Package 'tidyseurat'

September 15, 2020

Type Package

Title Brings Seurat to the Tidyverse

Version 0.1.8

Description It creates an invisible layer that allow to see the 'Seurat' object as tibble and interact seamlessly with the tidyverse.

License GPL-3

Depends R (>= 3.6.0), Seurat

Imports tibble, dplyr, magrittr, tidyr, ggplot2, rlang, purrr, lifecycle, methods, plotly, ellipsis, tidyselect, utils

Suggests BiocStyle, testthat, knitr, GGally, markdown, SingleCellSignalR, dittoSeq

VignetteBuilder knitr

RdMacros lifecycle

Biarch true

biocViews AssayDomain, Infrastructure, RNASeq, DifferentialExpression, GeneExpression, Normalization, Clustering, QualityControl, Sequencing, Transcription, Transcriptomics

Encoding UTF-8

LazyData true

RoxygenNote 7.1.1

R topics documented:

`arrange` *drplyr-methods*

Description

'arrange()' order the rows of a data frame rows by the values of selected columns.

Unlike other dplyr verbs, 'arrange()' largely ignores grouping; you need to explicit mention grouping variables (or use 'by_group = TRUE') in order to group by them, and functions of variables are evaluated once per data frame, not once per group.

Usage

```
arrange(.data, ..., .by_group = FALSE)

## Default S3 method:
arrange(.data, ..., .by_group = FALSE)

bind_rows(..., .id = NULL, add.cell.ids = NULL)

bind_cols(..., .id = NULL)
```

Arguments

.data	A data frame, data frame extension (e.g. a tibble), or a lazy data frame (e.g. from dbplyr or dtplyr). See *Methods*, below, for more details.
...	<[‘tidy-eval‘][dplyr_tidy_eval]> Variables, or functions or variables. Use [desc()] to sort a variable in descending order.
.by_group	If ‘TRUE‘, will sort first by grouping variable. Applies to grouped data frames only.
.id	Data frame identifier. When ‘.id‘ is supplied, a new column of identifiers is created to link each row to its original data frame. The labels are taken from the named arguments to ‘bind_rows()‘. When a list of data frames is supplied, the labels are taken from the names of the list. If no names are found a numeric sequence is used instead.
add.cell.ids	from Seurat 3.0 A character vector of length(x = c(x, y)). Appends the corresponding values to the start of each objects’ cell names.

Details

Locales The sort order for character vectors will depend on the collating sequence of the locale in use: see [locales()].

Missing values Unlike base sorting with ‘sort()‘, ‘NA‘ are: * always sorted to the end for local data, even when wrapped with ‘desc()‘. * treated differently for remote data, depending on the backend.

Value

A tibble Arrange rows by column values

An object of the same type as ‘.data‘.

* All rows appear in the output, but (usually) in a different place. * Columns are not modified. * Groups are not modified. * Data frame attributes are preserved.

Methods

This function is a **generic**, which means that packages can provide implementations (methods) for other classes. See the documentation of individual methods for extra arguments and differences in behaviour.

The following methods are currently available in loaded packages:

See Also

Other single table verbs: filter(), mutate(), rename(), select(), slice(), summarise()

Examples

```
`%>%` = magrittr::`%>%`
pbmc_small %>% tidy %>% arrange(nFeature_RNA)
```

---

as_tibble *Coerce lists, matrices, and more to data frames*

Description

‘r lifecycle::badge("maturing")‘

‘as_tibble()‘ turns an existing object, such as a data frame or matrix, into a so-called tibble, a data frame with class [‘tbl_df‘]. This is in contrast with [tibble()], which builds a tibble from individual columns. ‘as_tibble()‘ is to [‘tibble()‘] as [base::as.data.frame()] is to [base::data.frame()].

‘as_tibble()‘ is an S3 generic, with methods for: * [‘data.frame‘][base::data.frame()]: Thin wrapper around the ‘list‘ method that implements tibble’s treatment of [rownames]. * [‘matrix‘][methods::matrix-class], [‘poly‘][stats::poly()], [‘ts‘][stats::ts()], [‘table‘][base::table()] * Default: Other inputs are first coerced with [base::as.data.frame()].

Usage

```
as_tibble(
  x,
  ...,
  .name_repair = c("check_unique", "unique", "universal", "minimal"),
  rownames = pkgconfig::get_config("tibble::rownames", NULL)
)
```

Arguments

x	A data frame, list, matrix, or other object that could reasonably be coerced to a tibble.
...	Unused, for extensibility.
.name_repair	see tidyr For compatibility only, do not use for new code.
rownames	How to treat existing row names of a data frame or matrix: * ‘NULL‘: remove row names. This is the default. * ‘NA‘: keep row names. * A string: the name of a new column. Existing rownames are transferred into this column and the ‘row.names‘ attribute is deleted. Read more in [rownames].

Value

A tibble

Row names

The default behavior is to silently remove row names.

New code should explicitly convert row names to a new column using the ‘rownames‘ argument.

For existing code that relies on the retention of row names, call ‘pkgconfig::set_config("tibble::rownames" = NA)‘ in your script or in your package’s [.onLoad()] function.

Life cycle

Using ‘as_tibble()‘ for vectors is superseded as of version 3.0.0, prefer the more expressive maturing ‘as_tibble_row()‘ and ‘as_tibble_col()‘ variants for new code.

See Also

[tibble()] constructs a tibble from individual columns. [enframe()] converts a named vector to a tibble with a column of names and column of values. Name repair is implemented using [vctrs::vec_as_names()].

Examples

```
pbmc_small %>% tidy %>% as_tibble()
```

---

bind *Efficiently bind multiple data frames by row and column*

Description

This is an efficient implementation of the common pattern of ‘do.call(rbind, dfs)‘ or ‘do.call(cbind, dfs)‘ for binding many data frames into one.

