# Package ‘poorman’

April 13, 2020

**Type** Package  
**Title** A Poor Man's Base R Copy of 'dplyr' Verbs  
**Version** 0.1.10  
**Description** A simple replication of key 'dplyr' verbs using only base R.  
**URL** [https://github.com/nathaneastwood/poorman](https://github.com/nathaneastwood/poorman)  
**BugReports** [https://github.com/nathaneastwood/poorman/issues](https://github.com/nathaneastwood/poorman/issues)  
**Depends** R (>= 3.4)  
**Suggests** knitr, roxygen2, tinytest  
**License** MIT + file LICENSE  
**Encoding** UTF-8  
**LazyData** true  
**RoxygenNote** 7.0.2  
**NeedsCompilation** no  
**Author** Nathan Eastwood [aut, cre]  
**Maintainer** Nathan Eastwood <nathan.eastwood@icloud.com>  
**Repository** CRAN  
**Date/Publication** 2020-04-13 14:10:02 UTC

---

## R topics documented:

<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>arrange</td>
<td>2</td>
</tr>
<tr>
<td>filter</td>
<td>2</td>
</tr>
<tr>
<td>filter_joins</td>
<td>3</td>
</tr>
<tr>
<td>groups</td>
<td>4</td>
</tr>
<tr>
<td>group_vars</td>
<td>5</td>
</tr>
<tr>
<td>joins</td>
<td>5</td>
</tr>
<tr>
<td>mutate</td>
<td>6</td>
</tr>
<tr>
<td>pipe</td>
<td>7</td>
</tr>
<tr>
<td>print.grouped_data</td>
<td>8</td>
</tr>
<tr>
<td>pull</td>
<td>9</td>
</tr>
</tbody>
</table>
filter

Return rows with matching conditions

Description
Use `filter()` to choose rows/cases where conditions are TRUE.

Usage
```
filter(.data, ...)
```

Arguments
- `.data` A `data.frame`.
- `...` A comma separated vector of unquoted name(s) to define the condition.

Value
A `data.frame`.

Examples
```
filter(mtcars, mpg > 20)
```
Arguments
.data
A data.frame.
...
Logical predicated defined in terms of the variables in .data. Multiple conditions are combined with &. Arguments within ... are automatically quoted and evaluated within the context of the data.frame.

Value
A data.frame.

Useful filter functions
- ==, >, >=, etc.
- & | ! xor()
- is.na()

Examples
filter(mtcars, am == 1)
mtcars %>% filter(cyl == 4)
mtcars %>% filter(cyl <= 5 & am > 0)
mtcars %>% filter(cyl == 4 | cyl == 8)
mtcars %>% filter(!(cyl %in% c(4, 6)), am != 0)

Description
- semi_join() return all rows from x with a match in y.
- anti_join() return all rows from x without a match in y.

Usage
anti_join(x, y, by = NULL)
semi_join(x, y, by = NULL)

Arguments
x, y
The data.frames to join.
by
A character vector of variables to join by. If NULL, the default, *_join() will do a natural join, using all variables with common names across the two tables. A message lists the variables so that you can check they’re right (to suppress the message, simply explicitly list the variables that you want to join).
Examples

table1 <- data.frame(
  pupil = rep(1:3, each = 2),
  test = rep(c("A", "B"), 3),
  score = c(60, 70, 65, 80, 85, 70),
  stringsAsFactors = FALSE
)
table2 <- table1[c(1, 3, 4), ]
table1 %>% anti_join(table2, by = c("pupil", "test"))
table1 %>% semi_join(table2, by = c("pupil", "test"))

groups

Group by one or more variables

Description

Determine the groups within a data.frame to perform operations on. ungroup() removes the grouping levels.

Usage

group_by(.data, ...)

ungroup(x, ...)

Arguments

.data data.frame. The data to group.
...
One or more unquoted column names to group/ungroup the data by.

x A data.frame.

Value

When using group_by(), a data.frame, grouped by the grouping variables.
When using ungroup(), data.frame.

Examples

group_by(mtcars, am, cyl)
ungroup(mutate(group_by(mtcars, am, cyl), sumMpg = sum(mpg)))
mtcars %>%
  group_by(am, cyl) %>%
  mutate(sumMpg = sum(mpg)) %>%
  ungroup()
mtcars %>%
  group_by(carb) %>%
```
group_vars

    filter(any(gear == 5))

---

<table>
<thead>
<tr>
<th>group_vars</th>
<th>Return grouping variables</th>
</tr>
</thead>
</table>

Description

Return grouping variables

Usage

```r
group_vars(x)
```

Arguments

- `x`: A `data.frame`.

Value

A character vector of group names.

Examples

```r
df <- mtcars %>% group_by(am, cyl)
group_vars(df)
```

---

<table>
<thead>
<tr>
<th>joins</th>
<th>Join two <code>data.frame</code>s together</th>
</tr>
</thead>
</table>

Description

Join two `data.frame`s together

Usage

