Package 'janitor'

July 31, 2018

Title Simple Tools for Examining and Cleaning Dirty Data

Version 1.1.1

Description The main janitor functions can: perfectly format data.frame column names; provide quick counts of variable combinations (i.e., frequency tables and crosstabs); and isolate duplicate records. Other janitor functions nicely format the tabulation results. 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 (>= 0.7.0), tidyr (>= 0.7.0), snakecase (>= 0.9.0), magrittr, purrr, rlang

License MIT + file LICENSE

LazyData true

RoxygenNote 6.0.1.9000

Suggests testthat, knitr, rmarkdown

VignetteBuilder knitr

NeedsCompilation no

Author Sam Firke [aut, cre],
Chris Haid [ctb],
Ryan Knight [ctb],
Bill Denney [ctb]

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

Repository CRAN

Date/Publication 2018-07-31 04:30:02 UTC

R topics documented:

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

Description

This function is deprecated, use `adorn_totals` instead.

Usage

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

Arguments

`dat`	an input data.frame with at least one numeric column.
`na.rm`	should missing values (including NaN) be omitted from the calculations?

Value

Returns a data.frame with a totals column containing row-wise sums.

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

Description

This function is deprecated, use `adorn_totals` instead.

Usage

```
add_totals_row(dat, fill = "-", na.rm = TRUE)
```

Arguments

`dat`	an input data.frame with at least one numeric column.
`fill`	if there are more than one non-numeric columns, what string should fill the bottom row of those columns?
`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.

adorn_crosstab *Add presentation formatting to a crosstabulation table.*

Description

This function is deprecated, use the `adorn_` family of functions instead.

Usage

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

Arguments

`dat`	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.

`adorn_ns` *Add underlying Ns to a tabyl displaying percentages.*

Description

This function adds back the underlying Ns to a `tabyl` whose percentages were calculated using `adorn_percentages()`, to display the Ns and percentages together. You can also call it on a non-tabyl data.frame with tabyl-like format to which you wish to append Ns.

Usage

```
adorn_ns(dat, position = "rear", ns = attr(dat, "core"))
```

Arguments

`dat`	a data.frame of class `tabyl` that has had `adorn_percentages` and/or `adorn_pct_formatting` called on it. If given a list of data.frames, this function will apply itself to each data.frame in the list (designed for 3-way `tabyl` lists).
`position`	should the N go in the front, or in the rear, of the percentage?
`ns`	the Ns to append. The default is the "core" attribute of the input tabyl `dat`, where the original Ns of a two-way `tabyl` are stored. However, if you need to modify the numbers, e.g., to format `4000` as `4,000` or `4k`, you can do that separately and supply the formatted result here.

Value

a data.frame with Ns appended

Examples

```
mtcars %>%
  tabyl(am, cyl) %>%
  adorn_percentages("col") %>%
  adorn_pct_formatting() %>%
  adorn_ns(position = "front")
```

adorn_pct_formatting *Format a data.frame of decimals as percentages.*

Description

Numeric columns get multiplied by 100 and formatted as percentages according to user specifications. This function excludes the first column of the input data.frame, assuming that it contains a descriptive variable. Other non-numeric columns are also excluded.

Usage

```
adorn_pct_formatting(dat, digits = 1, rounding = "half to even",
  affix_sign = TRUE)
```

Arguments

dat	a data.frame with decimal values, typically the result of a call to adorn_percentages on a tabyl. If given a list of data.frames, this function will apply itself to each data.frame in the list (designed for 3-way tabyl lists).
digits	how many digits should be displayed after the decimal point?
rounding	method to use for rounding - either "half to even", the base R default method, or "half up", where 14.5 rounds up to 15.
affix_sign	should the % sign be affixed to the end?

Value

a data.frame with formatted percentages

Examples

```
mtcars %>%
  tabyl(am, cyl) %>%
  adorn_percentages("col") %>%
  adorn_pct_formatting()
```

adorn_percentages *Convert a data.frame of counts to percentages.*

Description

This function excludes the first column of the input data.frame, assuming that it contains a descriptive variable. If the input data.frame is not a tabyl, it will convert to one in order to preserve the underlying values in the core attribute.