Arguments

`...`	Data frames to combine. Each argument can either be a data frame, a list that could be a data frame, or a list of data frames. When row-binding, columns are matched by name, and any missing columns will be filled with NA. When column-binding, rows are matched by position, so all data frames must have the same number of rows. To match by value, not position, see [mutate-joins].
`.id`	Data frame identifier. When ‘.id‘ is supplied, a new column of identifiers is created to link each row to its original data frame. The labels are taken from the named arguments to ‘bind_rows()‘. When a list of data frames is supplied, the labels are taken from the names of the list. If no names are found a numeric sequence is used instead.
`add.cell.ids`	from Seurat 3.0 A character vector of length(x = c(x, y)). Appends the corresponding values to the start of each objects’ cell names.

Details

The output of ‘bind_rows()‘ will contain a column if that column appears in any of the inputs.

Value

‘bind_rows()‘ and ‘bind_cols()‘ return the same type as the first input, either a data frame, ‘tbl_df‘, or ‘grouped_df‘.

Examples

```
`%>%` = magrittr::`%>%`
tt = pbmc_small %>% tidy
bind_rows(    tt, tt  )

tt_bind = tt %>% select(nCount_RNA ,nFeature_RNA)
tt %>% bind_cols(tt_bind)
```

---

cell_type_df *Example data set 2*

Description

Example data set 2

Usage

```
cell_type_df
```

Format

An object of class `tbl_df` (inherits from `tbl`, `data.frame`) with 80 rows and 2 columns.

---

count *Count observations by group*

Description

‘count()‘ lets you quickly count the unique values of one or more variables: ‘df ‘df ‘count()‘ is paired with ‘tally()‘, a lower-level helper that is equivalent to ‘df switching the summary from ‘n = n()‘ to ‘n = sum(wt)‘.

‘add_count()‘ are ‘add_tally()‘ are equivalents to ‘count()‘ and ‘tally()‘ but use ‘mutate()‘ instead of ‘summarise()‘ so that they add a new column with group-wise counts.

Usage

```
count(
  x,
  ...,
  wt = NULL,
  sort = FALSE,
  name = NULL,
  .drop = group_by_drop_default(x)
)
```

Arguments

x	A data frame, data frame extension (e.g. a tibble), or a lazy data frame (e.g. from dbplyr or dtplyr).
...	<[‘data-masking‘][dplyr_data_masking]> Variables to group by.
wt	<[‘data-masking‘][dplyr_data_masking]> Frequency weights. Can be ‘NULL‘ or a variable: * 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	The name of the new column in the output. If omitted, it will default to ‘n‘. If there’s already a column called ‘n‘, it will error, and require you to specify the name.
.drop	For ‘count()‘: if ‘FALSE‘ will include counts for empty groups (i.e. for levels of factors that don’t exist in the data). Deprecated in ‘add_count()‘ since it didn’t actually affect the output.

Value

An object of the same type as ‘.data‘. ‘count()‘ and ‘add_count()‘ group transiently, so the output has the same groups as the input.

Examples

```
`%>%` = magrittr::`%>%`
pbmc_small %>% tidy %>% count(groups)
```

---

distinct *distinct*

Description

distinct

Usage

```
distinct(.data, ..., .keep_all = FALSE)
```

Arguments

.data	A tbl. (See dplyr)
...	Data frames to combine (See dplyr)
.keep_all	If TRUE, keep all variables in .data. If a combination of ... is not distinct, this keeps the first row of values. (See dplyr)

Value

A tidyseurat object

Examples

```
`%>%` = magrittr::`%>%`
pbmc_small %>% tidy %>% distinct(groups)
```

`extract` *Extract a character column into multiple columns using regular expression groups*

Description

Given a regular expression with capturing groups, ‘extract()‘ turns each group into a new column. If the groups don’t match, or the input is NA, the output will be NA.

Usage

```
extract(
  data,
  col,
  into,
  regex = "([[:alnum:]]+)",
  remove = TRUE,
  convert = FALSE,
  ...
)
```

Arguments

`data`	A tidyseurat object
`col`	Column name or position. This is passed to [tidyselect::vars_pull()]. This argument is passed by expression and supports [quasiquotation][rlang::quasiquotation] (you can unquote column names or column positions).
`into`	Names of new variables to create as character vector. Use ‘NA‘ to omit the variable in the output.
`regex`	a regular expression used to extract the desired values. There should be one group (defined by ‘()‘) for each element of ‘into‘.
`remove`	If ‘TRUE‘, remove input column from output data frame.
`convert`	If ‘TRUE‘, will run [type.convert()] with ‘as.is = TRUE‘ on new columns. This is useful if the component columns are integer, numeric or logical. NB: this will cause string ‘"NA"‘s to be converted to ‘NA‘s.
`...`	Additional arguments passed on to methods.

Value

A tidyseurat objector a tibble depending on input

See Also

[separate()] to split up by a separator.

Examples

```
pbmc_small %>% tidy %>% extract(groups, into = "g", regex = "g([0-9])", convert = TRUE)
```

`filter` *Subset rows using column values*

Description

'filter()' retains the rows where the conditions you provide a 'TRUE'. Note that, unlike base subsetting with '[', rows where the condition evaluates to 'NA' are dropped.

Usage

```
filter(.data, ..., .preserve = FALSE)
```

Arguments

`.data`	A data frame, data frame extension (e.g. a tibble), or a lazy data frame (e.g. from dbplyr or dtplyr). See *Methods*, below, for more details.
`...`	<['tidy-eval'][dplyr_tidy_eval]> Logical predicates defined in terms of the variables in '.data'. Multiple conditions are combined with '&'. Only rows where the condition evaluates to 'TRUE' are kept.
`.preserve`	when 'FALSE' (the default), the grouping structure is recalculated based on the resulting data, otherwise it is kept as is.

Details

dplyr is not yet smart enough to optimise filtering optimisation on grouped datasets that don't need grouped calculations. For this reason, filtering is often considerably faster on [ungroup()]ed data.

Value

An object of the same type as '.data'.

* Rows are a subset of the input, but appear in the same order. * Columns are not modified. * The number of groups may be reduced (if '.preserve' is not 'TRUE'). * Data frame attributes are preserved.

