the logical columns generated if `mode = "exhaustive"` was used in `assignMarkerCombinations`, or to custom resolutions created by combining outputs from `assignMarkerCombinations`. For example, `atleast_IFN+TNFa+ == TRUE & MarkerCount == 3` refers to cells with three functions positive for at least IFN and TNFa. Users must first create the logical column in `x` (e.g., using `dplyr::mutate()`) before specifying it here.
`condition_col`	A string giving the name of the column in `x` containing the stimulation condition (e.g., antigen-stimulated, unstimulated). Default = `"Stimulation"`.
`background_val`	A value from `condition_col` indicating the background condition (e.g., "unstimulated"). This is used as the reference for stimulation conditions. Default = `"unstimulated"`.

Value

A dataframe similar to `x`, with the addition of four new columns:

- `n_cells_pos`: Number of positive cells
- `n_cells_tot`: Number of total cells
- `perc_pos`: Percentage positive cells, i.e. `(n_cells_pos/n_cells_tot)*100`
- `perc_pos_backgroundSubtracted`: Percentage positive cells with corresponding background subtracted. Negative values are set to zero.

See Also

getMarkerPositivity, assignMarkerCombinations

Examples

```
# Define path to data
path_data <- system.file("extdata", package = "polyICSFlow")

# Read fcs files as flowSet
fs <- flowCore::read.flowSet(path = path_data,
                             pattern = ".fcs",
                             truncate_max_range = FALSE,
                             transformation = FALSE)

# Create GatingSet
gs <- flowWorkspace::GatingSet(fs)

# Apply GatingTemplate
gt <- openCyto::gatingTemplate(file.path(path_data,
                                         "gatingTemp_cytokines.csv"))
openCyto::gt_gating(x = gt, y = gs)

# Get marker positivity per cell
df_markerPos <- getMarkerPositivity(x = gs,
                                    gate_names = c("IFN+","TNFa+","IL2+","CD107a+"))

# Assign marker combinations to each cell based on marker positivity
df_markerComb <- assignMarkerCombinations(data_markerPos = df_markerPos,
                                          mode = "exhaustive")

# Read metadata and cell labels
df_md <- read.csv(file.path(path_data, "metadata.csv"))
metacluster_labels <- readRDS(file.path(path_data, "cell_labels.rds"))

# Add metadata and cell labels to dataframe
df <- df_markerComb %>%
    dplyr::mutate(ID = df_md$ID,
                  Stimulation = df_md$Stimulation,
                  cell_type = metacluster_labels)

# Calculate by Marker Combination
result <- calcPolyfunctionality(x = df,
                                md_cols = c("ID"),
                                pop_col = "cell_type",
                                resolution = "MarkerComb",
                                condition_col = "Stimulation",
                                background_val = "Unstim")

# Calculate by Marker Count
result <- calcPolyfunctionality(x = df,
                                md_cols = c("ID"),
                                pop_col = "cell_type",
```

```
                                resolution = "MarkerCount",
                                condition_col = "Stimulation",
                                background_val = "Unstim")

# Calculate by Functionality
result <- calcPolyfunctionality(x = df,
                                md_cols = c("ID"),
                                pop_col = "cell_type",
                                resolution = "Functionality",
                                condition_col = "Stimulation",
                                background_val = "Unstim")

# Calculate by logical at least n
result <- calcPolyfunctionality(x = df,
                                md_cols = c("ID"),
                                pop_col = "cell_type",
                                resolution = "atleast_2_IFN+TNFa+",
                                condition_col = "Stimulation",
                                background_val = "Unstim")

# Calculate by logical Polyfunctional, at least marker
result <- calcPolyfunctionality(x = df,
                                md_cols = c("ID"),
                                pop_col = "cell_type",
                                resolution = "Polyfunctional_atleast_IFN+",
                                condition_col = "Stimulation",
                                background_val = "Unstim")

# Calculate by custom resolution
df2 <- df %>%
  dplyr::mutate(MarkerCountAbove3_atleast_IFN_TNFa = MarkerCount == 3 &
                  .data[["atleast_2_IFN+TNFa+"]])

result <- calcPolyfunctionality(x = df2,
                                md_cols = c("ID"),
                                pop_col = "cell_type",
                                resolution = "MarkerCountAbove3_atleast_IFN_TNFa",
                                condition_col = "Stimulation",
                                background_val = "Unstim")
```

`extractCellCounts` *Get number of cells positive for each distinct marker combination*

Description

Get number of cells in your dataset positive for each distinct marker combination (MarkerComb), number of functions (MarkerCount), and Functionality.

