

Package ‘tidyFlowCore’

January 24, 2026

Type Package

Title tidyFlowCore: Bringing flowCore to the tidyverse

Version 1.5.0

Description tidyFlowCore bridges the gap between flow cytometry analysis using the flowCore Bioconductor package and the tidy data principles advocated by the tidyverse. It provides a suite of dplyr-, ggplot2-, and tidyr-like verbs specifically designed for working with flowFrame and flowSet objects as if they were tibbles; however, your data remain flowCore data structures under this layer of abstraction. tidyFlowCore enables intuitive and streamlined analysis workflows that can leverage both the Bioconductor and tidyverse ecosystems for cytometry data.

License MIT + file LICENSE

Encoding UTF-8

URL <https://github.com/keyes-timothy/tidyFlowCore>,
<https://keyes-timothy.github.io/tidyFlowCore/>

BugReports <https://github.com/keyes-timothy/tidyFlowCore/issues>

Depends R (>= 4.3)

Imports Biobase, dplyr, flowCore, ggplot2, methods, purrr, rlang,
stats, stringr, tibble, tidyr

RoxygenNote 7.3.1

Suggests BiocStyle, HDCytoData, knitr, RefManageR, rmarkdown,
sessioninfo, testthat (>= 3.0.0)

Config/testthat/edition 3

biocViews SingleCell, FlowCytometry, Infrastructure

VignetteBuilder knitr

git_url <https://git.bioconductor.org/packages/tidyFlowCore>

git_branch devel

git_last_commit 1848897

git_last_commit_date 2025-10-29

Repository Bioconductor 3.23

Date/Publication 2026-01-23

Author Timothy Keyes [cre] (ORCID: <<https://orcid.org/0000-0003-0423-9679>>),
Kara Davis [rth, own],
Garry Nolan [rth, own]

Maintainer Timothy Keyes <tkeyes@stanford.edu>

Contents

arrange.flowFrame	3
arrange.flowSet	4
as_flowFrame	4
as_flowSet	5
as_tof_tbl	6
as_tof_tbl.flowSet	7
count.flowFrame	8
count.flowSet	9
filter.flowFrame	10
filter.flowSet	11
ggplot.flowFrame	12
ggplot.flowSet	13
group_by.flowFrame	14
make_flowcore_annotated_data_frame	15
metal_masterlist	15
mutate.flowFrame	16
mutate.flowSet	17
nest.flowFrame	18
new_tof_tibble	19
pull.flowFrame	19
pull.flowSet	20
reexports	21
rename.flowFrame	21
rename.flowSet	22
rename_with.flowFrame	22
rename_with.flowSet	23
select.flowFrame	24
select.flowSet	25
simulate_cytometry_data	25
slice.flowFrame	26
slice.flowSet	27
slice_head.flowFrame	27
slice_head.flowSet	28
slice_max.flowFrame	29
slice_max.flowSet	30
slice_min.flowFrame	32
slice_min.flowSet	33
slice_sample.flowFrame	34

slice_sample.flowSet	35
slice_tail.flowFrame	36
slice_tail.flowSet	37
summarise.flowFrame	38
summarise.flowSet	38
summarize.flowFrame	39
summarize.flowSet	40
tof_find_panel_info	41
tof_get_panel	41
tof_set_panel	42
transmute.flowFrame	43
transmute.flowSet	43
ungroup.flowSet	44
unnest.flowSet	45

Index	47
-------	----

arrange.flowFrame	<i>Order rows using column values</i>
-------------------	---------------------------------------

Description

Order rows using column values

Usage

```
## S3 method for class 'flowFrame'  
arrange(.data, ..., .by_group = FALSE)
```

Arguments

.data	A flowFrame
...	Variables, or functions of variables, to arrange by.
.by_group	Unused.

Value

An object of the same type as .data. The output has the following properties: * All rows appear in the output, but (usually) in a different place. * Columns are not modified. * The [flowFrame](#)'s [identifier](#) will be preserved.

Examples

```
my_flowframe <- simulate_cytometry_data()$flowframe  
  
my_flowframe |>  
  dplyr::arrange(feature_1)
```

arrange.flowSet	<i>Order rows using column values</i>
-----------------	---------------------------------------

Description

Order rows using column values

Usage

```
## S3 method for class 'flowSet'
arrange(.data, ..., .by_group = FALSE)
```

Arguments

.data	A flowSet
...	Variables, or functions of variables, to arrange by.
.by_group	Unused.

Value

An object of the same type as .data. The output has the following properties: * All rows appear in the output, but (usually) in a different place. * Columns are not modified. * The [flowSet](#)'s [pData](#) will be preserved.

Examples

```
my_flowset <- simulate_cytometry_data()$flowset

my_flowset |>
  dplyr::arrange(feature_1)
```

as_flowFrame	<i>Coerce an object into a flowFrame</i>
--------------	--

Description

Coerce an object into a [flowFrame](#)
 Coerce a data.frame, tbl_df, or tof_tbl into a [flowFrame](#)

Usage

```
as_flowFrame(x, ...)

## S3 method for class 'tof_tbl'
as_flowFrame(x, ...)
```

Arguments

x	A data.frame, tbl_df, or tof_tbl.
...	Unused.

Value

A [flowFrame](#)

A [flowFrame](#). Note that all non-numeric columns in 'x' will be removed.

Examples

```
NULL
```

```
NULL
```

as_flowSet	<i>Coerce an object into a flowSet</i>
------------	--

Description

Coerce an object into a [flowSet](#)

Coerce a tof_tbl into a [flowSet](#)

Usage

```
as_flowSet(x, ...)

## S3 method for class 'tof_tbl'
as_flowSet(x, group_cols, ...)
```

Arguments

x	A tof_tbl.
...	Unused.
group_cols	Unquoted names of the columns in 'x' that should be used to group cells into separate flowFrames . Supports tidyselect helpers. Defaults to NULL (all cells are written into a single flowFrame). Note that the metadata column name "name" is a special value in the flowSet class, so if any of 'group_cols' refers to a column named "name," an error will be thrown.

Value

A [flowSet](#)

A [flowSet](#) in which cells are grouped into constituent [flowFrames](#) based on the values in 'group_cols'. If no 'group_cols' are specified, a [flowFrame](#) will be returned instead. Note that all non-numeric columns in will be removed.

Examples

```
NULL
```

```
NULL
```

```
as_tof_tbl
```

```
Coerce flowFrames or flowSets into tibbles.
```

Description

Coerce flowFrames or flowSets into tibbles.

Usage

```
as_tof_tbl(
  flow_data,
  .name_method = c("tidyFlowCore", "featureNames", "colnames"),
  sep = "|",
  ...
)
```

Arguments

<code>flow_data</code>	A flowFrame or flowSet
<code>.name_method</code>	A string indicating how tidyFlowCore should extract column names from 'flow_data'. Available options are "tidyFlowCore" (the default), which uses tidyFlowCore's internal heuristic to name columns; "featureNames", which uses featureNames to name the columns; and "colnames", which uses colnames to name the columns. Note that, in most cases, "featureNames" and "colnames" will give identical results.
<code>sep</code>	A string indicating which symbol should be used to separate antigen names and channel names in the columns of the output tof_tbl when .name_method = 'tidyFlowCore'.
<code>...</code>	Optional method-specific arguments.

Value

A cytometry-specialized tibble called a 'tof_tbl'.