Useful filter functions

* ['=='], ['>'], ['>='] etc * ['&'], ['|'], ['¡], [xor()] * [is.na()] * [between()], [near()]

Grouped tibbles

Because filtering expressions are computed within groups, they may yield different results on grouped tibbles. This will be the case as soon as an aggregating, lagging, or ranking function is involved. Compare this ungrouped filtering:

The former keeps rows with 'mass' greater than the global average whereas the latter keeps rows with 'mass' greater than the gender

average.

Methods

This function is a **generic**, which means that packages can provide implementations (methods) for other classes. See the documentation of individual methods for extra arguments and differences in behaviour.

The following methods are currently available in loaded packages:

See Also

[filter_all()], [filter_if()] and [filter_at()].

Other single table verbs: `arrange()`, `mutate()`, `rename()`, `select()`, `slice()`, `summarise()`

Examples

```
`%>%` = magrittr::`%>%`
pbmc_small %>% tidy %>% filter(groups == "g1")

# Learn more in ?dplyr_tidy_eval
```

---

full_join *Full join datasets*

Description

Full join datasets

Usage

```
full_join(x, y, by = NULL, copy = FALSE, suffix = c(".x", ".y"), ...)
```

Arguments

x	tbls to join. (See dplyr)
y	tbls to join. (See dplyr)
by	A character vector of variables to join by. (See dplyr)
copy	If x and y are not from the same data source, and copy is TRUE, then y will be copied into the same src as x. (See dplyr)
suffix	If there are non-joined duplicate variables in x and y, these suffixes will be added to the output to disambiguate them. Should be a character vector of length 2. (See dplyr)
...	Data frames to combine (See dplyr)

Value

A tidyseurat object

Examples

```
`%>%` = magrittr::`%>%`

tt = pbmc_small %>% tidy
tt %>% full_join(tibble::tibble(groups = "g1", other=1:4))
```

---

ggplot *Create a new ggplot from a tidyseurat object*

Description

'ggplot()' initializes a ggplot object. It can be used to declare the input data frame for a graphic and to specify the set of plot aesthetics intended to be common throughout all subsequent layers unless specifically overridden.

Usage

```
ggplot(.data = NULL, mapping = aes(), ..., environment = parent.frame())
```

Arguments

.data	Default dataset to use for plot. If not already a data.frame, will be converted to one by [fortify()]. 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.
...	Other arguments passed on to methods. Not currently used.
environment	DEPRECATED. Used prior to tidy evaluation.

Details

'ggplot()' is used to construct the initial plot object, and is almost always followed by '+' to add component to the plot. There are three common ways to invoke 'ggplot()':

The first method is recommended if all layers use the same data and the same set of aesthetics, although this method can also be used to add a layer using data from another data frame. See the first example below. The second method specifies the default data frame to use for the plot, but no aesthetics are defined up front. This is useful when one data frame is used predominantly as layers are added, but the aesthetics may vary from one layer to another. The third method initializes a skeleton 'ggplot' object which is fleshed out as layers are added. This method is useful when multiple data frames are used to produce different layers, as is often the case in complex graphics.

Value

A ggplot

Examples

```
# Generate some sample data, then compute mean and standard deviation
# in each group
```

group_by *Group by one or more variables*

Description

Most data operations are done on groups defined by variables. ‘group_by()‘ takes an existing tbl and converts it into a grouped tbl where operations are performed "by group". ‘ungroup()‘ removes grouping.

Usage

```
group_by(.data, ..., .add = FALSE, .drop = group_by_drop_default(.data))
```

Arguments

.data	A data frame, data frame extension (e.g. a tibble), or a lazy data frame (e.g. from dbplyr or dtplyr). See *Methods*, below, for more details.
...	In ‘group_by()‘, variables or computations to group by. In ‘ungroup()‘, variables to remove from the grouping.
.add	When ‘FALSE‘, the default, ‘group_by()‘ will override existing groups. To add to the existing groups, use ‘.add = TRUE‘. This argument was previously called ‘add‘, but that prevented creating a new grouping variable called ‘add‘, and conflicts with our naming conventions.
.drop	When ‘.drop = TRUE‘, empty groups are dropped. See [group_by_drop_default()] for what the default value is for this argument.

Value

A [grouped data frame][grouped_df()], unless the combination of ‘...‘ and ‘add‘ yields a non empty set of grouping columns, a regular (ungrouped) data frame otherwise.

Methods

These function are **generic**s, which means that packages can provide implementations (methods) for other classes. See the documentation of individual methods for extra arguments and differences in behaviour.

Methods available in currently loaded packages:

Examples

```
`%>%` = magrittr::`%>%`
pbmc_small %>% tidy %>% group_by(groups)
```

inner_join *Inner join datasets*

Description

Inner join datasets

Usage

```
inner_join(x, y, by = NULL, copy = FALSE, suffix = c(".x", ".y"), ...)
```

Arguments

x	tbls to join. (See dplyr)
y	tbls to join. (See dplyr)
by	A character vector of variables to join by. (See dplyr)
copy	If x and y are not from the same data source, and copy is TRUE, then y will be copied into the same src as x. (See dplyr)
suffix	If there are non-joined duplicate variables in x and y, these suffixes will be added to the output to disambiguate them. Should be a character vector of length 2. (See dplyr)
...	Data frames to combine (See dplyr)

Value

A tidyseurat object

Examples

```
`%>%` = magrittr::`%>%`

tt = pbmc_small %>% tidy
tt %>% inner_join(tt %>% distinct(groups) %>% mutate(new_column = 1:2) %>% slice(1))
```

join_transcripts *Extract and join information for transcripts.*

Description

join_transcripts() extracts and joins information for specified transcripts

Usage

```
join_transcripts(
  .data,
  transcripts = NULL,
  all = FALSE,
  exclude_zeros = FALSE,
  shape = "long"
)
```

Arguments

.data	A tidyseurat object
transcripts	A vector of transcript identifiers to join
all	If TRUE return all
exclude_zeros	If TRUE exclude zero values
shape	Format of the returned table "long" or "wide"

Details

Experimental

This function extracts information for specified transcripts and returns the information in either long or wide format.

