

# Package ‘hacksaw’

July 18, 2020

**Title** Additional Tools for Splitting and Cleaning Data

**Version** 0.0.1

**Description** Move between data frames and lists more efficiently with precision splitting via 'dplyr' verbs. Easily cast variables to different data types. Keep rows with NAs. Shift row values.

**License** MIT + file LICENSE

**Encoding** UTF-8

**LazyData** true

**RoxygenNote** 7.1.0

**Imports** dplyr, purrr, rlang, utils, tidyselect, tibble, zeallot, magrittr

**Suggests** testthat, knitr, rmarkdown, tidyr

**NeedsCompilation** no

**Author** David Ranzolin [aut, cre, cph]

**Maintainer** David Ranzolin <daranzolin@gmail.com>

**Repository** CRAN

**Date/Publication** 2020-07-18 09:40:02 UTC

## R topics documented:

cast_character . . . . .	2
filter_pattern . . . . .	2
filter_split . . . . .	3
keep_na . . . . .	4
keep_pattern . . . . .	5
pluck_when . . . . .	6
shift_row_values . . . . .	6
var_max . . . . .	7
var_min . . . . .	8
<b>Index</b>	<b>9</b>

---

cast_character	<i>Cast columns to a specified data type</i>
----------------	--

---

**Description**

Cast columns to a specified data type

**Usage**

```
cast_character(.data, ...)
```

```
cast_numeric(.data, ...)
```

```
cast_logical(.data, ...)
```

**Arguments**

.data	a table of data.
...	A selection of columns.

**Value**

a data frame.

**Examples**

```
library(dplyr)
df <- tibble(x = 1:3, y = as.character(1:3), z = c(0, 0, 1))
df %>% cast_character(x)
df %>% cast_numeric(y)
df %>% cast_logical(z)
```

---

filter_pattern	<i>Grep and filter a data frame by pattern</i>
----------------	--

---

**Description**

Grep and filter a data frame by pattern

**Usage**

```
filter_pattern(.data, col, pattern, ...)
```

**Arguments**

.data	a table of data.
col	a variable.
pattern	string containing a regular expression to be matched in the given character vector.
...	additional arguments passed to grepl

**Value**

a data frame.

**Examples**

```
library(dplyr)
starwars %>% filter_pattern(homeworld, "oo")
```

---

filter\_split

*Perform various operations before splitting*

---

**Description**

Evaluate expressions over a data frame, resulting in a list.

**Usage**

```
filter_split(.data, ...)
select_split(.data, ...)
mutate_split(.data, ...)
distinct_split(.data, ..., simplify = TRUE)
transmute_split(.data, ..., simplify = TRUE)
slice_split(.data, ...)
pull_split(.data, ...)
eval_split(.data, ...)
precision_split(.data, ...)
```

**Arguments**

.data            A table of data.  
 ...             Expressions to be evaluated.  
 simplify        Boolean, whether to unlist the returned split.

**Value**

A list.

**Examples**

```
library(dplyr)
mtcars %>% filter_split(cyl == 4, cyl == 6)
iris %>% select_split(starts_with("Sepal"), starts_with("Petal"))
mtcars %>% mutate_split(mpg2 = mpg^2, mpg3 = mpg^3)
mtcars %>% distinct_split(cyl, carb)
mtcars %>% transmute_split(mpg^2, sqrt(mpg))
mtcars %>% slice_split(1:10, 11:20)
mtcars %>% pull_split(mpg, hp)
mtcars %>% eval_split(select(mpg, hp), filter(mpg>25), mutate(mpg2 = mpg^2))
mtcars %>% precision_split(mpg > 25)
```

---

keep_na	<i>Keep rows containing missing values</i>
---------	--

---

**Description**

Keep rows containing missing values

**Usage**

```
keep_na(.data, ...)
```

**Arguments**

.data            A table of data.  
 ...             A selection of columns. If empty, all columns are selected.