Examples

```
input_file <- system.file("extdata", "0877408774.B08", package="flowCore")

input_flowframe <- flowCore::read.FCS(input_file)

tof_tibble <- as_tof_tbl(input_flowframe)
```

as_tof_tbl.flowSet *Convert an object into a tibble-flowCore abstraction (a 'tof_tbl')*

Description

Convert an object into a tibble-flowCore abstraction (a 'tof_tbl')

Usage

```
## S3 method for class 'flowSet'
as_tof_tbl(
  flow_data,
  .name_method = c("tidyFlowCore", "featureNames", "colnames"),
  sep = "|",
  ...,
  include_metadata = FALSE,
  include_tidyFlowCore_identifier = FALSE
)
```

Arguments

flow_data	A FlowSet
.name_method	A string indicating how tidyFlowCore should extract column names for the output tof_tbl from 'flow_data'. Available options are "tidyFlowCore" (the default), which uses tidyFlowCore's internal heuristic to name columns; "featureNames", which uses featureNames to name the columns; and "colnames", which uses colnames to name the columns.
sep	A string to use to separate the antigen name and its associated channel name in the column names of the output tibble. Defaults to " ".
...	Currently unused.
include_metadata	A boolean value indicating if the metadata for each .fcs file read by flowCore (stored in pData) should be merged into the final result. Defaults to FALSE.
include_tidyFlowCore_identifier	A boolean value indicating if tidyFlowCore's internal identifier for each flowFrame in the flowSet should be included in the output tof_tbl result. Defaults to FALSE.

Value

A cytometry-specialized tibble called a 'tof_tbl'.

count.flowFrame	<i>Count the observations in each group.</i>
-----------------	--

Description

Count the observations in each group.

Usage

```
## S3 method for class 'flowFrame'
count(x, ..., wt = NULL, sort = FALSE, name = NULL)
```

Arguments

x	A flowFrame
...	Variables to group by, named according to featureNames
wt	If NULL (the default), counts the number of rows in each group. If a variable, computes sum(wt) for each group.
sort	If TRUE, will show the largest groups at the top.
name	If omitted, it will default to n. If there's already a column called n, it will use nn. If there's a column called n and nn, it'll use nnn, and so on, adding ns until it gets a new name.

Value

A data.frame containing the groupwise counts.

Examples

```
my_flowframe <-
  simulate_cytometry_data()$flowframe |>
  dplyr::mutate(
    random_group =
      sample(
        c("a", "b"),
        size = nrow(simulate_cytometry_data()$flowframe),
        replace = TRUE
      )
  )

my_flowframe |>
  dplyr::count(random_group)
```

count.flowSet	<i>Count the observations in each group.</i>
---------------	--

Description

Count the observations in each group.

Usage

```
## S3 method for class 'flowSet'
count(x, ..., wt = NULL, sort = FALSE, name = NULL)
```

Arguments

x	A flowSet
...	Variables to group by, named according to featureNames or the columns of the flowSet's pData
wt	If NULL (the default), counts the number of rows in each group. If a variable, computes sum(wt) for each group.
sort	If TRUE, will show the largest groups at the top.
name	If omitted, it will default to n. If there's already a column called n, it will use nn. If there's a column called n and nn, it'll use nnn, and so on, adding ns until it gets a new name.

Value

A data.frame containing the groupwise counts. If no columns are specified in '...', the grouping is performed by experiment in the flowSet. Otherwise, the columns specified by '...' will be used for grouping.

Examples

```
my_flowset <- simulate_cytometry_data()$flowset

my_flowset |>
  dplyr::count()

my_flowset |>
  dplyr::count(cell_type)
```

filter.flowFrame	<i>Keep rows that match a condition.</i>
------------------	--

Description

Keep rows that match a condition.

Usage

```
## S3 method for class 'flowFrame'  
filter(.data, ..., .by = NULL, .preserve = FALSE)
```

Arguments

.data	A flowFrame
...	Expressions that return a logical value, and are defined in terms of the variables in the featureNames of .data. If multiple expressions are included, they are combined with the & operator. Only rows for which all conditions evaluate to TRUE are kept.
.by	Optionally, a selection of columns to group by for just this operation, functioning as an alternative to group_by().
.preserve	Unused.

Value

An object of the same type as .data. The output has the following properties: * Rows are a subset of the input, but appear in the same order. * Columns are not modified. * The [flowFrame](#)'s [identifier](#) will be preserved.

Examples

```
my_flowframe <- simulate_cytometry_data()$flowframe  
  
my_flowframe |>  
  dplyr::filter(feature_1 > 50)
```

filter.flowSet	<i>Keep rows that match a condition.</i>
----------------	--

Description

Keep rows that match a condition.

Usage

```
## S3 method for class 'flowSet'  
filter(.data, ..., .by = NULL, .preserve = FALSE)
```

Arguments

.data	A flowSet
...	Expressions that return a logical value, and are defined in terms of the variables in the featureNames of the flowFrames in .data. If multiple expressions are included, they are combined with the & operator. Only rows for which all conditions evaluate to TRUE are kept.
.by	Optionally, a selection of columns to group by for just this operation, functioning as an alternative to <code>group_by()</code> .
.preserve	Unused.

Value

An object of the same type as .data. The output has the following properties: * Rows are a subset of the input, but appear in the same order. * Columns are not modified. * The [flowSet](#)'s [pData](#) will be preserved.

Examples

```
my_flowset <- simulate_cytometry_data()$flowset  
  
my_flowset |>  
  dplyr::filter(feature_1 > 50)
```

ggplot.flowFrame	Create a new ggplot.
------------------	----------------------

Description

Create a new ggplot.

Usage

```
## S3 method for class 'flowFrame'
ggplot(
  data = NULL,
  mapping = ggplot2::aes(),
  ...,
  environment = parent.frame()
)
```

Arguments

data	Default dataset to use for plot in the form of a flowFrame . If not specified, must be supplied in each layer added to the plot.
mapping	Default list of aesthetic mappings to use for plot. If not specified, must be supplied in each layer added to the plot. Note that variable names used for aesthetic mappings come from the featureNames of the input flowFrame .
...	Other arguments passed on to methods. Not currently used.
environment	Deprecated. Used prior to tidy evaluation.

Value

A [ggplot](#)

Examples

```
simulations <- simulate_cytometry_data()
test_flowframe <- simulations$flowframe

flowframe_plot <-
  test_flowframe |>
  ggplot2::ggplot(ggplot2::aes(x = feature_1, y = feature_2)) +
  ggplot2::geom_point()
```

ggplot.flowSet	Create a new ggplot.
----------------	----------------------

Description

Create a new ggplot.

Usage

```
## S3 method for class 'flowSet'
ggplot(
  data = NULL,
  mapping = ggplot2::aes(),
  ...,
  environment = parent.frame()
)
```

Arguments

data	Default dataset to use for plot in the form of a flowSet . If not specified, must be supplied in each layer added to the plot. Note that any metadata stored in pData will be merged into the underlying flowCore-tibble abstraction and will thus be available for plotting.
mapping	Default list of aesthetic mappings to use for plot. If not specified, must be supplied in each layer added to the plot. Note that variable names used for aesthetic mappings come from the featureNames of the input flowSet 's constituent flowFrames .
...	Other arguments passed on to methods. Not currently used.
environment	Deprecated. Used prior to tidy evaluation.