Usage

```
extractCellCounts(
  data_markerComb,
  cell_subset_params = list(cellTypes = NULL, cellTypes_plot = NULL)
)
```

Arguments

`data_markerComb`
: A dataframe generated by the `assignMarkerCombinations` function, containing marker combination data.

`cell_subset_params`
: List with parameters describing what subset of cells (cellTypes_plot) should be counted, from a vector of cell types (cellTypes). Default = `list(cellTypes = NULL, cellTypes_plot = NULL)`, i.e. all cells in the dataset are included in the cell count.

Value

A grouped tibble with cell counts of the input data grouped by `"MarkerComb"`, `"MarkerCount"`, and `"Functionality"`.

Examples

```
# Define path to data
path_data <- system.file("extdata", package = "polyICSFlow")

# Read fcs files as flowSet
fs <- flowCore::read.flowSet(path = path_data,
                             pattern = ".fcs",
                             truncate_max_range = FALSE,
                             transformation = FALSE)

# Create GatingSet
gs <- flowWorkspace::GatingSet(fs)

# Apply GatingTemplate
gt <- openCyto::gatingTemplate(file.path(path_data, "gatingTemp_cytokines.csv"))
openCyto::gt_gating(x = gt, y = gs)

# Get marker positivity per cell
df_markerPos <- getMarkerPositivity(x = gs,
                                    gate_names = c("IFN+","TNFa+","IL2+","CD107a+"))

# Assign marker combinations to each cell based on marker positivity
df_markerComb <- assignMarkerCombinations(data_markerPos = df_markerPos,
                                          mode = "exhaustive")

# Get cell counts from all data
cell_counts <- extractCellCounts(data_markerComb = df_markerComb)
```

```
# Get cell counts from subset of data (metacluster 2 and 4)
metacluster_labels <- readRDS(file.path(path_data, "cell_labels.rds"))
cell_counts <- extractCellCounts(data_markerComb = df_markerComb,
                                 cell_subset_params = list(cellTypes = metacluster_labels,
                                                           cellTypes_plot = c(2,4)))
```

`getMarkerPositivity` *Create dataframe of marker positivity*

Description

Create a dataframe describing whether each cell is included by the gates where each row is a cell and each column is a marker.

Usage

```
getMarkerPositivity(x, gate_names = NULL)
```

Arguments

`x`	An object of class `GatingHierarchy` or `GatingSet` with an applied `gatingTemplate`.
`gate_names`	The names of the cytokine gates. If `NULL` (default), all leaf nodes will be used.

Value

A data.frame where each row is a cell and each column is a gated marker which can be used as input for the `assignMarkerCombinations` function.

References

This code is strongly based on the `GetFlowJoLabels` function.

See Also

`GatingSet`, `gatingTemplate`, `gt_gating`, `assignMarkerCombinations`, `calcPolyfunctionality`

Examples

```
# Define path to data
path_data <- system.file("extdata", package = "polyICSFlow")

# Read fcs files as flowSet
fs <- flowCore::read.flowSet(path = path_data,
                             pattern = ".fcs",
                             truncate_max_range = FALSE,
                             transformation = FALSE)

# Create GatingSet
```

```
gs <- flowWorkspace::GatingSet(fs)

# Apply GatingTemplate
gt <- openCyto::gatingTemplate(file.path(path_data, "gatingTemp_cytokines.csv"))
openCyto::gt_gating(x = gt, y = gs)

# Get marker positivity per cell
df_markerPos <- getMarkerPositivity(x = gs,
                                    gate_names = c("IFN+","TNFa+","IL2+","CD107a+"))
```

`plotCellCountsAtLeast` *Plot counts of cells expressing at least 1, 2, ..., n markers.*

Description

Plot a stacked bar plot of cell counts by marker combination, faceted by the minimum number of markers co-expressed $(1, 2, , n)$. Within each facet, the x-axis shows all marker or marker-combination categories corresponding to that level (e.g., single markers for 1, two-marker combinations for 2, etc.).

Usage

```
plotCellCountsAtLeast(
  data_markerComb,
  cell_subset_params = list(cellTypes = NULL, cellTypes_plot = NULL),
  palette_comb = NULL
)
```

Arguments

`data_markerComb`
: A dataframe generated by the `assignMarkerCombinations` function, containing marker combination data.

`cell_subset_params`
: List with parameters describing what subset of cells (cellTypes_plot) should be plotted, from a vector of cell types (cellTypes). Default = `list(cellTypes = NULL, cellTypes_plot = NULL)`, i.e. all cells in the dataset are plotted.

`palette_comb`
: A named character vector mapping marker combination names (as produced by `assignMarkerCombinations`) to colors. If `NULL` (default), a qualitative palette is generated with `colorRampPalette` and `CATALYST::.cluster_cols`.