```r
inner_join(x, y, by = NULL, suffix = c(".x", ".y"))
left_join(x, y, by = NULL, suffix = c(".x", ".y"))
right_join(x, y, by = NULL, suffix = c(".x", ".y"))
full_join(x, y, by = NULL, suffix = c(".x", ".y"))
```
mutate

Create or transform variables

Arguments

x, y
The data.frames to join.

by
A character vector of variables to join by. If NULL, the default, *_join() will do
a natural join, using all variables with common names across the two tables. A
message lists the variables so that you can check they’re right (to suppress the
message, simply explicitly list the variables that you want to join).

To join by different variables on x and y use a named vector. For example, by =
c("a" = "b") will match x.a to y.b.

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.

Description

mutate() adds new variables and preserves existing ones; transmute() adds new variables and
drops existing ones. Both functions preserve the number of rows of the input. New variables
overwrite existing variables of the same name.

Usage

mutate(.data, ...)

transmute(.data, ...)

Arguments

.data
A data.frame.

...  
Name-value pairs of expressions, each with length 1L. The name of each argu-
ment will be the name of a new column and the value will be its corresponding
value. Use a NULL value in mutate to drop a variable. New variables overwrite
existing variables of the same name.

Examples

mutate(mtcars, mpg2 = mpg * 2)
mtcars %>% mutate(mpg2 = mpg * 2)
mtcars %>% mutate(mpg2 = mpg * 2, cyl2 = cyl * 2)

# Newly created variables are available immediately
mtcars %>% mutate(mpg2 = mpg * 2, mpg4 = mpg2 * 2)

# You can also use mutate() to remove variables and modify existing variables
mtcars %>% mutate(
  mpg = NULL,
disp = disp * 0.0163871)  # convert to litres
# mutate() vs transmute --------------------------
# mutate() keeps all existing variables
mtcars %>%
  mutate(displ_l = disp / 61.0237)

# transmute keeps only the variables you create
mtcars %>%
  transmute(displ_l = disp / 61.0237)

---

**Description**

Pipe an object forward into a function or call expression.

**Usage**

```
lhs %>% rhs
```

**Arguments**

- `lhs` The result you are piping.
- `rhs` Where you are piping the result to.

**Details**

Unlike the `magrittr` pipe, you must supply an actual function instead of just a function name. For example `mtcars %>% head` will not work, but `mtcars %>% head()` will.

**Examples**

```
mtcars %>% head()
mtcars %>% select(mpg)
```
print.grouped_data

Print a grouped data.frame

Description

A print method for grouped data.frames. Uses the standard print.data.frame() method but also reports the groups.

Usage

## S3 method for class 'grouped_data'
print(
  x,
  ..., 
  digits = NULL,
  quote = FALSE,
  right = TRUE,
  row.names = TRUE,
  max = NULL
)

Arguments

x  An object of class grouped_data.

... Additional arguments to print().

digits the minimum number of significant digits to be used: see print.default.

quote logical, indicating whether or not entries should be printed with surrounding quotes.

right logical, indicating whether or not strings should be right-aligned. The default is right-alignment.

row.names logical (or character vector), indicating whether (or what) row names should be printed.

max numeric or NULL, specifying the maximal number of entries to be printed. By default, when NULL,getOption("max.print") used.

Examples

mtcars %>% group_by(cyl, am) %>% print()
pull

Pull out a single variable

Description

This is a direct replacement for [[.data.frame.

Usage

pull(.data, var = -1)

Arguments

.data A data.frame.
.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).

Examples

mtcars %>% pull(-1)
mtcars %>% pull(1)
mtcars %>% pull(cyl)

relocate

Select/relocate variables by name

Description

Choose or relocate variables from a data.frame. select() keeps only the variables you mention; relocate() keeps all the variables.

Usage

relocate(.data, ..., .before = NULL, .after = NULL)

select(.data, ...)

### Arguments

- `.data` A data.frame.
- `...` The name(s) of the column(s) to select.
- `.before, .after` Destination of the columns selected by `...`. Supplying neither will move the columns to the left-hand side whereas supplying both will result in an error.

### Value

A data.frame.

### Useful functions

There are a number of special functions which are designed to work in `select()` and `relocate()`:

- `starts_with()`, `ends_with()`, `contains()`
- `matches()`
- `num_range()`
- `everything()`

### Examples