Value

A [ggplot](#)

Examples

```
simulations <- simulate_cytometry_data()
test_flowset <- simulations$flowset

flowset_plot <-
  test_flowset |>
  ggplot2::ggplot(ggplot2::aes(x = feature_1, y = feature_2)) +
  ggplot2::geom_point()

flowset_plot_with_metadata <-
  test_flowset |>
  # note that `patient` below comes from the flowSet's metadata (pData)
```

```
ggplot2::ggplot(ggplot2::aes(x = feature_1, y = feature_2, color = patient)) +
  ggplot2::geom_point()
```

group_by.flowFrame	<i>Group a flowFrame into a flowSet using one or more variables.</i>
--------------------	--

Description

Group a flowFrame into a flowSet using one or more variables.

Usage

```
## S3 method for class 'flowFrame'
group_by(.data, ..., .add = FALSE, .drop = dplyr::group_by_drop_default(.data))
```

Arguments

.data	A flowFrame
...	Unquoted variables or columns to group by according to .data's featureNames .
.add	Unused.
.drop	Unused.

Value

A [flowSet](#) containing one [flowFrame](#) for each of the unique combinations of columns selected in Metadata about grouping columns will be stored in the output [flowSet](#)'s [pData](#).

Examples

```
my_flowframe <-
  simulate_cytometry_data()$flowframe |>
  dplyr::mutate(
    random_group =
      sample(
        c("a", "b"),
        size = nrow(simulate_cytometry_data()$flowframe),
        replace = TRUE
      )
  )

my_flowframe |>
  dplyr::group_by(random_group)
```

```
make_flowcore_annotated_data_frame
```

Make the AnnotatedDataFrame needed for the flowFrame class

Description

Make the AnnotatedDataFrame needed for the flowFrame class

Usage

```
make_flowcore_annotated_data_frame(maxes_and_mins)
```

Arguments

`maxes_and_mins` a data.frame containing information about the max and min values of each channel to be saved in the flowFrame.

Value

An AnnotatedDataFrame.

Examples

```
NULL
```

<code>metal_masterlist</code>	<i>A character vector of CyTOF metal name patterns supported by tidyFlowCore</i>
-------------------------------	--

Description

A character vector used by 'tof_find_panel_info' to detect and parse which CyTOF metals correspond to each channel in an input .fcs file.

Usage

```
data(metal_masterlist)
```

Format

A character vector in which each entry is a pattern that tidyFlowCore searches for in every CyTOF channel in input .fcs files. These patterns are an amalgamate of example .fcs files sampled from the studies linked below.

Value

None

Source

<https://github.com/kara-davis-lab/DDPR> <https://cytobank.org/nolanlab/reports/Levine2015.html> <https://cytobank.org/nolanlab/reports/Spitzer2015.html> <https://cytobank.org/nolanlab/reports/Spitzer2017.html> <https://community.cytobank.org/cytobank/projects/609>

mutate.flowFrame

Create, modify, and delete columns.

Description

Create, modify, and delete columns.

Usage

```
## S3 method for class 'flowFrame'
mutate(.data, ...)
```

Arguments

<code>.data</code>	A flowFrame
<code>...</code>	Name-value pairs. The name (the left side of the equals sign) gives the name of the column in the output. The right side of the equation performs computations using the names of each channel according to featureNames . Supports tidyselection.

Value

A [flowFrame](#). The output has the following properties: * Columns from `.data` will be preserved according to the `.keep` argument. * Existing columns that are modified by `...` will always be returned in their original location. * New columns created through `...` will be placed according to the `.before` and `.after` arguments. * The number of rows is not affected. * Columns given the value `NULL` will be removed.

Examples

```
my_flowframe <-
  simulate_cytometry_data()$flowframe |>
  dplyr::mutate(
    random_group =
      sample(
        c("a", "b"),
        size = nrow(simulate_cytometry_data()$flowframe),
        replace = TRUE
      )
  )

my_flowframe |>
```

```
dplyr::mutate(new_feature = feature_1 + feature_2)
```

mutate.flowSet	Create, modify, and delete columns.
----------------	-------------------------------------

Description

Create, modify, and delete columns.

Usage

```
## S3 method for class 'flowSet'  
mutate(.data, ...)
```

Arguments

.data	A flowSet
...	Name-value pairs. The name (the left side of the equals sign) gives the name of the column in the output. The right side of the equation performs computations using the names of each channel according to featureNames . Supports tidyselection.

Value

A [flowSet](#). The output has the following properties: * Columns from .data will be preserved according to the .keep argument. * Existing columns that are modified by ... will always be returned in their original location. * New columns created through ... will be placed according to the .before and .after arguments. * The number of rows is not affected. * Columns given the value NULL will be removed.

Examples

```
my_flowset <-  
  simulate_cytometry_data()$flowset  
  
my_flowset |>  
  dplyr::mutate(new_feature = feature_1 + feature_2)
```

nest.flowFrame	Nest a flowFrame into a flowSet
----------------	---

Description

Nest a [flowFrame](#) into a [flowSet](#)

Usage

```
## S3 method for class 'flowFrame'
nest(.data, ..., .by = NULL, .key = NULL, .names_sep = NULL)
```

Arguments

<code>.data</code>	A flowFrame
<code>...</code>	Columns to nest; these will appear in the inner flowFrames comprising the output flowSet . Specified using name-variable pairs of the form <code>new_col = c(col1, col2, col3)</code> . The right hand side can be any valid tidyselect expression. If not supplied, then <code>...</code> is derived as all columns not selected by <code>.by</code> .
<code>.by</code>	Columns to nest by; these will be stored in the pData of the output flowSet . <code>.by</code> can be used in place of or in conjunction with columns supplied through <code>...</code> . If not supplied, then <code>.by</code> is derived as all columns not selected by <code>...</code>
<code>.key</code>	Unused.
<code>.names_sep</code>	Unused.

Value

A [flowSet](#) wherein cells are grouped into constituent [flowFrames](#) based on which columns are used to nest.

Examples

```
my_flowframe <-
  simulate_cytometry_data()$flowframe |>
  dplyr::mutate(
    random_group =
      sample(
        c("a", "b"),
        size = nrow(simulate_cytometry_data()$flowframe),
        replace = TRUE
      )
  )
my_flowframe |>
  tidyr::nest(.by = random_group)
```

new_tof_tibble	<i>Constructor for a tof_tibble.</i>
----------------	--------------------------------------

Description

Constructor for a tof_tibble.

Usage

```
new_tof_tibble(x = dplyr::tibble(), panel = dplyr::tibble())
```

Arguments

x	A data.frame or tibble containing single-cell mass cytometry data such that rows are cells and columns are CyTOF measurements.
panel	A data.frame or tibble containing information about the panel for the mass cytometry data in x.

Value

A 'tof_tbl', a tibble extension that tracks a few other attributes that are useful for CyTOF data analysis.

See Also

Other tof_tbl utilities: [tof_get_panel\(\)](#), [tof_set_panel\(\)](#)

pull.flowFrame	<i>Extract a single column.</i>
----------------	---------------------------------

Description

pull() is similar to \$. It's mostly useful because it looks a little nicer in pipes.

Usage

```
## S3 method for class 'flowFrame'
pull(.data, var = -1, name = NULL, ...)
```

Arguments

.data	A flowFrame .
var	A variable specified as: * a literal variable name * a positive integer, giving the position counting from the left * a negative integer, giving the position counting from the right.
name	An optional parameter that specifies the column to be used as names for a named vector. Specified in a similar manner as var.
...	For use by methods.

