Package 'janitor'

October 3, 2016

Title Simple Tools for Examining and Cleaning Dirty Data

Version 0.2.0

Description The main janitor functions can: perfectly format data.frame column names; isolate duplicate records; and provide quick one- and two-variable tabulations (i.e., frequency tables and crosstabs). Other janitor functions nicely format the results of these tabulations. These tabulate-and-report functions approximate popular features of SPSS and Microsoft Excel. This package follows the principles of the ``tidyverse" and works well with the pipe function %>%. janitor was built with beginning-to-intermediate R users in mind and is optimized for user-friendliness. Advanced R users can already do everything covered here, but with janitor they can do it faster and save their thinking for the fun stuff.

URL https://github.com/sfirke/janitor

BugReports https://github.com/sfirke/janitor/issues

Depends R (>= 3.1.2)

Imports dplyr, tidyr, magrittr

License MIT + file LICENSE

LazyData true

RoxygenNote 5.0.1

Suggests testthat, knitr, rmarkdown

VignetteBuilder knitr

NeedsCompilation no

Author Sam Firke [aut, cre],
Chris Haid [ctb]

Maintainer Sam Firke <samuel.firke@gmail.com>

Repository CRAN

Date/Publication 2016-10-03 08:17:44

R topics documented:

add_totals_col . . . 2
add_totals_row . . . 3
adorn_crosstab . . . 3
clean_names . . . 4
convert_to_NA . . . 5
excel_numeric_to_date . . . 6
get_dupes . . . 6
janitor . . . 7
ns_to_percents . . . 8
remove_empty_cols . . . 9
remove_empty_rows . . . 9
top_levels . . . 10
use_first_valid_of . . . 11

Index **12**

`add_totals_col` *Append a totals column to a data.frame.*

Description

This function excludes the first column of the input data.frame, assuming that it contains a descriptive variable not to be summed.

Usage

```
add_totals_col(dat, na.rm = TRUE)
```

Arguments

`dat`	an input data.frame with numeric values in all columns beyond the first.
`na.rm`	should missing values (including NaN) be omitted from the calculations?

Value

Returns a data.frame with a totals column, consisting of "Total" in the first row and row sums in the others.

Examples

```
library(dplyr) # for the %>% pipe
mtcars %>%
  crosstab(am, cyl) %>%
  add_totals_col
```

add_totals_row *Append a totals row to a data.frame.*

Description

This function excludes the first column of the input data.frame, assuming that it contains a descriptive variable not to be summed.

Usage

```
add_totals_row(dat, na.rm = TRUE)
```

Arguments

dat	an input data.frame with numeric values in all columns beyond the first.
na.rm	should missing values (including NaN) be omitted from the calculations?

Value

Returns a data.frame with a totals row, consisting of "Total" in the first column and column sums in the others.

Examples

```
library(dplyr) # for the %>% pipe
mtcars %>%
  crosstab(am, cyl) %>%
  add_totals_row
```

adorn_crosstab *Add formatting to a crosstabulation table.*

Description

Designed to run on the output of a call to `crosstab`, this adds formatting, percentage sign, Ns, totals row/column, and custom rounding to a table of numeric values. The result is no longer clean data, but it saves time in reporting table results.

Usage

```
adorn_crosstab(crosstab, denom = "row", show_n = TRUE, digits = 1,
  show_totals = FALSE, rounding = "half to even")
```

Arguments

crosstab	a data.frame with row names in the first column and numeric values in all other columns. Usually the piped-in result of a call to crosstab that included the argument percent = "none".
denom	the denominator to use for calculating percentages. One of "row", "col", or "all".
show_n	should counts be displayed alongside the percentages?
digits	how many digits should be displayed after the decimal point?
show_totals	display a totals summary? Will be a row, column, or both depending on the value of denom.
rounding	method to use for truncating percentages - either "half to even", the base R default method, or "half up", where 14.5 rounds up to 15.

Value

Returns a data.frame.