**Value**

A data frame.

**Examples**

```
library(dplyr)
df <- tibble(x = c(1, 2, NA, NA), y = c("a", NA, "b", NA))
df %>% keep_na()
df %>% keep_na(x)

vars <- "y"
df %>% keep_na(x, any_of(vars))
```

---

keep_pattern	<i>Grep, keep or discard a list or vector by pattern</i>
--------------	--

---

**Description**

Grep, keep or discard a list or vector by pattern

**Usage**

```
keep_pattern(x, pattern, ...)
discard_pattern(x, pattern, ...)
```

**Arguments**

x	a list or vector.
pattern	string containing a regular expression to be matched in the given character vector.
...	additional arguments passed to grepl.

**Value**

A list.

**Examples**

```
l <- list("David", "Daniel", "Damien", "Eric", "Jared", "Zach")
l %>% keep_pattern("^D")
l %>% discard_pattern("^D")
```

---

pluck\_when                      *Pluck a value based on other criteria*

---

### Description

Pluck a value based on other criteria

### Usage

```
pluck_when(.x, .p, .i = 1, .else = NA)
```

### Arguments

.x                      Vector from which to select value.  
.p                      Logical expression.  
.i                      First TRUE index to return.  
.else                    If no matches from .p, value to return.

### Value

A vector of length 1.

### Examples

```
library(dplyr)
df <- tibble(
  id = c(1, 1, 1, 2, 2, 2, 3, 3),
  tested = c("no", "no", "yes", "no", "no", "no", "yes", "yes"),
  year = c(2015:2017, 2010:2012, 2019:2020)
)
df %>%
  group_by(id) %>%
  mutate(year_first_tested = pluck_when(year, tested == "yes"))
```

---

shift\_row\_values                      *Shift row values left or right*

---

### Description

Shift row values left or right

### Usage

```
shift_row_values(.data, .dir = "left", at = NULL)
```

**Arguments**

.data            a table of data.  
 .dir            the shift direction as a string, one of "left" or "right".  
 at              the row indices at which to shift.

**Value**

a data frame.

**Examples**

```
library(dplyr)
df <- tibble(
  s = c(NA, 1, NA, NA),
  t = c(NA, NA, 1, NA),
  u = c(NA, NA, 2, 5),
  v = c(5, 1, 9, 2),
  x = c(1, 5, 6, 7),
  y = c(NA, NA, 8, NA),
  z = 1:4
)
df %>% shift_row_values()
df %>% shift_row_values(at = 1:3)
df %>% shift_row_values(at = 1:2, .dir = "right")
```

---

var\_max

*Return the indices of n max values of a variable*


---

**Description**

Return the indices of n max values of a variable

**Usage**

```
var_max(var, n = 6)
```

**Arguments**

var            the variable to use.  
 n              number of rows to return.

**Examples**

```
var_max(1:10)
```

---

var_min	<i>Return the indices of n min values of a variable</i>
---------	---

---

**Description**

Return the indices of n min values of a variable

**Usage**

```
var_min(var, n = 6)
```

**Arguments**

var	the variable to use.
n	number of rows to return.

**Examples**

```
var_min(1:10)
```



# Index

cast\_character, 2  
cast\_logical (cast\_character), 2  
cast\_numeric (cast\_character), 2  
  
discard\_pattern (keep\_pattern), 5  
distinct\_split (filter\_split), 3  
  
eval\_split (filter\_split), 3  
  
filter\_pattern, 2  
filter\_split, 3  
  
keep\_na, 4  
keep\_pattern, 5  
  
mutate\_split (filter\_split), 3  
  
pluck\_when, 6  
precision\_split (filter\_split), 3  
pull\_split (filter\_split), 3  
  
select\_split (filter\_split), 3  
shift\_row\_values, 6  
slice\_split (filter\_split), 3  
  
transmute\_split (filter\_split), 3  
  
var\_max, 7  
var\_min, 8