Value

A vector the same size as .data.

Examples

```
my_flowframe <- simulate_cytometry_data()$flowframe

my_flowframe |>
  dplyr::pull(feature_1)
```

pull.flowSet	<i>Extract a single column.</i>
--------------	---------------------------------

Description

pull() is similar to \$. It's mostly useful because it looks a little nicer in pipes.

Usage

```
## S3 method for class 'flowSet'
pull(.data, var = -1, name = NULL, ...)
```

Arguments

.data	A flowSet .
var	A variable specified as: * a literal variable name * a positive integer, giving the position counting from the left * a negative integer, giving the position counting from the right.
name	An optional parameter that specifies the column to be used as names for a named vector. Specified in a similar manner as var.
...	For use by methods.

Value

A vector the same size as .data.

Examples

```
my_flowset <- simulate_cytometry_data()$flowset

my_flowset |>
  dplyr::pull(feature_1)
```

reexports	<i>Objects exported from other packages</i>
-----------	---

Description

These objects are imported from other packages. Follow the links below to see their documentation.

rlang [:=](#), [.data](#)

Value

See documentation in each object's original package.

Examples

```
# See examples in each object's original package
NULL
```

rename.flowFrame	<i>Rename columns in a flowFrame</i>
------------------	--

Description

Rename columns in a [flowFrame](#)

Usage

```
## S3 method for class 'flowFrame'
rename(.data, ...)
```

Arguments

<code>.data</code>	A flowFrame
<code>...</code>	Unquoted name-value pairs (as specified by featureNames). Use <code>new_name = old_name</code> to rename selected columns

Value

An object of the same type as `.data`. The output has the following properties: * Rows are not affected. * Column names are changed; column order is preserved. * The [flowFrame](#)'s `identifier` will be preserved.

Examples

```
my_flowframe <- simulate_cytometry_data()$flowframe

my_flowframe |>
  dplyr::rename(new_feature = feature_1)
```

rename.flowSet	Rename columns in a flowSet
----------------	---

Description

Rename columns in a [flowSet](#)

Usage

```
## S3 method for class 'flowSet'
rename(.data, ...)
```

Arguments

.data	A flowSet
...	Unquoted name-value pairs (as specified by the featureNames of the flowFrames making up the flowSet). Use new_name = old_name to rename selected columns

Value

An object of the same type as .data. The output has the following properties: * Rows are not affected. * Column names are changed; column order is preserved. * The [flowSet](#)'s [pData](#) will be preserved.

Examples

```
my_flowset <- simulate_cytometry_data()$flowset

my_flowset |>
  dplyr::rename(new_feature = feature_1)
```

rename_with.flowFrame	Rename columns in a flowFrame
-----------------------	---

Description

Rename columns in a [flowFrame](#)

Usage

```
## S3 method for class 'flowFrame'
rename_with(.data, .fn, .cols = dplyr::everything(), ...)
```

Arguments

<code>.data</code>	A flowFrame
<code>.fn</code>	A function used to transform the selected <code>.cols</code> . Should return a character vector the same length as the input.
<code>.cols</code>	Unquoted column names indicating which columns to rename (as specified by featureNames).
<code>...</code>	Additional arguments passed onto <code>.fn</code> .

Value

An object of the same type as `.data`. The output has the following properties: * Rows are not affected. * Column names are changed; column order is preserved. * The [flowFrame](#)'s `identifier` will be preserved.

Examples

```
my_flowframe <- simulate_cytometry_data()$flowframe

my_flowframe |>
  dplyr::rename_with(.fn = toupper)
```

rename_with.flowSet	Rename columns in a flowSet
---------------------	---

Description

Rename columns in a [flowSet](#)

Usage

```
## S3 method for class 'flowSet'
rename_with(.data, .fn, .cols = dplyr::everything(), ...)
```

Arguments

<code>.data</code>	A flowSet
<code>.fn</code>	A function used to transform the selected <code>.cols</code> . Should return a character vector the same length as the input.
<code>.cols</code>	Unquoted column names indicating which columns to rename (as specified by the featureNames of the flowFrames making up the flowSet).
<code>...</code>	Additional arguments passed onto <code>.fn</code> .

Value

An object of the same type as `.data`. The output has the following properties: * Rows are not affected. * Column names are changed; column order is preserved. * The `flowSet`'s `pData` will be preserved.

Examples

```
my_flowset <- simulate_cytometry_data()$flowset

my_flowset |>
  dplyr::rename_with(.fn = toupper)
```

<code>select.flowFrame</code>	<i>Keep or drop columns using their names and types.</i>
-------------------------------	--

Description

Keep or drop columns using their names and types.

Usage

```
## S3 method for class 'flowFrame'
select(.data, ...)
```

Arguments

<code>.data</code>	A <code>flowFrame</code>
<code>...</code>	One or more unquoted expressions separated by commas. Variables names (as specified by <code>featureNames</code>) can be used as if they were positions in the <code>flowFrame</code> . Supports tidyselection.

Value

A `flowFrame`. The output has the following properties: * Rows are not affected. * Output columns are a subset of input columns, potentially with a different order. Columns will be renamed if `new_name = old_name` form is used. * The `flowFrame`'s `identifier` will be preserved.

Examples

```
my_flowframe <- simulate_cytometry_data()$flowframe

my_flowframe |>
  dplyr::select(feature_1)
```

select.flowSet	<i>Keep or drop columns using their names and types.</i>
----------------	--

Description

Keep or drop columns using their names and types.

Usage

```
## S3 method for class 'flowSet'
select(.data, ...)
```

Arguments

.data	A flowSet
...	One or more unquoted expressions separated by commas. Variables names (as specified by the featureNames of the component flowFrames that make up the flowSet) can be used as if they were positions in the flowSet). Supports tidyselection.

Value

A [flowSet](#). The output has the following properties: * Rows are not affected. * Output columns are a subset of input columns, potentially with a different order. Columns will be renamed if new_name = old_name form is used. * The [flowSet](#)'s [pData](#) will be preserved.

Examples

```
my_flowset <-
  simulate_cytometry_data()$flowset

my_flowset |>
  dplyr::select(feature_1)
```

simulate_cytometry_data	<i>Simulate Cytometry Data for FlowSet and FlowFrame Analysis</i>
-------------------------	---

Description

Simulate Cytometry Data for FlowSet and FlowFrame Analysis

Usage

```
simulate_cytometry_data(num_cells = 100, num_features = 10, num_flowframes = 5)
```

Arguments

num_cells An integer indicating the number of cells to simulate.
 num_features An integer indicating how many features to simulate.
 num_flowframes An integer indicating how many flowFrames to simulate for the simulated flowSet.

Value

A list containing two entries: a flowFrame and a flowSet.

Examples

```
simulate_cytometry_data()
```

slice.flowFrame	<i>Subset rows using their positions</i>
-----------------	--

Description

Subset rows using their positions

Usage

```
## S3 method for class 'flowFrame'
slice(.data, ..., .by = NULL, .preserve = FALSE)
```

Arguments

.data A [flowFrame](#)
 ... Integer row values (to keep).
 .by Optionally, an unquoted selection of columns to group by for just this operation.
 An alternative to group_by.
 .preserve Currently unused.