Examples

```
library(dplyr) # for the %>% pipe
mtcars %>%
 crosstab(gear, cyl) %>%
 adorn_crosstab(denom = "all")

# showing with all parameters
mtcars %>%
 crosstab(gear, cyl) %>%
 adorn_crosstab(., denom = "col", rounding = "half up", show_n = FALSE, digits = 2)
mtcars %>%
 crosstab(cyl, am) %>%
 adorn_crosstab(., denom = "all", digits = 0, rounding = "half up")
```

---

clean_names *Cleans names of a data.frame.*

Description

Resulting names are unique and consist only of the _ character, lowercase letters, and numbers.

Usage

```
clean_names(dat)
```

Arguments

dat	the input data.frame.

Value

Returns the data.frame with clean names.

Examples

```
# not run:
# clean_names(poorly_named_df)

# library(dplyr) ; library(readxl)
# not run:
# readxl("messy_excel_file.xlsx") %>% clean_names()
```

---

convert_to_NA *Convert string values to true* NA *values.*

Description

Converts instances of user-specified strings into NA. Can operate on either a single vector or an entire data.frame.

Usage

```
convert_to_NA(dat, strings)
```

Arguments

dat	vector or data.frame to operate on.
strings	character vector of strings to convert.

Value

Returns a cleaned object. Can be a vector, data.frame, or tibble::tbl_df depending on the provided input.

Examples

```
convert_to_NA(mtcars, "4") # a silly example;
# mtcars has no string NA values, but this will convert 4s to NA

# a more typical call would be (not run):
# convert_to_NA(my_df, c("NA", "#N/A", "N/A", "n/a", "#NAME?"))
# catches common strings that should be NA

convert_to_NA(letters, c("b", "d"))
```

excel_numeric_to_date *Convert dates encoded as serial numbers to Date class.*

Description

Converts numbers like `42370` into date values like `2016-01-01`.

Defaults to the modern Excel date encoding system. However, Excel for Mac 2008 and earlier Mac versions of Excel used a different date system. To determine what platform to specify: if the date 2016-01-01 is represented by the number 42370 in your spreadsheet, it's the modern system. If it's 40908, it's the old Mac system. More on date encoding systems at http://support.office.com/en-us/article/Date-calculations-in-Excel-e7fe7167-48a9-4b96-bb53-5612a800b487.

Usage

```
excel_numeric_to_date(date_num, date_system = "modern")
```

Arguments

`date_num`	numeric vector of serial numbers to convert.
`date_system`	the date system, either `"modern"` or `"mac pre-2011"`.

Value

Returns a vector of class Date.

Examples

```
excel_numeric_to_date(40000)
```

get_dupes *Get rows of a* `data.frame` *with identical values for the specified variables.*

Description

For hunting duplicate records during data cleaning. Specify the data.frame and the variable combination to search for duplicates and get back the duplicated rows.

Usage

```
get_dupes(dat, ...)
```

Arguments

`dat`	the input data.frame.
`...`	unquoted variable names to search for duplicates.

Value

Returns a data.frame (actually a `tbl_df`) with the full records where the specified variables have duplicated values, as well as a variable `dupe_count` showing the number of rows sharing that combination of duplicated values.

Examples

```
get_dupes(mtcars, mpg, hp)
# or called with magrittr pipe %>% :
library(dplyr)
mtcars %>% get_dupes(wt)
```

`janitor` *janitor*

Description

janitor has simple little tools for examining and cleaning dirty data.

Main functions

The main janitor functions can: perfectly format ugly `data.frame` column names; isolate duplicate records for further study; and provide quick one- and two-variable tabulations (i.e., frequency tables and crosstabs) that improve on the base R function `table()`.

Other functions in the package can format for reporting the results of these tabulations. These tabulate-and-report functions approximate popular features of SPSS and Microsoft Excel.

Package context

This package follows the principles of the "tidyverse" and in particular works well with the `%>%` pipe function.

janitor was built with beginning-to-intermediate R users in mind and is optimized for user-friendliness. Advanced users can already do everything covered here, but they can do it faster with janitor and save their thinking for more fun tasks.