Usage

```
adorn_percentages(dat, denominator = "row", na.rm = TRUE)
```

Arguments

dat	a `tabyl` or other data.frame with a tabyl-like layout. If given a list of data.frames, this function will apply itself to each data.frame in the list (designed for 3-way `tabyl` lists).
denominator	the direction to use for calculating percentages. One of "row", "col", or "all".
na.rm	should missing values (including NaN) be omitted from the calculations?

Value

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

Examples

```
mtcars %>%
  tabyl(am, cyl) %>%
  adorn_percentages("col")

# calculates correctly even with totals column and/or row:
mtcars %>%
  tabyl(am, cyl) %>%
  adorn_totals("row") %>%
  adorn_percentages()
```

---

adorn_rounding *Round the numeric columns in a data.frame.*

Description

Can run on any data.frame with at least one numeric column. This function defaults to excluding the first column of the input data.frame, assuming that it contains a descriptive variable, but this can be overridden with the argument `skip_first_col = FALSE`.

If you're formatting percentages, e.g., the result of `adorn_percentages()`, use `adorn_pct_formatting()` instead. This is a more flexible variant for ad-hoc usage. Compared to `adorn_pct_formatting()`, it can run on the first column and does not multiply by 100 or pad the numbers with spaces for alignment in the results data.frame. This function retains the class of numeric input columns.

Usage

```
adorn_rounding(dat, digits = 1, rounding = "half to even",
  skip_first_col = TRUE)
```

Arguments

dat	a data.frame with at least one numeric column
digits	how many digits should be displayed after the decimal point?
rounding	method to use for rounding - either "half to even", the base R default method, or "half up", where 14.5 rounds up to 15.
skip_first_col	should the first column be left unrounded, assuming it contains values of a descriptive variable as in a tabyl? Defaults to TRUE.

Value

Returns the data.frame with rounded numeric columns.

Examples

```
mtcars %>%
  tabyl(am, cyl) %>%
  adorn_percentages() %>%
  adorn_rounding(digits = 2, rounding = "half up")

# tolerates non-numeric columns:
library(dplyr)
mtcars %>%
  tabyl(am, cyl) %>%
  adorn_percentages("all") %>%
  mutate(dummy = "a") %>%
  adorn_rounding()
```

---

adorn_title *Add column name to the top of a two-way tabyl.*

Description

This function adds the column variable name to the top of a tabyl for a complete display of information. This makes the tabyl prettier, but renders the data.frame less useful for further manipulation.

Usage

```
adorn_title(dat, placement = "top", row_name, col_name)
```

Arguments

dat	a data.frame of class tabyl or other data.frame with a tabyl-like layout. If given a list of data.frames, this function will apply itself to each data.frame in the list (designed for 3-way tabyl lists).

placement	whether the column name should be added to the top of the tabyl in an otherwise-empty row "top" or appended to the already-present row name variable ("combined"). The formatting in the "top" option has the look of base R's table(); it also wipes out the other column names, making it hard to further use the data.frame besides formatting it for reporting. The "combined" option is more conservative in this regard.
row_name	(optional) default behavior is to pull the row name from the attributes of the input tabyl object. If you wish to override that text, or if your input is not a tabyl, supply a string here.
col_name	(optional) default behavior is to pull the column_name from the attributes of the input tabyl object. If you wish to override that text, or if your input is not a tabyl, supply a string here.

Value

the input tabyl, augmented with the column title. Non-tabyl inputs that are of class tbl_df are downgraded to basic data.frames so that the title row prints correctly.

Examples

```
mtcars %>%
  tabyl(am, cyl) %>%
  adorn_title(placement = "top")

# Adding a title to a non-tabyl
library(tidyr); library(dplyr)
mtcars %>%
  group_by(gear, am) %>%
  summarise(avg_mpg = mean(mpg)) %>%
  spread(gear, avg_mpg) %>%
  adorn_title("top", row_name = "Gears", col_name = "Cylinders")
```

---

adorn_totals *Append a totals row and/or column to a data.frame.*

Description

This function excludes the first column of the input data.frame, assuming it's a descriptive variable not to be summed. Non-numeric columns are converted to character class and have a user-specified fill character inserted in the totals row.