Value

An object of the same type as .data. The output has the following properties: * Each row may appear 0, 1, or many times in the output. * Columns are not modified. * Groups are not modified.
 * A [flowSet](#)'s [pData](#) is preserved.

Examples

```
my_flowframe <- simulate_cytometry_data()$flowframe

my_flowframe |>
  dplyr::slice(1)
```

slice.flowSet	<i>Subset rows using their positions</i>
---------------	--

Description

Subset rows using their positions

Usage

```
## S3 method for class 'flowSet'
slice(.data, ..., .by = NULL, .preserve = FALSE)
```

Arguments

.data	A flowSet
...	Integer row values (to keep).
.by	Optionally, an unquoted selection of columns to group by for just this operation. An alternative to group_by.
.preserve	Currently unused.

Value

An object of the same type as .data. The output has the following properties: * Each row may appear 0, 1, or many times in the output. * Columns are not modified. * Groups are not modified. * A [flowSet](#)'s [pData](#) is preserved.

Examples

```
my_flowset <- simulate_cytometry_data()$flowset

my_flowset |>
  dplyr::slice(1)
```

slice_head.flowFrame	<i>Subset rows at the head of a data structure.</i>
----------------------	---

Description

Subset rows at the head of a data structure.

Usage

```
## S3 method for class 'flowFrame'
slice_head(.data, ..., n, prop, by = NULL)
```

Arguments

<code>.data</code>	A flowFrame
<code>...</code>	Unused.
<code>n, prop</code>	Provide either <code>n</code> , the number of rows, or <code>prop</code> , the proportion of rows to select. If neither are supplied, <code>n = 1</code> will be used. If <code>n</code> is greater than the number of rows in the group (or <code>prop > 1</code>), the result will be silently truncated to the group size. <code>prop</code> will be rounded towards zero to generate an integer number of rows. A negative value of <code>n</code> or <code>prop</code> will be subtracted from the group size. For example, <code>n = -2</code> with a group of 5 rows will select $5 - 2 = 3$ rows; <code>prop = -0.25</code> with 8 rows will select $8 * (1 - 0.25) = 6$ rows.
<code>by</code>	Optionally, an unquoted selection of columns to group by for just this operation. An alternative to <code>group_by</code> .

Value

An object of the same type as `.data`. The output has the following properties: * Each row may appear 0, 1, or many times in the output. * Columns are not modified. * Groups are not modified. * A [flowFrame](#)'s `identifier` is preserved.

Examples

```
my_flowframe <- simulate_cytometry_data()$flowframe

my_flowframe |>
  dplyr::slice_head(n = 5)
```

<code>slice_head.flowSet</code>	<i>Subset rows at the head of a data structure.</i>
---------------------------------	---

Description

Subset rows at the head of a data structure.

Usage

```
## S3 method for class 'flowSet'
slice_head(.data, ..., n, prop, by = NULL)
```

Arguments

<code>.data</code>	A flowSet
<code>...</code>	Unused.

n, prop	Provide either n, the number of rows, or prop, the proportion of rows to select. If neither are supplied, n = 1 will be used. If n is greater than the number of rows in the group (or prop > 1), the result will be silently truncated to the group size. prop will be rounded towards zero to generate an integer number of rows. A negative value of n or prop will be subtracted from the group size. For example, n = -2 with a group of 5 rows will select 5 - 2 = 3 rows; prop = -0.25 with 8 rows will select $8 * (1 - 0.25) = 6$ rows.
by	Optionally, an unquoted selection of columns to group by for just this operation. An alternative to group_by.

Value

An object of the same type as .data. The output has the following properties: * Each row may appear 0, 1, or many times in the output. * Columns are not modified. * Groups are not modified.
* A [flowSet](#)'s [pData](#) is preserved.

Examples

```
my_flowset <- simulate_cytometry_data()$flowset

my_flowset |>
  dplyr::slice_head(n = 10)
```

slice_max.flowFrame	<i>Subset rows of a data structure in order.</i>
---------------------	--

Description

Subset rows of a data structure in order.

Usage

```
## S3 method for class 'flowFrame'
slice_max(
  .data,
  order_by,
  ...,
  n,
  prop,
  by = NULL,
  with_ties = TRUE,
  na_rm = FALSE
)
```

Arguments

<code>.data</code>	A flowFrame
<code>order_by</code>	Variable or function of variables to order by. To order by multiple variables, wrap them in a data frame or tibble.
<code>...</code>	Unused.
<code>n, prop</code>	Provide either <code>n</code> , the number of rows, or <code>prop</code> , the proportion of rows to select. If neither are supplied, <code>n = 1</code> will be used. If <code>n</code> is greater than the number of rows in the group (or <code>prop > 1</code>), the result will be silently truncated to the group size. <code>prop</code> will be rounded towards zero to generate an integer number of rows. A negative value of <code>n</code> or <code>prop</code> will be subtracted from the group size. For example, <code>n = -2</code> with a group of 5 rows will select $5 - 2 = 3$ rows; <code>prop = -0.25</code> with 8 rows will select $8 * (1 - 0.25) = 6$ rows.
<code>by</code>	Optionally, an unquoted selection of columns to group by for just this operation. An alternative to <code>group_by</code> .
<code>with_ties</code>	Should ties be kept together? The default, <code>TRUE</code> , may return more rows than you request. Use <code>FALSE</code> to ignore ties, and return the first <code>n</code> rows.
<code>na_rm</code>	Should missing values in <code>order_by</code> be removed from the result? If <code>FALSE</code> , NA values are sorted to the end so they will only be included if there are insufficient non-missing values to reach <code>n/prop</code> .

Value

An object of the same type as `.data`. The output has the following properties: * Each row may appear 0, 1, or many times in the output. * Columns are not modified. * Groups are not modified. * A [flowFrame](#)'s `identifier` is preserved.

Examples

```
my_flowframe <- simulate_cytometry_data()$flowframe

my_flowframe |>
  dplyr::slice_max(order_by = feature_1, n = 5)
```

<code>slice_max.flowSet</code>	<i>Subset rows of a data structure in order.</i>
--------------------------------	--

Description

Subset rows of a data structure in order.

Usage

```
## S3 method for class 'flowSet'
slice_max(
  .data,
  order_by,
  ...,
  n,
  prop,
  by = NULL,
  with_ties = TRUE,
  na_rm = FALSE
)
```

Arguments

.data	A flowSet
order_by	Variable or function of variables to order by. To order by multiple variables, wrap them in a data frame or tibble.
...	Unused.
n, prop	Provide either n, the number of rows, or prop, the proportion of rows to select. If neither are supplied, n = 1 will be used. If n is greater than the number of rows in the group (or prop > 1), the result will be silently truncated to the group size. prop will be rounded towards zero to generate an integer number of rows. A negative value of n or prop will be subtracted from the group size. For example, n = -2 with a group of 5 rows will select 5 - 2 = 3 rows; prop = -0.25 with 8 rows will select 8 * (1 - 0.25) = 6 rows.
by	Optionally, an unquoted selection of columns to group by for just this operation. An alternative to group_by.
with_ties	Should ties be kept together? The default, TRUE, may return more rows than you request. Use FALSE to ignore ties, and return the first n rows.
na_rm	Should missing values in order_by be removed from the result? If FALSE, NA values are sorted to the end so they will only be included if there are insufficient non-missing values to reach n/prop.