Value

A ggplot.

See Also

plotCellCountsExactly

Examples

```
# Define path to data
path_data <- system.file("extdata", package = "polyICSFlow")

# Read fcs files as flowSet
fs <- flowCore::read.flowSet(path = path_data,
                             pattern = ".fcs",
                             truncate_max_range = FALSE,
                             transformation = FALSE)

# Create GatingSet
gs <- flowWorkspace::GatingSet(fs)

# Apply GatingTemplate
gt <- openCyto::gatingTemplate(file.path(path_data, "gatingTemp_cytokines.csv"))
openCyto::gt_gating(x = gt, y = gs)

# Get marker positivity per cell
df_markerPos <- getMarkerPositivity(x = gs,
                                    gate_names = c("IFN+","TNFa+","IL2+","CD107a+"))

# Assign marker combinations to each cell based on marker positivity
df_markerComb <- assignMarkerCombinations(data_markerPos = df_markerPos,
                                          mode = "simple")

# Plot all cells
plotCellCountsAtLeast(data_markerComb = df_markerComb)

# Plot only subset of data (metacluster 2 and 4)
metacluster_labels <- readRDS(file.path(path_data, "cell_labels.rds"))
plotCellCountsAtLeast(data_markerComb = df_markerComb,
                      cell_subset_params = list(cellTypes = metacluster_labels,
                                                cellTypes_plot = c(2,4)))
```

---

plotCellCountsExactly

Plot counts of cells expressing exactly 1, 2, ..., n markers.

Description

Plot a stacked bar plot of cell counts within each marker combination for cells positive for exactly 1, 2, ..., n markers.

Usage

```
plotCellCountsExactly(
  data_markerComb,
  cell_subset_params = list(cellTypes = NULL, cellTypes_plot = NULL),
  palette_comb = NULL
)
```

Arguments

data_markerComb	A dataframe generated by the assignMarkerCombinations function, containing marker combination data.
cell_subset_params	List with parameters describing what subset of cells (cellTypes_plot) should be plotted, from a vector of cell types (cellTypes). Default = list(cellTypes = NULL, cellTypes_plot = NULL), i.e. all cells in the dataset are plotted.
palette_comb	A named character vector mapping marker combination names (as produced by assignMarkerCombinations) to colors. If NULL (default), a qualitative palette is generated with colorRampPalette and CATALYST:::.cluster_cols.

Value

A ggplot.

See Also

plotCellCountsAtLeast

Examples

```
# Define path to data
path_data <- system.file("extdata", package = "polyICSFlow")

# Read fcs files as flowSet
fs <- flowCore::read.flowSet(path = path_data,
                             pattern = ".fcs",
                             truncate_max_range = FALSE,
                             transformation = FALSE)

# Create GatingSet
gs <- flowWorkspace::GatingSet(fs)

# Apply GatingTemplate
gt <- openCyto::gatingTemplate(file.path(path_data, "gatingTemp_cytokines.csv"))
openCyto::gt_gating(x = gt, y = gs)

# Get marker positivity per cell
df_markerPos <- getMarkerPositivity(x = gs,
                                    gate_names = c("IFN+","TNFa+","IL2+","CD107a+"))
```

```
# Assign marker combinations to each cell based on marker positivity
df_markerComb <- assignMarkerCombinations(data_markerPos = df_markerPos,
                                          mode = "simple")

# Plot all cells
plotCellCountsExactly(data_markerComb = df_markerComb)

# Plot only subset of data (metacluster 2 and 4)
metacluster_labels <- readRDS(file.path(path_data, "cell_labels.rds"))
plotCellCountsExactly(data_markerComb = df_markerComb,
                      cell_subset_params = list(cellTypes = metacluster_labels,
                                                cellTypes_plot = c(2,4)))
```

`plotMarkerComb2DScatters` *Plot 2D scatterplots of all marker combinations.*

Description

Plots 2D scatter plots of all possible marker combinations, where each panel (facet) corresponds to one of the 2^n possible combinations of n markers. The top-left panel represents cells negative for all markers, while the bottom-right represents cells positive for all markers, with intermediate panels showing mixed combinations. Within each panel, the x-axis denotes the n markers and the y-axis shows the transformed MFI values. Each point corresponds to a single cell, with cells belonging to the combination represented in that facet highlighted in color, and all other cells displayed in grey background for reference.