Usage

```
adorn_totals(dat, where = "row", fill = "-", na.rm = TRUE)
```

Arguments

dat	an input data.frame with at least one numeric column. If given a list of data.frames, this function will apply itself to each data.frame in the list (designed for 3-way tabyl lists).
where	one of "row", "col", or c("row", "col")
fill	if there are multiple non-numeric columns, what string should fill the bottom row of those columns?
na.rm	should missing values (including NaN) be omitted from the calculations?

Value

Returns a data.frame augmented with a totals row, column, or both. The data.frame is now also of class tabyl and stores information about the attached totals and underlying data in the tabyl attributes.

Examples

```
mtcars %>%
  tabyl(am, cyl) %>%
  adorn_totals()
```

---

as_tabyl *Add* tabyl *attributes to a data.frame.*

Description

A tabyl is a data.frame containing counts of a variable or co-occurrences of two variables (a.k.a., a contingency table or crosstab). This specialized kind of data.frame has attributes that enable adorn_ functions to be called for precise formatting and presentation of results. E.g., display results as a mix of percentages, Ns, add totals rows or columns, rounding options, in the style of Microsoft Excel PivotTable.

A tabyl can be the result of a call to janitor::tabyl(), in which case these attributes are added automatically. This function adds tabyl class attributes to a data.frame that isn't the result of a call to tabyl but meets the requirements of a two-way tabyl: 1) First column contains values of variable 1 2) Column names 2:n are the values of variable 2 3) Numeric values in columns 2:n are counts of the co-occurrences of the two variables.*

* = this is the ideal form of a tabyl, but janitor's adorn_ functions tolerate and ignore non-numeric columns in positions 2:n.

For instance, the result of dplyr::count() followed by tidyr::spread() can be treated as a tabyl.

The result of calling tabyl() on a single variable is a special class of one-way tabyl; this function only pertains to the two-way tabyl.

Usage

```
as_tabyl(dat, axes = 2, row_var_name = NULL, col_var_name = NULL)
```

Arguments

dat	a data.frame with variable values in the first column and numeric values in all other columns.
axes	is this a two_way tabyl or a one_way tabyl? If this function is being called by a user, this should probably be "2". One-way tabyls are created by `tabyl` but are a special case.
row_var_name	(optional) the name of the variable in the row dimension; used by `adorn_title()`.
col_var_name	(optional) the name of the variable in the column dimension; used by `adorn_title()`.

Value

Returns the same data.frame, but with the additional class of "tabyl" and the attribute "core".

Examples

```
as_tabyl(mtcars)
```

---

clean_names *Cleans names of a data.frame.*

Description

Resulting names are unique and consist only of the _ character, numbers, and letters. Capitalization preferences can be specified using the `case` parameter.

Accented characters are transliterated to ASCII. For example, an "o" with a German umlaut over it becomes "o", and the Spanish character "enye" becomes "n".

Usage

```
clean_names(dat, case = c("snake", "lower_camel", "upper_camel",
  "screaming_snake", "lower_upper", "upper_lower", "all_caps",
  "small_camel", "big_camel", "old_janitor", "parsed", "mixed"))
```

Arguments

dat	the input data.frame.
case	The desired target case (default is `"snake"`), indicated by these possible values: • `"snake"` produces snake_case • `"lower_camel"` or `"small_camel"` produces lowerCamel • `"upper_camel"` or `"big_camel"` produces UpperCamel • `"screaming_snake"` or `"all_caps"` produces ALL_CAPS • `"lower_upper"` produces lowerUPPER • `"upper_lower"` produces UPPERlower

- `old_janitor`: legacy compatibility option to preserve behavior of `clean_names` prior to addition of the "case" argument(janitor versions <= 0.3.1). Provided as a quick fix for old scripts broken by the changes to `clean_names` in janitor v1.0.
- `"parsed"`, `"mixed"`, `"none"`, `"internal_parsing"`: less-common cases offered by `snakecase::to_any_case`. See `to_any_case` for details.

Value

Returns the data.frame with clean names.

Examples

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

# or pipe in the input data.frame:
# poorly_named_df %>% clean_names()

# if you prefer camelCase variable names:
# poorly_named_df %>% clean_names(., "small_camel")

# not run:
# library(readxl)
# 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.

Warning

Deprecated, do not use in new code. Use `dplyr::na_if()` instead.