Value

An object of the same type as .data. The output has the following properties: * Each row may appear 0, 1, or many times in the output. * Columns are not modified. * Groups are not modified. * A [flowSet](#)'s [pData](#) is preserved.

Examples

```
my_flowset <- simulate_cytometry_data()$flowset

my_flowset |>
  dplyr::slice_max(order_by = feature_1, n = 10)
```

slice_min.flowFrame	<i>Subset rows of a data structure in order.</i>
---------------------	--

Description

Subset rows of a data structure in order.

Usage

```
## S3 method for class 'flowFrame'
slice_min(
  .data,
  order_by,
  ...,
  n,
  prop,
  by = NULL,
  with_ties = TRUE,
  na_rm = FALSE
)
```

Arguments

.data	A flowFrame
order_by	Variable or function of variables to order by. To order by multiple variables, wrap them in a data frame or tibble.
...	Unused.
n, prop	Provide either n, the number of rows, or prop, the proportion of rows to select. If neither are supplied, n = 1 will be used. If n is greater than the number of rows in the group (or prop > 1), the result will be silently truncated to the group size. prop will be rounded towards zero to generate an integer number of rows. A negative value of n or prop will be subtracted from the group size. For example, n = -2 with a group of 5 rows will select 5 - 2 = 3 rows; prop = -0.25 with 8 rows will select 8 * (1 - 0.25) = 6 rows.
by	Optionally, an unquoted selection of columns to group by for just this operation. An alternative to group_by.
with_ties	Should ties be kept together? The default, TRUE, may return more rows than you request. Use FALSE to ignore ties, and return the first n rows.
na_rm	Should missing values in order_by be removed from the result? If FALSE, NA values are sorted to the end so they will only be included if there are insufficient non-missing values to reach n/prop.

Value

An object of the same type as .data. The output has the following properties: * Each row may appear 0, 1, or many times in the output. * Columns are not modified. * Groups are not modified. * A [flowFrame](#)'s [identifier](#) is preserved.

Examples

```
my_flowframe <- simulate_cytometry_data()$flowframe

my_flowframe |>
  dplyr::slice_min(order_by = feature_1, n = 5)
```

slice_min.flowSet	<i>Subset rows of a data structure in order.</i>
-------------------	--

Description

Subset rows of a data structure in order.

Usage

```
## S3 method for class 'flowSet'
slice_min(
  .data,
  order_by,
  ...,
  n,
  prop,
  by = NULL,
  with_ties = TRUE,
  na_rm = FALSE
)
```

Arguments

.data	A flowSet
order_by	Variable or function of variables to order by. To order by multiple variables, wrap them in a data frame or tibble.
...	Unused.
n, prop	Provide either n, the number of rows, or prop, the proportion of rows to select. If neither are supplied, n = 1 will be used. If n is greater than the number of rows in the group (or prop > 1), the result will be silently truncated to the group size. prop will be rounded towards zero to generate an integer number of rows. A negative value of n or prop will be subtracted from the group size. For example, n = -2 with a group of 5 rows will select 5 - 2 = 3 rows; prop = -0.25 with 8 rows will select 8 * (1 - 0.25) = 6 rows.
by	Optionally, an unquoted selection of columns to group by for just this operation. An alternative to group_by.
with_ties	Should ties be kept together? The default, TRUE, may return more rows than you request. Use FALSE to ignore ties, and return the first n rows.

`na_rm` Should missing values in `order_by` be removed from the result? If `FALSE`, NA values are sorted to the end so they will only be included if there are insufficient non-missing values to reach `n/prop`.

Value

An object of the same type as `.data`. The output has the following properties: * Each row may appear 0, 1, or many times in the output. * Columns are not modified. * Groups are not modified. * A `flowSet`'s `pData` is preserved.

Examples

```
my_flowset <- simulate_cytometry_data()$flowset

my_flowset |>
  dplyr::slice_max(order_by = feature_1, n = 10)
```

`slice_sample.flowFrame`

Subset rows randomly

Description

Subset rows randomly

Usage

```
## S3 method for class 'flowFrame'
slice_sample(.data, ..., n, prop, by = NULL, weight_by = NULL, replace = FALSE)
```

Arguments

<code>.data</code>	A <code>flowFrame</code>
<code>...</code>	Unused.
<code>n, prop</code>	Provide either <code>n</code> , the number of rows, or <code>prop</code> , the proportion of rows to select. If neither are supplied, <code>n = 1</code> will be used. If <code>n</code> is greater than the number of rows in the group (or <code>prop > 1</code>), the result will be silently truncated to the group size. <code>prop</code> will be rounded towards zero to generate an integer number of rows. A negative value of <code>n</code> or <code>prop</code> will be subtracted from the group size. For example, <code>n = -2</code> with a group of 5 rows will select $5 - 2 = 3$ rows; <code>prop = -0.25</code> with 8 rows will select $8 * (1 - 0.25) = 6$ rows.
<code>by</code>	Optionally, an unquoted selection of columns to group by for just this operation. An alternative to <code>group_by</code> .
<code>weight_by</code>	Sampling weights. This must evaluate to a vector of non-negative numbers the same length as the input. Weights are automatically standardized to sum to 1.
<code>replace</code>	Should sampling be performed with (<code>TRUE</code>) or without (<code>FALSE</code> , the default) replacement.

Value

An object of the same type as .data. The output has the following properties: * Each row may appear 0, 1, or many times in the output. * Columns are not modified. * Groups are not modified. * A `flowFrame`'s `identifier` is preserved.

Examples

```
my_flowframe <- simulate_cytometry_data()$flowframe

my_flowframe |>
  dplyr::slice_sample(n = 5)
```

slice_sample.flowSet	<i>Subset rows randomly</i>
----------------------	-----------------------------

Description

Subset rows randomly

Usage

```
## S3 method for class 'flowSet'
slice_sample(.data, ..., n, prop, by = NULL, weight_by = NULL, replace = FALSE)
```

Arguments

.data	A <code>flowSet</code>
...	Unused.
n, prop	Provide either n, the number of rows, or prop, the proportion of rows to select. If neither are supplied, n = 1 will be used. If n is greater than the number of rows in the group (or prop > 1), the result will be silently truncated to the group size. prop will be rounded towards zero to generate an integer number of rows. A negative value of n or prop will be subtracted from the group size. For example, n = -2 with a group of 5 rows will select 5 - 2 = 3 rows; prop = -0.25 with 8 rows will select 8 * (1 - 0.25) = 6 rows.
by	Optionally, an unquoted selection of columns to group by for just this operation. An alternative to <code>group_by</code> .
weight_by	Sampling weights. This must evaluate to a vector of non-negative numbers the same length as the input. Weights are automatically standardized to sum to 1.
replace	Should sampling be performed with (TRUE) or without (FALSE, the default) replacement.

Value

An object of the same type as `.data`. The output has the following properties: * Each row may appear 0, 1, or many times in the output. * Columns are not modified. * Groups are not modified. * A `flowSet`'s `pData` is preserved.

Examples

```
my_flowset <- simulate_cytometry_data()$flowset

my_flowset |>
  dplyr::slice_sample(n = 10)
```

`slice_tail.flowFrame` *Subset rows at the tail of a data structure.*

Description

Subset rows at the tail of a data structure.