Usage

```
plotMarkerComb2DScatters(
  input,
  assigned_MarkerCombs,
  cell_subset_params = list(cellTypes = NULL, cellTypes_plot = NULL),
  palette_comb = NULL,
  return_list = FALSE
)
```

Arguments

`input`	Either a matrix of marker expression or an object of class `SingleCellExperiment`, `GatingSet`, or `flowSet`, from which the expression data can be extracted. If input is either GatingSet or flowSet, an aggregate of the data is created with the `AggregateFlowFrames` function.
`assigned_MarkerCombs`	A vector of assigned marker combinations per cell in `x`, as created by `assignMarkerCombinations`.

`cell_subset_params`
: List with parameters describing what subset of cells (cellTypes_plot) should be plotted, from a vector of cell types (cellTypes). Default = `list(cellTypes = NULL, cellTypes_plot = NULL)`, i.e. all cells in the dataset are plotted.

`palette_comb`
: A named character vector mapping marker combination names (as produced by `assignMarkerCombinations`) to colors. If `NULL` (default), a qualitative palette is generated with `colorRampPalette` and `CATALYST:::.cluster_cols`.

`return_list`
: Logical. If FALSE (default), the plots are combined by cowplot. If TRUE, a list of plots are returned to allow further customization.

Value

A `ggplot` object visualizing 2D scatterplots of all marker combinations.

Examples

```
# Define path to data
path_data <- system.file("extdata", package = "polyICSFlow")

# Read fcs files as flowSet
fs <- flowCore::read.flowSet(path = path_data,
                             pattern = ".fcs",
                             truncate_max_range = FALSE,
                             transformation = FALSE)

# Create GatingSet
gs <- flowWorkspace::GatingSet(fs)

# Apply GatingTemplate
gt <- openCyto::gatingTemplate(file.path(path_data, "gatingTemp_cytokines.csv"))
openCyto::gt_gating(x = gt, y = gs)

# Get marker positivity per cell
df_markerPos <- getMarkerPositivity(x = gs,
                                    gate_names = c("IFN+","TNFa+","IL2+","CD107a+"))

# Assign marker combinations to each cell based on marker positivity
df_markerComb <- assignMarkerCombinations(data_markerPos = df_markerPos)

# Plot 2D scatter of marker combinations in dataset (all data)
# From gatingSet
plotMarkerComb2DScatters(input = gs,
                         assigned_MarkerCombs = df_markerComb$MarkerComb)

# From flowSet
plotMarkerComb2DScatters(input = fs,
                         assigned_MarkerCombs = df_markerComb$MarkerComb)

# Plot only subset of data (metacluster 2 and 4)
metacluster_labels <- readRDS(file.path(path_data, "cell_labels.rds"))
plotMarkerComb2DScatters(input = fs,
```

```
                          assigned_MarkerCombs = df_markerComb$MarkerComb,
                          cell_subset_params = list(cellTypes = metacluster_labels,
                                                    cellTypes_plot = c(2,4)))
# As list
plot_list <- plotMarkerComb2DScatters(input = fs,
                                      assigned_MarkerCombs = df_markerComb$MarkerComb,
                                      return_list = TRUE)
```

`plotMarkerCombHeatmap` *Plot a heatmap of all possible marker combinations.*

Description

Plot a heatmap of all possible marker combinations.

Usage

```
plotMarkerCombHeatmap(
  marker_names,
  orientation = c("vertical", "horizontal"),
  palette_count = NULL,
  palette_comb = NULL,
  full_names = TRUE
)
```

Arguments

`marker_names`	A character vector specifying the marker names used to generate all possible combinations.
`orientation`	A string specifying the orientation of the heatmap, either vertical or horizontal. Default = `"vertical"`.
`palette_count`	A named character vector mapping the number of markers per cell to colors. If `NULL` (default), a qualitative palette is generated with `brewer_pal`.
`palette_comb`	A named character vector mapping marker combination names to colors. If `NULL` (default), a qualitative palette is generated with `CATALYST:::.cluster_cols`.
`full_names`	Logical. If TRUE (default) the marker combinations are displayed with their full names (i.e., negative markers are also present). If FALSE, only the positive markers are displayed.

Value

A `Heatmap` object visualizing all marker combinations. The heatmap includes two layers of annotation:

- `MarkerCount`: A marker count annotation (colored with `palette_count`) showing the number of markers in each combination.
- `MarkerComb`: A combination annotation (colored with `palette_comb`) showing the exact subset of markers present.

See Also

Heatmap

Examples

```
# Vertical heatmap
plotMarkerCombHeatmap(marker_names = c("IFN","TNFa","IL2","CD107a"),
                      orientation = "vertical")

# Horizontal heatmap
plotMarkerCombHeatmap(marker_names = c("IFN","TNFa","IL2","CD107a"),
                      orientation = "horizontal")
```

Index