Value

A ‘tbl‘ containing the information.for the specified transcripts

Examples

```
pbmc_small %>%
tidy %>%
join_transcripts(transcripts = c("HLA-DRA", "LYZ"))
```

---

left_join *Left join datasets*

Description

Left join datasets

Usage

```
left_join(x, y, by = NULL, copy = FALSE, suffix = c(".x", ".y"), ...)
```

Arguments

x	tbls to join. (See dplyr)
y	tbls to join. (See dplyr)
by	A character vector of variables to join by. (See dplyr)
copy	If x and y are not from the same data source, and copy is TRUE, then y will be copied into the same src as x. (See dplyr)
suffix	If there are non-joined duplicate variables in x and y, these suffixes will be added to the output to disambiguate them. Should be a character vector of length 2. (See dplyr)
...	Data frames to combine (See dplyr)

Value

A tidyseurat object

Examples

```
`%>%` = magrittr::`%>%`

tt = pbmc_small %>% tidy
tt %>% left_join(tt %>% distinct(groups) %>% mutate(new_column = 1:2))
```

`mutate` *Create, modify, and delete columns*

Description

‘mutate()‘ adds new variables and preserves existing ones; ‘transmute()‘ adds new variables and drops existing ones. New variables overwrite existing variables of the same name. Variables can be removed by setting their value to ‘NULL‘.

Usage

```
mutate(.data, ...)
```

Arguments

`.data`	A data frame, data frame extension (e.g. a tibble), or a lazy data frame (e.g. from dbplyr or dtplyr). See *Methods*, below, for more details.
`...`	<[‘tidy-eval‘][dplyr_tidy_eval]> Name-value pairs. The name gives the name of the column in the output. The value can be: * A vector of length 1, which will be recycled to the correct length. * A vector the same length as the current group (or the whole data frame if ungrouped). * ‘NULL‘, to remove the column. * A data frame or tibble, to create multiple columns in the output.

Value

An object of the same type as ‘.data‘.

For ‘mutate()‘:

* Rows are not affected. * Existing columns will be preserved unless explicitly modified. * New columns will be added to the right of existing columns. * Columns given value ‘NULL‘ will be removed * Groups will be recomputed if a grouping variable is mutated. * Data frame attributes are preserved.

For ‘transmute()‘:

* Rows are not affected. * Apart from grouping variables, existing columns will be remove unless explicitly kept. * Column order matches order of expressions. * Groups will be recomputed if a grouping variable is mutated. * Data frame attributes are preserved.

Useful mutate functions

* [‘+‘], [‘-‘], [log()], etc., for their usual mathematical meanings

* [lead()], [lag()]

* [dense_rank()], [min_rank()], [percent_rank()], [row_number()], [cume_dist()], [ntile()]

* [cumsum()], [cummean()], [cummin()], [cummax()], [cumany()], [cumall()]

* [na_if()], [coalesce()]

* [if_else()], [recode()], [case_when()]

Grouped tibbles

Because mutating expressions are computed within groups, they may yield different results on grouped tibbles. This will be the case as soon as an aggregating, lagging, or ranking function is involved. Compare this ungrouped mutate:

With the grouped equivalent:

The former normalises ‘mass‘ by the global average whereas the latter normalises by the averages within gender levels.

Methods

These function are **generic**s, which means that packages can provide implementations (methods) for other classes. See the documentation of individual methods for extra arguments and differences in behaviour.

Methods available in currently loaded packages:

See Also

Other single table verbs: `arrange()`, `filter()`, `rename()`, `select()`, `slice()`, `summarise()`

Examples

```
`%>%` = magrittr::`%>%`
pbmc_small %>% tidy %>% mutate(nFeature_RNA = 1)
```

---

pbmc_small *Example data set*

Description

Example data set

Usage

```
pbmc_small
```

Format

An object of class `Seurat` with 230 rows and 80 columns.

pbmc_small_nested_interactions *Example data set 2*

Description

Example data set 2

Usage

```
pbmc_small_nested_interactions
```

Format

An object of class `tbl_df` (inherits from `tbl`, `data.frame`) with 100 rows and 9 columns.

pivot_longer *Pivot data from wide to long*

Description

Maturing

‘pivot_longer()‘ "lengthens" data, increasing the number of rows and decreasing the number of columns. The inverse transformation is [pivot_wider()]

Learn more in ‘vignette("pivot")‘.

Usage

```
pivot_longer(
  data,
  cols,
  names_to = "name",
  names_prefix = NULL,
  names_sep = NULL,
  names_pattern = NULL,
  names_ptypes = list(),
  names_transform = list(),
  names_repair = "check_unique",
  values_to = "value",
  values_drop_na = FALSE,
  values_ptypes = list(),
  values_transform = list(),
  ...
)
```