See Also

janitor_deprecated

`crosstab` *Generate a crosstabulation of two vectors.*

Description

This function is deprecated, use `tabyl(dat, var1, var2)` instead.

Usage

```
crosstab(...)

## Default S3 method:
crosstab(vec1, vec2, percent = "none",
  show_na = TRUE, ...)

## S3 method for class 'data.frame'
crosstab(.data, ...)
```

Arguments

`...`	additional arguments, if calling `crosstab` on a data.frame.
`vec1`	the vector to place on the crosstab column. If supplying a data.frame, this should be an unquoted column name.
`vec2`	the vector to place on the crosstab row. If supplying a data.frame, this should be an unquoted column name.
`percent`	which grouping to use for percentages, if desired (defaults to "none", which returns simple counts). Must be one of "none", "row", "col", or "all".
`show_na`	a logical value indicating whether counts should be displayed where either variable is `NA`.
`.data`	(optional) a data.frame, in which case `vec1` and `vec2` should be unquoted column names.

Value

Returns a data.frame with the frequencies of the crosstabulated variables.

`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",
  include_time = FALSE, round_seconds = TRUE)
```

Arguments

`date_num`	numeric vector of serial numbers to convert.
`date_system`	the date system, either `"modern"` or `"mac pre-2011"`.
`include_time`	Include the time (hours, minutes, seconds) in the output? (See details)
`round_seconds`	Round the seconds to an integer (only has an effect when `include_time` is `TRUE`)?

Details

When using `include_time=TRUE` the returned object will not include a time zone since excel does not store dates and times with time zones. When adding time zones, ensure that any conversion done does not change the intended time including if the time is or is not in daylight savings time. Also, days with leap seconds will not be accurately handled as they do not appear to be accurately handled by Windows (as described in https://support.microsoft.com/en-us/help/2722715/support-for-the-leap-second).

Value

Returns a vector of class Date if `include_time` is `FALSE`. Returns a vector of class POSIXlt if `include_time` is `TRUE`.

Examples

```
excel_numeric_to_date(40000)
excel_numeric_to_date(40000.5) # No time is included
excel_numeric_to_date(40000.5, include_time = TRUE) # Time is included
excel_numeric_to_date(40000.521, include_time = TRUE) # Time is included
excel_numeric_to_date(40000.521, include_time = TRUE,
  round_seconds = FALSE) # Time with fractional seconds is included
```

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 the magrittr pipe %>% :
mtcars %>% get_dupes(wt)
```

janitor_deprecated — *Deprecated Functions in Package janitor*

Description

These functions are provided for compatibility with older versions of `janitor` only, and may be defunct as soon as the next release.

Details

- `use_first_valid_of`
- `convert_to_NA`

remove_empty *Remove empty rows and/or columns from a data.frame.*

Description

Removes all rows and/or columns from a data.frame that are composed entirely of NA values.

Usage

```
remove_empty(dat, which = c("rows", "cols"))
```

Arguments

dat	the input data.frame.
which	one of "rows", "cols", or c("rows", "cols"). Where no value of which is provided, defaults to removing both empty rows and empty columns, declaring the behavior with a printed message.

Value

Returns the data.frame without its missing rows or columns.

Examples

```
# not run:
# dat %>% remove_empty("rows")
```

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

Description

This function is deprecated, use remove_empty("cols") instead.

Usage

```
remove_empty_cols(dat)
```

Arguments

dat	the input data.frame.

Value

Returns the data.frame with no empty columns.

Examples

```
# not run:
# dat %>% remove_empty_cols
```

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

Description

This function is deprecated, use `remove_empty("rows")` instead.

Usage

```
remove_empty_rows(dat)
```

Arguments

`dat`	the input data.frame.

Value

Returns the data.frame with no empty rows.

Examples

```
# not run:
# dat %>% remove_empty_rows
```

`round_half_up` *Round a numeric vector; halves will be rounded up, ala Microsoft Excel.*

Description

In base R `round()`, halves are rounded to even, e.g., 12.5 and 11.5 are both rounded to 12. This function rounds 12.5 to 13 (assuming `digits = 0`). Negative halves are rounded away from zero, e.g., -0.5 is rounded to -1.