Usage

```
## S3 method for class 'flowFrame'
slice_tail(.data, ..., n, prop, by = NULL)
```

Arguments

<code>.data</code>	A <code>flowFrame</code>
<code>...</code>	Unused.
<code>n, prop</code>	Provide either <code>n</code> , the number of rows, or <code>prop</code> , the proportion of rows to select. If neither are supplied, <code>n = 1</code> will be used. If <code>n</code> is greater than the number of rows in the group (or <code>prop > 1</code>), the result will be silently truncated to the group size. <code>prop</code> will be rounded towards zero to generate an integer number of rows. A negative value of <code>n</code> or <code>prop</code> will be subtracted from the group size. For example, <code>n = -2</code> with a group of 5 rows will select $5 - 2 = 3$ rows; <code>prop = -0.25</code> with 8 rows will select $8 * (1 - 0.25) = 6$ rows.
<code>by</code>	Optionally, an unquoted selection of columns to group by for just this operation. An alternative to <code>group_by</code> .

Value

An object of the same type as `.data`. The output has the following properties: * Each row may appear 0, 1, or many times in the output. * Columns are not modified. * Groups are not modified. * A `flowFrame`'s `identifier` is preserved.

Examples

```
my_flowframe <- simulate_cytometry_data()$flowframe

my_flowframe |>
  dplyr::slice_tail(n = 5)
```

slice_tail.flowSet	<i>Subset rows at the tail of a data structure.</i>
--------------------	---

Description

Subset rows at the tail of a data structure.

Usage

```
## S3 method for class 'flowSet'
slice_tail(.data, ..., n, prop, by = NULL)
```

Arguments

.data	A flowSet
...	Unused.
n, prop	Provide either n, the number of rows, or prop, the proportion of rows to select. If neither are supplied, n = 1 will be used. If n is greater than the number of rows in the group (or prop > 1), the result will be silently truncated to the group size. prop will be rounded towards zero to generate an integer number of rows. A negative value of n or prop will be subtracted from the group size. For example, n = -2 with a group of 5 rows will select 5 - 2 = 3 rows; prop = -0.25 with 8 rows will select 8 * (1 - 0.25) = 6 rows.
by	Optionally, an unquoted selection of columns to group by for just this operation. An alternative to group_by.

Value

An object of the same type as .data. The output has the following properties: * Each row may appear 0, 1, or many times in the output. * Columns are not modified. * Groups are not modified. * A [flowSet](#)'s [pData](#) is preserved.

Examples

```
my_flowset <- simulate_cytometry_data()$flowset

my_flowset |>
  dplyr::slice_tail(n = 10)
```

summarise.flowFrame *Summarize a flowFrame.*

Description

Summarize a flowFrame.

Usage

```
## S3 method for class 'flowFrame'
summarise(.data, ..., .by = NULL, .groups = NULL)
```

Arguments

.data	.data A flowFrame
...	Name-value pairs of summary functions. The name will be the name of the variable in the result.
.by	Optionally, a selection of columns to group by for just this operation, functioning as an alternative to group_by().
.groups	Grouping structure of the result. * "drop_last": dropping the last level of grouping. * "drop": All levels of grouping are dropped. * "keep": Same grouping structure as .data. * "rowwise": Each row is its own group.

Value

A data.frame containing the summarized result.

Examples

```
my_flowframe <- simulate_cytometry_data()$flowframe

my_flowframe |>
  dplyr::summarise(feature_1_mean = mean(feature_1))
```

summarise.flowSet *Summarize a flowSet.*

Description

Summarize a flowSet.

Usage

```
## S3 method for class 'flowSet'
summarise(.data, ..., .by = NULL, .groups = NULL)
```

Arguments

<code>.data</code>	<code>.data</code> A flowSet
<code>...</code>	Name-value pairs of summary functions. The name will be the name of the variable in the result.
<code>.by</code>	Optionally, a selection of columns to group by for just this operation, functioning as an alternative to <code>group_by()</code> .
<code>.groups</code>	Grouping structure of the result. * <code>"drop_last"</code> : dropping the last level of grouping. * <code>"drop"</code> : All levels of grouping are dropped. * <code>"keep"</code> : Same grouping structure as <code>.data</code> . * <code>"rowwise"</code> : Each row is its own group.

Value

A data.frame containing the summarized result.

Examples

```
my_flowset <- simulate_cytometry_data()$flowset

my_flowset |>
  dplyr::summarise(feature_1_mean = mean(feature_1))
```

<code>summarize.flowFrame</code>	<i>Summarize a flowFrame.</i>
----------------------------------	-------------------------------

Description

Summarize a flowFrame.

Usage

```
## S3 method for class 'flowFrame'
summarize(.data, ..., .by = NULL, .groups = NULL)
```

Arguments

<code>.data</code>	<code>.data</code> A flowFrame
<code>...</code>	Name-value pairs of summary functions. The name will be the name of the variable in the result.
<code>.by</code>	Optionally, a selection of columns to group by for just this operation, functioning as an alternative to <code>group_by()</code> .
<code>.groups</code>	Grouping structure of the result. * <code>"drop_last"</code> : dropping the last level of grouping. * <code>"drop"</code> : All levels of grouping are dropped. * <code>"keep"</code> : Same grouping structure as <code>.data</code> . * <code>"rowwise"</code> : Each row is its own group.

Value

A data.frame containing the summarized result.

Examples

```
my_flowframe <- simulate_cytometry_data()$flowframe

my_flowframe |>
  dplyr::summarize(feature_1_mean = mean(feature_1))
```

summarize.flowSet	<i>Summarize a flowSet.</i>
-------------------	-----------------------------

Description

Summarize a flowSet.

Usage

```
## S3 method for class 'flowSet'
summarize(.data, ..., .by = NULL, .groups = NULL)
```

Arguments

<code>.data</code>	A flowSet
<code>...</code>	Name-value pairs of summary functions. The name will be the name of the variable in the result.
<code>.by</code>	Optionally, a selection of columns to group by for just this operation, functioning as an alternative to <code>group_by()</code> .
<code>.groups</code>	Grouping structure of the result. * "drop_last": dropping the last level of grouping. * "drop": All levels of grouping are dropped. * "keep": Same grouping structure as <code>.data</code> . * "rowwise": Each row is its own group.

Value

A data.frame containing the summarized result.

Examples

```
my_flowset <- simulate_cytometry_data()$flowset

my_flowset |>
  dplyr::summarize(feature_1_mean = mean(feature_1))
```

tof_find_panel_info	<i>Use tidyFlowCore's opinionated heuristic for extracting a high-dimensional cytometry panel's channel-antigen pairs from a flowFrame (read from a .fcs file.)</i>
---------------------	---

Description

Using the character vectors obtained from the 'name' and 'desc' columns of the parameters of the data of a flowFrame, infer the cytometry panel used to collect the data and return it as a tidy tibble.

Usage

```
tof_find_panel_info(input_flowFrame)
```

Arguments

input_flowFrame

A flowFrame (just read from an .fcs file) from which a high-dimensional cytometry panel should be extracted

Value

A tibble with 4 columns ('channels', 'antigens', '.flowCore_featureNames' and '.flowCore_colnames'). The first two columns correspond to the channels and antigens of the high-dimensional cytometry panel used during data acquisition, respectively. The last two channels represent the featureNames and colnames attributes used to represent each channel in the input flowFrame.

tof_get_panel	<i>Get panel information from a tof_tibble</i>
---------------	--

Description

Get panel information from a tof_tibble

Usage

```
tof_get_panel(tof_tibble)
```

Arguments

tof_tibble A 'tof_tbl'.