Arguments

data	A data frame to pivot.
cols	<[‘tidy-select‘][tidyr_tidy_select]> Columns to pivot into longer format.
names_to	A string specifying the name of the column to create from the data stored in the column names of ‘data‘. Can be a character vector, creating multiple columns, if ‘names_sep‘ or ‘names_pattern‘ is provided. In this case, there are two special values you can take advantage of: * ‘NA‘ will discard that component of the name. * ‘.value‘ indicates that component of the name defines the name of the column containing the cell values, overriding ‘values_to‘.
names_prefix	A regular expression used to remove matching text from the start of each variable name.
names_sep, names_pattern	If ‘names_to‘ contains multiple values, these arguments control how the column name is broken up. ‘names_sep‘ takes the same specification as [separate()], and can either be a numeric vector (specifying positions to break on), or a single string (specifying a regular expression to split on). ‘names_pattern‘ takes the same specification as [extract()], a regular expression containing matching groups (‘()‘). If these arguments do not give you enough control, use ‘pivot_longer_spec()‘ to create a spec object and process manually as needed.
names_ptypes, values_ptypes	A list of column name-prototype pairs. A prototype (or ptype for short) is a zero-length vector (like ‘integer()‘ or ‘numeric()‘) that defines the type, class, and attributes of a vector. Use these arguments to confirm that the created columns are the types that you expect. If not specified, the type of the columns generated from ‘names_to‘ will be character, and the type of the variables generated from ‘values_to‘ will be the common type of the input columns used to generate them.
names_transform, values_transform	A list of column name-function pairs. Use these arguments if you need to change the type of specific columns. For example, ‘names_transform = list(week = as.integer)‘ would convert a character week variable to an integer.
names_repair	What happens if the output has invalid column names? The default, ‘"check_unique"‘ is to error if the columns are duplicated. Use ‘"minimal"‘ to allow duplicates in the output, or ‘"unique"‘ to de-duplicated by adding numeric suffixes. See [vctrs::vec_as_names()] for more options.
values_to	A string specifying the name of the column to create from the data stored in cell values. If ‘names_to‘ is a character containing the special ‘.value‘ sentinel, this value will be ignored, and the name of the value column will be derived from part of the existing column names.
values_drop_na	If ‘TRUE‘, will drop rows that contain only ‘NA‘s in the ‘value_to‘ column. This effectively converts explicit missing values to implicit missing values, and should generally be used only when missing values in ‘data‘ were created by its structure.
...	Additional arguments passed on to methods.

Details

‘pivot_longer()‘ is an updated approach to [gather()], designed to be both simpler to use and to handle more use cases. We recommend you use ‘pivot_longer()‘ for new code; ‘gather()‘ isn’t going away but is no longer under active development.

Value

A tidyseurat objector a tibble depending on input

Examples

```
# See vignette("pivot") for examples and explanation

library(dplyr)
pbmc_small %>% tidy %>% pivot_longer(c(orig.ident, groups), names_to = "name", values_to = "value")
```

---

plot_ly *Initiate a plotly visualization*

Description

This function maps R objects to [plotly.js](https://plot.ly/javascript/), an (MIT licensed) web-based interactive charting library. It provides abstractions for doing common things (e.g. mapping data values to fill colors (via ‘color‘) or creating [animation]s (via ‘frame‘)) and sets some different defaults to make the interface feel more ’R-like’ (i.e., closer to [plot()] and [ggplot2::qplot()]).

Usage

```
plot_ly(
  data = data.frame(),
  ...,
  type = NULL,
  name = NULL,
  color = NULL,
  colors = NULL,
  alpha = NULL,
  stroke = NULL,
  strokes = NULL,
  alpha_stroke = 1,
  size = NULL,
  sizes = c(10, 100),
  span = NULL,
  spans = c(1, 20),
  symbol = NULL,
  symbols = NULL,
  linetype = NULL,
  linetypes = NULL,
  split = NULL,
  frame = NULL,
  width = NULL,
```

```
  height = NULL,
  source = "A"
)
```

Arguments

data	A data frame (optional) or [crosstalk::SharedData] object.
...	Arguments (i.e., attributes) passed along to the trace 'type'. See [schema()] for a list of acceptable attributes for a given trace 'type' (by going to 'traces' -> 'type' -> 'attributes'). Note that attributes provided at this level may override other arguments (e.g. 'plot_ly(x = 1:10, y = 1:10, color = I("red"), marker = list(color = "blue"))').
type	A character string specifying the trace type (e.g. '"scatter"', '"bar"', '"box"', etc). If specified, it *always* creates a trace, otherwise
name	Values mapped to the trace's name attribute. Since a trace can only have one name, this argument acts very much like 'split' in that it creates one trace for every unique value.
color	Values mapped to relevant 'fill-color' attribute(s) (e.g. [fillcolor](https://plot.ly/r/reference#scatter-fillcolor), [marker.color](https://plot.ly/r/reference#scatter-marker-color), [textfont.color](https://plot textfont-color), etc.). The mapping from data values to color codes may be controlled using 'colors' and 'alpha', or avoided altogether via [I()] (e.g., 'color = I("red")'). Any color understood by [grDevices::col2rgb()] may be used in this way.
colors	Either a colorbrewer2.org palette name (e.g. "YlOrRd" or "Blues"), or a vector of colors to interpolate in hexadecimal "#RRGGBB" format, or a color interpolation function like 'colorRamp()'.
alpha	A number between 0 and 1 specifying the alpha channel applied to 'color'. Defaults to 0.5 when mapping to [fillcolor](https://plot.ly/r/reference#scatter-fillcolor) and 1 otherwise.
stroke	Similar to 'color', but values are mapped to relevant 'stroke-color' attribute(s) (e.g., [marker.line.color](https://plot.ly/r/reference#scatter-marker-line-color) and [line.color](https://plot.ly/r/reference#scatter-line-color) for filled polygons). If not specified, 'stroke' inherits from 'color'.
strokes	Similar to 'colors', but controls the 'stroke' mapping.
alpha_stroke	Similar to 'alpha', but applied to 'stroke'.
size	(Numeric) values mapped to relevant 'fill-size' attribute(s) (e.g., [marker.size](https://plot.ly/r/referen marker-size), [textfont.size](https://plot.ly/r/reference#scatter-textfont-size), and [error_x.width](https://plot.ly/r/reference#scatter-error_x-width)). The mapping from data values to symbols may be controlled using 'sizes', or avoided altogether via [I()] (e.g., 'size = I(30)').
sizes	A numeric vector of length 2 used to scale 'size' to pixels.
span	(Numeric) values mapped to relevant 'stroke-size' attribute(s) (e.g., [marker.line.width](https://plot.ly/ marker-line-width), [line.width](https://plot.ly/r/reference#scatter-line-width) for filled polygons, and [error_x.thickness](https://plot.ly/r/reference#scatter-error_x-thickness)) The mapping from data values to symbols may be controlled using 'spans', or avoided altogether via [I()] (e.g., 'span = I(30)').
spans	A numeric vector of length 2 used to scale 'span' to pixels.