This may skew subsequent statistical analysis of the data, but may be desirable in certain contexts. This function is implemented exactly from http://stackoverflow.com/a/12688836; see that question and comments for discussion of this issue.

Usage

```
round_half_up(x, digits = 0)
```

Arguments

x	a numeric vector to round.
digits	how many digits should be displayed after the decimal point?

Examples

```
round_half_up(12.5)
round_half_up(1.125, 2)
round_half_up(1.125, 1)
round_half_up(-0.5, 0) # negatives get rounded away from zero
```

---

row_to_names *Elevate a row to be the column names of a data.frame.*

Description

Elevate a row to be the column names of a data.frame.

Usage

```
row_to_names(dat, row_number, remove_row = TRUE,
  remove_rows_above = TRUE)
```

Arguments

dat	The input data.frame
row_number	The row of `dat` containing the variable names
remove_row	Should the row `row_number` be removed from the resulting data.frame?
remove_rows_above	If `row_number != 1`, should the rows above `row_number` - that is, between `1:(row_number-1)` - be removed from the resulting data.frame?

Value

A data.frame with new names (and some rows removed, if specified)

Examples

```
x <- data.frame(X_1 = c(NA, "Title", 1:3),
           X_2 = c(NA, "Title2", 4:6))
x %>%
  row_to_names(row_number = 2)
```

tabyl *Generate a frequency table (1-, 2-, or 3-way).*

Description

A fully-featured alternative to `table()`. Results are data.frames and can be formatted and enhanced with janitor's family of `adorn_` functions.

Specify a data.frame and the one, two, or three unquoted column names you want to tabulate. Three variables generates a list of 2-way tabyls, split by the third variable.

Alternatively, you can tabulate a single variable that isn't in a data.frame by calling `tabyl` on a vector, e.g., `tabyl(mtcars$gear)`.

Usage

```
tabyl(dat, ...)

## Default S3 method:
tabyl(dat, show_na = TRUE,
  show_missing_levels = TRUE, ...)

## S3 method for class 'data.frame'
tabyl(dat, var1, var2, var3, show_na = TRUE,
  show_missing_levels = TRUE, ...)
```

Arguments

`dat`	a data.frame containing the variables you wish to count. Or, a vector you want to tabulate.
`...`	the arguments to tabyl (here just for the sake of documentation compliance, as all arguments are listed with the vector- and data.frame-specific methods)
`show_na`	should counts of `NA` values be displayed? In a one-way tabyl, the presence of `NA` values triggers an additional column showing valid percentages(calculated excluding `NA` values).
`show_missing_levels`	should counts of missing levels of factors be displayed? These will be rows and/or columns of zeroes. Useful for keeping consistent output dimensions even when certain factor levels may not be present in the data.
`var1`	the column name of the first variable.
`var2`	(optional) the column name of the second variable (the rows in a 2-way tabulation).
`var3`	(optional) the column name of the third variable (the list in a 3-way tabulation).

Value

Returns a data.frame with frequencies and percentages of the tabulated variable(s). A 3-way tabulation returns a list of data.frames.

Examples

```
tabyl(mtcars, cyl)
tabyl(mtcars, cyl, gear)
tabyl(mtcars, cyl, gear, am)

# or using the %>% pipe
mtcars %>%
  tabyl(cyl, gear)

# illustrating show_na functionality:
my_cars <- rbind(mtcars, rep(NA, 11))
my_cars %>% tabyl(cyl)
my_cars %>% tabyl(cyl, show_na = FALSE)

# Calling on a single vector not in a data.frame:
val <- c("hi", "med", "med", "lo")
tabyl(val)
```

---

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)
```

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?

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)
```

untabyl *Remove* tabyl *attributes from a data.frame.*

Description

Strips away all tabyl-related attributes from a data.frame.

Usage

```
untabyl(dat)
```

Arguments

dat	a data.frame of class tabyl.

Value

Returns the same data.frame, but without the tabyl class and attributes.

Examples

```
mtcars %>%
  tabyl(am) %>%
  untabyl() %>%
  attributes() # tabyl-specific attributes are gone
```

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.

Warning

Deprecated, do not use in new code. Use `dplyr::coalesce()` instead.

See Also

janitor_deprecated

Index