```r
select(mtcars, mpg, cyl)
select(mtcars, MilesPerGallon = mpg, Cylinders = cyl)
mtcars %>% select(mpg)
mtcars %>% select(mpg, cyl)
iris %>% select(contains("Petal"))

df <- as.data.frame(matrix(runif(100), nrow = 10))
df <- as.data.frame(df[order(c(3, 4, 7, 1, 9, 8, 5, 2, 6, 10))])
df %>% select(num_range("V", 4:6))

mtcars %>% relocate(ends_with("p"), .before = mpg)
```

---

### Rename

#### Rename columns

**Description**

`rename()` changes the names of individual variables using `new_name = old_name` syntax.

**Usage**

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

.data  A data.frame

...  Comma separated key-value pairs in the form of new_name = old_name to re-name selected variables.

Value

A data.frame

Examples

rename(mtcars, MilesPerGallon = mpg)
rename(mtcars, Cylinders = cyl, Gears = gear)
mtcars %>% rename(MilesPerGallon = mpg)

rownames  Tools for working with row names

Description

Tools for working with row names

Usage

rownames_to_column(.data, var = "rowname")

Arguments

.data  A data.frame.

var  character(1). The name of the column to use for row names.

Value

A data.frame

Examples

mtcars %>% rownames_to_column()
**select_helpers**  

**Select Helpers**

**Description**

These functions allow you to select variables based on their names.

- **starts_with()**: Starts with a prefix.
- **ends_with()**: Ends with a prefix.
- **contains()**: Contains a literal string.
- **matches()**: Matches a regular expression.
- **all_of()**: Matches variable names in a character vector. All names must be present, otherwise an error is thrown.
- **any_of()**: The same as all_of() except it doesn’t throw an error.
- **everything()**: Matches all variables.
- **last_col()**: Select the last variable, possibly with an offset.

**Usage**

```r
starts_with(match, ignore.case = TRUE, vars = peek_vars())
ends_with(match, ignore.case = TRUE, vars = peek_vars())
contains(match, ignore.case = TRUE, vars = peek_vars())
matches(match, ignore.case = TRUE, perl = FALSE, vars = peek_vars())
num_range(prefix, range, width = NULL, vars = peek_vars())
all_of(x, vars = peek_vars())
any_of(x, vars = peek_vars())
everything(vars = peek_vars())
last_col(offset = 0L, vars = peek_vars())
```

**Arguments**

- **match** character(n). If length > 1, the union of the matches is taken.
- **ignore.case** logical(1). If TRUE, the default, ignores case when matching names.
- **vars** character(n). A character vector of variable names. When called from inside selecting functions such as select(), these are automatically set to the names of the table.
Choose rows by position

Description

Choose rows by their original position in the data.frame. Grouped data.frames use the position within each group.

Usage

slice(.data, ...)

perl  logical(1). Should Perl-compatible regexps be used?
prefix  A prefix which starts the numeric range.
range  integer(n). A sequence of integers, e.g. 1:5.
width  numeric(1). Optionally, the "width" of the numeric range. For example, a range of 2 gives "01", a range of three "001", etc.
x  character(n). A vector of column names.
offset  integer(1). Select the nth variable from the end of the data.frame.

Value

An integer vector giving the position of the matched variables.

See Also

select(), relocate()
Arguments

.data A data.frame.

... Integer row values. Provide either positive values to keep, or negative values to drop. The values provided must be either all positive or negative. Indices beyond the number of rows in the input are silently ignored.

Examples

slice(mtcars, c(1, 2, 3))
mtcars %>% slice(1:3)

summarise Reduce multiple values down to a single value

Description

Create one or more scalar variables summarising the variables of an existing data.frame. Grouped data.frames will result in one row in the output for each group.

Usage

summarise(.data, ...)

summarize(.data, ...)

Arguments

.data A data.frame.

... Name-value pairs of summary functions. The name will be the name of the variable in the result. The value should be an expression that returns a single value, e.g. min(x).

Details

summarise() and summarize() are synonyms.

Examples

summarise(mtcars, mean(mpg))
summarise(mtcars, meanMpg = mean(mpg), sumMpg = sum(mpg))
mtcars %>% summarise(mean(mpg))
Index

%% (pipe), 7
all_of (select_helpers), 12
anti_join (filter_joins), 3
any_of (select_helpers), 12
arrange, 2
contains (select_helpers), 12
contains(), 10
ends_with (select_helpers), 12
ends_with(), 10
everything (select_helpers), 12
everything(), 10
filter, 2
filter_joins, 3
full_join (joins), 5
group_by (groups), 4
group_by(), 4
group_vars, 5
groups, 4
inner_join (joins), 5
joins, 5
last_col (select_helpers), 12
left_join (joins), 5
matches (select_helpers), 12
matches(), 10
mutate, 6
num_range (select_helpers), 12
num_range(), 10
pipe, 7
print, 8
print.default, 8
print.grouped_data, 8
pull, 9
relocate, 9
relocate(), 13
rename, 10
right_join (joins), 5
rownames, 11
rownames_to_column (rownames), 11
select (relocate), 9
select(), 13
select_helpers, 12
semi_join (filter_joins), 3
slice, 13
starts_with (select_helpers), 12
starts_with(), 10
summarise, 14
summarize (summarise), 14
transmute (mutate), 6
ungroup (groups), 4
ungroup(), 4