symbol	(Discrete) values mapped to [marker.symbol](https://plot.ly/r/reference#scatter-marker-symbol). The mapping from data values to symbols may be controlled using 'symbols', or avoided altogether via [I()] (e.g., 'symbol = I("pentagon")'). Any [pch] value or [symbol name](https://plot.ly/r/reference#scatter-marker-symbol) may be used in this way.
symbols	A character vector of [pch] values or [symbol names](https://plot.ly/r/reference#scatter-marker-symbol).
linetype	(Discrete) values mapped to [line.dash](https://plot.ly/r/reference#scatter-line-dash). The mapping from data values to symbols may be controlled using 'linetypes', or avoided altogether via [I()] (e.g., 'linetype = I("dash")'). Any 'lty' (see [par]) value or [dash name](https://plot.ly/r/reference#scatter-line-dash) may be used in this way.
linetypes	A character vector of 'lty' values or [dash names](https://plot.ly/r/reference#scatter-line-dash)
split	(Discrete) values used to create multiple traces (one trace per value).
frame	(Discrete) values used to create animation frames.
width	Width in pixels (optional, defaults to automatic sizing).
height	Height in pixels (optional, defaults to automatic sizing).
source	a character string of length 1. Match the value of this string with the source argument in [event_data()] to retrieve the event data corresponding to a specific plot (shiny apps can have multiple plots).

Details

Unless 'type' is specified, this function just initiates a plotly object with 'global' attributes that are passed onto downstream uses of [add_trace()] (or similar). A [formula] must always be used when referencing column name(s) in 'data' (e.g. 'plot_ly(mtcars, x = ~wt)'). Formulas are optional when supplying values directly, but they do help inform default axis/scale titles (e.g., 'plot_ly(x = mtcars$wt)' vs 'plot_ly(x = ~mtcars$wt)')

Value

A plotly

Author(s)

Carson Sievert

References

<https://plotly-r.com/overview.html>

See Also

- For initializing a plotly-geo object: [plot_geo()]
- For initializing a plotly-mapbox object: [plot_mapbox()]
- For translating a ggplot2 object to a plotly object: [ggplotly()]
- For modifying any plotly object: [layout()], [add_trace()], [style()]
- For linked brushing: [highlight()]
- For arranging multiple plots: [subplot()], [crosstalk::bscols()]

- For inspecting plotly objects: [plotly_json()]
- For quick, accurate, and searchable plotly.js reference: [schema()]

Examples

```
pbmc_small %>%
   tidy %>%
   plot_ly(x=~nCount_RNA, y=~nFeature_RNA)
```

`pull` *Extract a single column*

Description

'pull()' is similar to '$'. It's mostly useful because it looks a little nicer in pipes, it also works with remote data frames, and it can optionally name the output.

Usage

```
pull(.data, var = -1, name = NULL, ...)
```

Arguments

`.data`	A data frame, data frame extension (e.g. a tibble), or a lazy data frame (e.g. from dbplyr or dtplyr). See *Methods*, below, for more details.
`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. The default returns the last column (on the assumption that's the column you've created most recently). This argument is taken by expression and supports quasiquotation (you can unquote column names and column locations).
`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'.

Methods

This function is a **generic**, which means that packages can provide implementations (methods) for other classes. See the documentation of individual methods for extra arguments and differences in behaviour.

The following methods are currently available in loaded packages: no methods found.

Examples

```
`%>%` = magrittr::`%>%`
pbmc_small %>% tidy %>% pull(groups)
```

`rename` *Rename columns*

Description

Rename individual variables using ‘new_name = old_name‘ syntax.

Usage

```
rename(.data, ...)
```

Arguments

`.data`	A data frame, data frame extension (e.g. a tibble), or a lazy data frame (e.g. from dbplyr or dtplyr). See *Methods*, below, for more details.
`...`	<[‘tidy-select‘][dplyr_tidy_select]> Use ‘new_name = old_name‘ to rename selected variables.

Value

An object of the same type as ‘.data‘. * Rows are not affected. * Column names are changed; column order is preserved * Data frame attributes are preserved. * Groups are updated to reflect new names.

Scoped selection and renaming

Use the three scoped variants ([rename_all()], [rename_if()], [rename_at()]) to renaming a set of variables with a function.

Methods

This function is a **generic**, which means that packages can provide implementations (methods) for other classes. See the documentation of individual methods for extra arguments and differences in behaviour.

The following methods are currently available in loaded packages:

See Also

Other single table verbs: `arrange()`, `filter()`, `mutate()`, `select()`, `slice()`, `summarise()`

Examples

```
`%>%` = magrittr::`%>%`
pbmc_small %>% tidy %>% rename(s_score = nFeature_RNA)
```

right_join *Right join datasets*

Description

Right join datasets

Usage

```
right_join(x, y, by = NULL, copy = FALSE, suffix = c(".x", ".y"), ...)
```

Arguments

x	tbls to join. (See dplyr)
y	tbls to join. (See dplyr)
by	A character vector of variables to join by. (See dplyr)
copy	If x and y are not from the same data source, and copy is TRUE, then y will be copied into the same src as x. (See dplyr)
suffix	If there are non-joined duplicate variables in x and y, these suffixes will be added to the output to disambiguate them. Should be a character vector of length 2. (See dplyr)
...	Data frames to combine (See dplyr)

Value

A tidyseurat object

Examples

```
`%>%` = magrittr::`%>%`

tt = pbmc_small %>% tidy
tt %>% right_join(tt %>% distinct(groups) %>% mutate(new_column = 1:2) %>% slice(1))
```

rowwise *Group input by rows*

Description

See [this repository](https://github.com/jennybc/row-oriented-workflows) for alternative ways to perform row-wise operations.

Usage

```
rowwise(.data)
```

Arguments

.data	Input data frame.