Value

A tibble containing information about the CyTOF panel that was used during data acquisition for the data contained in 'tof_tibble'.

See Also

Other tof_tbl utilities: [new_tof_tibble\(\)](#), [tof_set_panel\(\)](#)

Examples

NULL

tof_set_panel	<i>Set panel information from a tof_tbl</i>
---------------	---

Description

Set panel information from a tof_tbl

Usage

```
tof_set_panel(tof_tibble, panel)
```

Arguments

tof_tibble	A 'tof_tbl'.
panel	A data.frame containing two columns ('channels' and 'antigens') representing the information about a panel

Value

A 'tof_tbl' containing information about the CyTOF panel that was used during data acquisition for the data contained in the input 'tof_tibble'. Two columns are required: "metals" and "antigens".

See Also

Other tof_tbl utilities: [new_tof_tibble\(\)](#), [tof_get_panel\(\)](#)

Examples

NULL

transmute.flowFrame	<i>Create, modify, and delete columns.</i>
---------------------	--

Description

Create, modify, and delete columns.

Usage

```
## S3 method for class 'flowFrame'
transmute(.data, ...)
```

Arguments

.data	A flowFrame
...	Name-value pairs. The name (the left side of the equals sign) gives the name of the column in the output. The right side of the equation performs computations using the names of each channel according to featureNames . Supports tidyselection.

Value

A [flowFrame](#). The output has the following properties: * Columns created or modified through ... will be returned in the order specified by * The number of rows is not affected. * Columns given the value NULL will be removed. * The [flowFrame](#)'s [identifier](#) will be preserved.

Examples

```
my_flowframe <- simulate_cytometry_data()$flowframe

my_flowframe |>
  dplyr::transmute(new_feature = feature_1 + feature_2)
```

transmute.flowSet	<i>Create, modify, and delete columns.</i>
-------------------	--

Description

Create, modify, and delete columns.

Usage

```
## S3 method for class 'flowSet'
transmute(.data, ...)
```

Arguments

`.data` A [flowSet](#)

`...` Name-value pairs. The name (the left side of the equals sign) gives the name of the column in the output. The right side of the equation performs computations using the names of each channel according to the [featureNames](#) of `.data`'s constituent [flowFrames](#). Supports tidyselection.

Value

A [flowSet](#). The output has the following properties: * Columns created or modified through ... will be returned in the order specified by * The number of rows is not affected. * Columns given the value NULL will be removed. * The [flowSet](#)'s [pData](#) will be preserved.

Examples

```
my_flowset <- simulate_cytometry_data()$flowset

my_flowset |>
  dplyr::transmute(new_feature = feature_1 + feature_2)
```

ungroup.flowSet	<i>Ungroup a flowSet</i>
-----------------	--------------------------

Description

Ungroup a flowSet

Usage

```
## S3 method for class 'flowSet'
ungroup(x, ...)
```

Arguments

`x` A [flowSet](#)

`...` Variables/columns in [pData](#) to remove from the grouping. Note that the "name" field in a [flowSet](#)'s [pData](#) is special in [flowCore](#), so requesting an ungrouping by name will result in a copied column called ".tidyFlowCore_name" in the result. Also note that the column ".tidyof_unique_identifier" is used internally and will not have any effect on the ungrouping.

Value

A [flowFrame](#) or [flowSet](#) depending on the degree of ungrouping. Note that unnest-ing and ungrouping a [flowSet](#) are equivalent.

Examples

```
my_flowset <- simulate_cytometry_data()$flowset

my_flowset |>
  dplyr::ungroup()
```

unnest.flowSet	Unnest a flowSet into a single flowFrame
----------------	--

Description

Unnest a [flowSet](#) into a single [flowFrame](#)

Usage

```
## S3 method for class 'flowSet'
unnest(
  data,
  cols,
  ...,
  keep_empty = FALSE,
  ptype = NULL,
  names_sep = NULL,
  names_repair = "check_unique"
)
```

Arguments

data	A flowSet
cols	Columns in pData to unnest.
...	Unused.
keep_empty	Unused.
ptype	Unused.
names_sep	Unused.
names_repair	Unused.

Value

A [flowFrame](#) or [flowSet](#) depending on the degree of unnest-ing. Note that unnest-ing and ungrouping a [flowSet](#) are equivalent.

Examples

```
my_flowset <- simulate_cytometry_data()$flowset

my_flowset |>
  tidyr::unnest(cols = c(patient, cell_type))

my_flowset |>
  tidyr::unnest(cols = patient)
```

Index

- * **datasets**
 - metal_masterlist, 15
- * **internal**
 - reexports, 21
- * **tof_tbl utilities**
 - new_tof_tibble, 19
 - tof_get_panel, 41
 - tof_set_panel, 42
- .data, 21
- .data (reexports), 21
- :=, 21
- := (reexports), 21
- arrange.flowFrame, 3
- arrange.flowSet, 4
- as_flowFrame, 4
- as_flowSet, 5
- as_tof_tbl, 6
- as_tof_tbl.flowSet, 7
- colnames, 6, 7
- count.flowFrame, 8
- count.flowSet, 9
- featureNames, 6–14, 16, 17, 21–25, 43, 44
- filter.flowFrame, 10
- filter.flowSet, 11
- flowFrame, 3–5, 8, 10–14, 16, 18, 19, 21–24, 26, 28, 30, 32, 34–36, 38, 39, 43–45
- flowSet, 4, 5, 9, 11, 13, 14, 17, 18, 20, 22–29, 31, 33–37, 39, 40, 44, 45
- ggplot, 12, 13
- ggplot.flowFrame, 12
- ggplot.flowSet, 13
- group_by.flowFrame, 14
- identifier, 3, 10, 21, 23, 24, 28, 30, 32, 35, 36, 43
- make_flowcore_annotated_data_frame, 15
- metal_masterlist, 15
- mutate.flowFrame, 16
- mutate.flowSet, 17
- nest.flowFrame, 18
- new_tof_tibble, 19, 42
- pData, 4, 7, 9, 11, 13, 14, 18, 22, 24–27, 29, 31, 34, 36, 37, 44, 45
- pull.flowFrame, 19
- pull.flowSet, 20
- reexports, 21
- rename.flowFrame, 21
- rename.flowSet, 22
- rename_with.flowFrame, 22
- rename_with.flowSet, 23
- select.flowFrame, 24
- select.flowSet, 25
- simulate_cytometry_data, 25
- slice.flowFrame, 26
- slice.flowSet, 27
- slice_head.flowFrame, 27
- slice_head.flowSet, 28
- slice_max.flowFrame, 29
- slice_max.flowSet, 30
- slice_min.flowFrame, 32
- slice_min.flowSet, 33
- slice_sample.flowFrame, 34
- slice_sample.flowSet, 35
- slice_tail.flowFrame, 36
- slice_tail.flowSet, 37
- summarise.flowFrame, 38
- summarise.flowSet, 38
- summarize.flowFrame, 39
- summarize.flowSet, 40
- tof_find_panel_info, 41
- tof_get_panel, 19, 41, 42
- tof_set_panel, 19, 42, 42

`transmute.flowFrame`, [43](#)

`transmute.flowSet`, [43](#)

`ungroup.flowSet`, [44](#)

`unnest.flowSet`, [45](#)