`ns_to_percents` *Convert a numeric data.frame to row-, column-, or totals-wise percentages.*

Description

This function excludes the first column of the input data.frame, assuming that it contains a descriptive variable.

Usage

```
ns_to_percents(dat, denom = "row", na.rm = TRUE, total_n = NULL)
```

Arguments

`dat`	a data.frame with row names in the first column and numeric values in all other columns.
`denom`	the denominator to use for calculating percentages. One of "row", "col", or "all".
`na.rm`	should missing values (including NaN) be omitted from the calculations?
`total_n`	an optional number to use as the denominator when calculating table-level percentages (when denom = "all"). Supply this if your input data.frame `dat` has values that would throw off the denominator if they were included, e.g., if there's a totals row appended to the bottom of the table.

Value

Returns a data.frame of percentages, expressed as numeric values between 0 and 1.

Examples

```
library(dplyr) # for the %>% pipe
mtcars %>%
  crosstab(am, cyl) %>%
  ns_to_percents(denom = "all")

  # when total_n is needed
  mtcars %>%
  crosstab(am, cyl) %>%
  add_totals_row() %>% # add a totals row that should not be included in the denominator
  ns_to_percents(denom = "all", total_n = nrow(mtcars)) # specify correct denominator
```

remove_empty_cols *Removes empty columns from a data.frame.*

Description

Removes all columns from a data.frame that are composed entirely of NA values.

Usage

```
remove_empty_cols(dat)
```

Arguments

dat	the input data.frame.

Value

Returns the data.frame with no empty columns.

Examples

```
# called with magrittr pipe %>% :
# library(dplyr)
# not run:
# dat %>% remove_empty_cols
```

remove_empty_rows *Removes empty rows from a data.frame.*

Description

Removes all rows from a data.frame that are composed entirely of NA values.

Usage

```
remove_empty_rows(dat)
```

Arguments

dat	the input data.frame.

Value

Returns the data.frame with no empty rows.

Examples

```
# called with magrittr pipe %>% :
# library(dplyr)
# not run:
# dat %>% remove_empty_rows
```

`top_levels` *Generate a frequency table of a factor grouped into top-n, bottom-n, and all other levels.*

Description

Get a frequency table of a factor variable, grouped into categories by level.

Usage

```
top_levels(input_vec, n = 2, show_na = FALSE, sort = FALSE)
```

Arguments

`input_vec`	the factor variable to tabulate.
`n`	number of levels to include in top and bottom groups
`show_na`	should cases where the variable is NA be shown?
`sort`	should the resulting table be sorted in descending order?

Value

Returns a data.frame (actually a `tbl_df`) with the frequencies of the grouped, tabulated variable. Includes counts and percentages, and valid percentages (calculated omitting `NA` values, if present in the vector and `show_na = TRUE`.)

Examples

```
top_levels(as.factor(mtcars$hp), 2)
```

`use_first_valid_of` *Returns first non-NA value from a set of vectors.*

Description

At each position of the input vectors, iterates through in order and returns the first non-NA value. This is a robust replacement of the common `ifelse(!is.na(x), x, ifelse(!is.na(y), y, z))`. It's more readable and handles problems like `ifelse`'s inability to work with dates in this way.

Usage

```
use_first_valid_of(..., if_all_NA = NA)
```

Arguments

`...`	the input vectors. Order matters: these are searched and prioritized in the order they are supplied.
`if_all_NA`	what value should be used when all of the vectors return `NA` for a certain index? Default is NA.

Value

Returns a single vector with the selected values.

Examples

```
x <- c(1, NA, NA); y <- c(2, 2, NA); z <- c(3, 3, 3)
use_first_valid_of(x, y, z)
use_first_valid_of(y, x, if_all_NA = 0)
```

Index

add_totals_col, 2
add_totals_row, 3
adorn_crosstab, 3

clean_names, 4
convert_to_NA, 5

excel_numeric_to_date, 6

get_dupes, 6

janitor, 7
janitor-package (janitor), 7

ns_to_percents, 8

remove_empty_cols, 9
remove_empty_rows, 9

top_levels, 10

use_first_valid_of, 11