Details

‘rowwise()‘ is used for the results of [do()] when you create list-variables. It is also useful to support arbitrary complex operations that need to be applied to each row.

Currently, rowwise grouping only works with data frames. Its main impact is to allow you to work with list-variables in [summarise()] and [mutate()] without having to use `[[1]]`. This makes ‘summarise()‘ on a rowwise tbl effectively equivalent to [plyr::ldply()].

Value

A ‘tbl‘

A ‘tbl‘

Examples

```
`%>%` = magrittr::`%>%`
```

`select` *Subset columns using their names and types*

Description

Select (and optionally rename) variables in a data frame, using a concise mini-language that makes it easy to refer to variables based on their name (e.g. ‘a:f‘ selects all columns from ‘a‘ on the left to ‘f‘ on the right). You can also use predicate functions like [is.numeric] to select variables based on their properties.

Overview of selection features

“‘r, child = "man/rmd/overview.Rmd" “‘

Usage

```
select(.data, ...)
```

Arguments

`.data`	A data frame, data frame extension (e.g. a tibble), or a lazy data frame (e.g. from dbplyr or dtplyr). See *Methods*, below, for more details.
`...`	<[‘tidy-select‘][dplyr_tidy_select]> One or more unquoted expressions separated by commas. Variable names can be used as if they were positions in the data frame, so expressions like ‘x:y‘ can be used to select a range of variables.

Value

An object of the same type as ‘.data‘. 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. * Data frame attributes are preserved. * Groups are maintained; you can’t select off grouping variables.

Methods

This function is a **generic**, which means that packages can provide implementations (methods) for other classes. See the documentation of individual methods for extra arguments and differences in behaviour.

The following methods are currently available in loaded packages: no methods found.

See Also

Other single table verbs: `arrange()`, `filter()`, `mutate()`, `rename()`, `slice()`, `summarise()`

Examples

```
`%>%` = magrittr::`%>%`
pbmc_small %>% tidy %>% select(cell, orig.ident )
```

`separate` *Separate a character column into multiple columns with a regular expression or numeric locations*

Description

Given either a regular expression or a vector of character positions, ‘separate()‘ turns a single character column into multiple columns.

Usage

```
separate(
  data,
  col,
  into,
  sep = "[^[:alnum:]]+",
  remove = TRUE,
  convert = FALSE,
  extra = "warn",
  fill = "warn",
  ...
)
```

Arguments

`data`	A tidyseurat object
`col`	Column name or position. This is passed to [tidyselect::vars_pull()]. This argument is passed by expression and supports [quasiquotation][rlang::quasiquotation] (you can unquote column names or column positions).
`into`	Names of new variables to create as character vector. Use ‘NA‘ to omit the variable in the output.

sep	Separator between columns. If character, ‘sep‘ is interpreted as a regular expression. The default value is a regular expression that matches any sequence of non-alphanumeric values. If numeric, ‘sep‘ is interpreted as character positions to split at. Positive values start at 1 at the far-left of the string; negative value start at -1 at the far-right of the string. The length of ‘sep‘ should be one less than ‘into‘.
remove	If ‘TRUE‘, remove input column from output data frame.
convert	If ‘TRUE‘, will run [type.convert()] with ‘as.is = TRUE‘ on new columns. This is useful if the component columns are integer, numeric or logical. NB: this will cause string ‘"NA"‘s to be converted to ‘NA‘s.
extra	If ‘sep‘ is a character vector, this controls what happens when there are too many pieces. There are three valid options: * "warn" (the default): emit a warning and drop extra values. * "drop": drop any extra values without a warning. * "merge": only splits at most ‘length(into)‘ times
fill	If ‘sep‘ is a character vector, this controls what happens when there are not enough pieces. There are three valid options: * "warn" (the default): emit a warning and fill from the right * "right": fill with missing values on the right * "left": fill with missing values on the left
...	Additional arguments passed on to methods.

Value

A tidyseurat objector a tibble depending on input

See Also

[unite()], the complement, [extract()] which uses regular expression capturing groups.

Examples

```
un = pbmc_small %>% tidy %>% unite("new_col", c(orig.ident, groups))
un %>% separate(col = new_col, into= c("orig.ident", "groups"))
```

---

slice *Subset rows using their positions*

Description

‘slice()‘ lets you index rows by their (integer) locations. It allows you to select, remove, and duplicate rows. It is accompanied by a number of helpers for common use cases:

* ‘slice_head()‘ and ‘slice_tail()‘ select the first or last rows. * ‘slice_sample()‘ randomly selects rows. * ‘slice_min()‘ and ‘slice_max()‘ select rows with highest or lowest values of a variable.

If ‘.data‘ is a [grouped_df], the operation will be performed on each group, so that (e.g.) ‘slice_head(df, n = 5)‘ will select the first five rows in each group.

Usage

```
slice(.data, ..., .preserve = FALSE)
```

Arguments

.data	A data frame, data frame extension (e.g. a tibble), or a lazy data frame (e.g. from dbplyr or dtplyr). See *Methods*, below, for more details.
...	For ‘slice()‘: <[‘data-masking‘][dplyr_data_masking]> Integer row values. Provide either positive values to keep, or negative values to drop. The values provided must be either all positive or all negative. Indices beyond the number of rows in the input are silently ignored. For ‘slice_helpers()‘, these arguments are passed on to methods. 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. If the ‘prop‘ortion of a group size is not an integer, it is rounded down.
.preserve	when ‘FALSE‘ (the default), the grouping structure is recalculated based on the resulting data, otherwise it is kept as is.

Details

Slice does not work with relational databases because they have no intrinsic notion of row order. If you want to perform the equivalent operation, use [filter()] and [row_number()].

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. * Data frame attributes are preserved.

Methods

These function are **generic**s, which means that packages can provide implementations (methods) for other classes. See the documentation of individual methods for extra arguments and differences in behaviour.

Methods available in currently loaded packages:

* ‘slice()‘: no methods found. * ‘slice_head()‘: no methods found. * ‘slice_tail()‘: no methods found. * ‘slice_min()‘: no methods found. * ‘slice_max()‘: no methods found. * ‘slice_sample()‘: no methods found.

See Also

Other single table verbs: `arrange()`, `filter()`, `mutate()`, `rename()`, `select()`, `summarise()`

Examples

```
`%>%` = magrittr::`%>%`
pbmc_small %>% tidy %>% slice(1)
```

summarise *Summarise each group to fewer rows*

Description

'summarise()' creates a new data frame. It will have one (or more) rows for each combination of grouping variables; if there are no grouping variables, the output will have a single row summarising all observations in the input. It will contain one column for each grouping variable and one column for each of the summary statistics that you have specified.

'summarise()' and 'summarize()' are synonyms.

Usage

```
summarise(.data, ...)
```

Arguments

.data	A data frame, data frame extension (e.g. a tibble), or a lazy data frame (e.g. from dbplyr or dtplyr). See *Methods*, below, for more details.
...	<['tidy-eval'][dplyr_tidy_eval]> Name-value pairs of summary functions. The name will be the name of the variable in the result. The value can be: * A vector of length 1, e.g. 'min(x)', 'n()', or 'sum(is.na(y))'. * A vector of length 'n', e.g. 'quantile()'. * A data frame, to add multiple columns from a single expression.

Value

An object _usually_ of the same type as '.data'.

* The rows come from the underlying 'group_keys()'. * The columns are a combination of the grouping keys and the summary expressions that you provide. * If 'x' is grouped by more than one variable, the output will be another [grouped_df] with the right-most group removed. * If 'x' is grouped by one variable, or is not grouped, the output will be a [tibble]. * Data frame attributes are **not** preserved, because 'summarise()' fundamentally creates a new data frame.

Useful functions

* Center: [mean()], [median()] * Spread: [sd()], [IQR()], [mad()] * Range: [min()], [max()], [quantile()] * Position: [first()], [last()], [nth()], * Count: [n()], [n_distinct()] * Logical: [any()], [all()]

Backend variations

The data frame backend supports creating a variable and using it in the same summary. This means that previously created summary variables can be further transformed or combined within the summary, as in [mutate()]. However, it also means that summary variables with the same names as previous variables overwrite them, making those variables unavailable to later summary variables.

This behaviour may not be supported in other backends. To avoid unexpected results, consider using new names for your summary variables, especially when creating multiple summaries.

Methods

This function is a **generic**, which means that packages can provide implementations (methods) for other classes. See the documentation of individual methods for extra arguments and differences in behaviour.

The following methods are currently available in loaded packages:

See Also

Other single table verbs: `arrange()`, `filter()`, `mutate()`, `rename()`, `select()`, `slice()`

Examples

```
`%>%` = magrittr::`%>%`
pbmc_small %>% tidy %>% summarise(mean(nCount_RNA))
```

---

tidy *tidy for seurat*

Description

tidy for seurat

Usage

```
tidy(object)
```

Arguments

`object`	A Seurat object

Value

A tidyseurat object

---

unite *Unite multiple columns into one by pasting strings together*

Description

Convenience function to paste together multiple columns into one.

Usage

```
unite(data, col, ..., sep = "_", remove = TRUE, na.rm = FALSE)
```

Arguments

data	A data frame.
col	The name of the new column, as a string or symbol. This argument is passed by expression and supports [quasiquotation][rlang::quasiquotation] (you can unquote strings and symbols). The name is captured from the expression with [rlang::ensym()] (note that this kind of interface where symbols do not represent actual objects is now discouraged in the tidyverse; we support it here for backward compatibility).
...	<[‘tidy-select‘][tidyr_tidy_select]> Columns to unite
sep	Separator to use between values.
remove	If ‘TRUE‘, remove input columns from output data frame.
na.rm	If ‘TRUE‘, missing values will be remove prior to uniting each value.

Value

A tidyseurat objector a tibble depending on input

See Also

[separate()], the complement.

Examples

```
pbmc_small %>% tidy %>% unite("new_col", c(orig.ident, groups))
```

---

unnest *unnest*

Description

unnest

nest

Usage

```
unnest(
  .data,
  cols,
  ...,
  keep_empty = FALSE,
  ptype = NULL,
  names_sep = NULL,
  names_repair = "check_unique"
)

## Default S3 method:
unnest(
```

```
  .data,
  cols,
  ...,
  keep_empty = FALSE,
  ptype = NULL,
  names_sep = NULL,
  names_repair = "check_unique"
)

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

nest(.data, ...)

## Default S3 method:
nest(.data, ...)

## S3 method for class 'tidyseurat'
nest(.data, ...)
```

Arguments

.data	A tbl. (See tidyr)
cols	<[‘tidy-select‘][tidyr_tidy_select]> Columns to unnest. If you ‘unnest()‘ multiple columns, parallel entries must be of compatible sizes, i.e. they’re either equal or length 1 (following the standard tidyverse recycling rules).
...	Name-variable pairs of the form new_col = c(col1, col2, col3) (See tidyr)
keep_empty	See tidyr::unnest
ptype	See tidyr::unnest
names_sep	If ‘NULL‘, the default, the names will be left as is. In ‘nest()‘, inner names will come from the former outer names; in ‘unnest()‘, the new outer names will come from the inner names. If a string, the inner and outer names will be used together. In ‘nest()‘, the names of the new outer columns will be formed by pasting together the outer and the inner column names, separated by ‘names_sep‘. In ‘unnest()‘, the new inner names will have the outer names (+ ‘names_sep‘) automatically stripped. This makes ‘names_sep‘ roughly symmetric between nesting and unnesting.
names_repair	See tidyr::unnest

Value

A tidyseurat objector a tibble depending on input

A tidyseurat objector a tibble depending on input

Examples

```
library(dplyr)
pbmc_small %>% tidy %>% nest(data = -groups) %>% unnest(data)
```


```
library(dplyr)
pbmc_small %>% tidy %>% nest(data = -groups) %>% unnest(data)
```

Index