Package ‘naniar’

September 2, 2020

Type Package

Title Data Structures, Summaries, and Visualisations for Missing Data

Version 0.6.0

Description Missing values are ubiquitous in data and need to be explored and handled in the initial stages of analysis. ‘naniar’ provides data structures and functions that facilitate the plotting of missing values and examination of imputations. This allows missing data dependencies to be explored with minimal deviation from the common work patterns of 'ggplot2' and tidy data. The work is fully discussed at Tierney & Cook (2018) <arXiv:1809.02264>.

License MIT + file LICENSE

LazyData TRUE

ByteCompile TRUE

Suggests knitr, rmarkdown, testthat (>= 2.1.0), rpart, rpart.plot, covr, gridExtra, wakefield, vdiffr, here, simputation, imputeTS, gdtools, Hmisc, spelling

VignetteBuilder knitr

Depends R (>= 3.1.2)

Imports dplyr, ggplot2, purrr, tidyr, tibble (>= 2.0.0), magrittr, stats, visdat, forcats, viridis, glue, UpSetR

<table>
<thead>
<tr>
<th>R topics documented:</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>GeomMissPoint</td>
<td>22</td>
</tr>
<tr>
<td>geom_miss_point</td>
<td>22</td>
</tr>
<tr>
<td>gg_miss_case</td>
<td>24</td>
</tr>
<tr>
<td>gg_miss_case_cumsum</td>
<td>25</td>
</tr>
<tr>
<td>gg_miss_fct</td>
<td>26</td>
</tr>
<tr>
<td>gg_miss_span</td>
<td>27</td>
</tr>
<tr>
<td>gg_miss_upset</td>
<td>28</td>
</tr>
<tr>
<td>gg_miss_var</td>
<td>29</td>
</tr>
<tr>
<td>gg_miss_var_cumsum</td>
<td>30</td>
</tr>
<tr>
<td>gg_miss_which</td>
<td>30</td>
</tr>
<tr>
<td>group_by_fun</td>
<td>31</td>
</tr>
<tr>
<td>impute_below</td>
<td>32</td>
</tr>
<tr>
<td>impute_below_all</td>
<td>32</td>
</tr>
<tr>
<td>impute_below_at</td>
<td>33</td>
</tr>
<tr>
<td>impute_below_if</td>
<td>34</td>
</tr>
<tr>
<td>impute_mean</td>
<td>35</td>
</tr>
<tr>
<td>impute_median</td>
<td>36</td>
</tr>
<tr>
<td>is_shade</td>
<td>37</td>
</tr>
<tr>
<td>label_missings</td>
<td>38</td>
</tr>
<tr>
<td>label_miss_1d</td>
<td>39</td>
</tr>
<tr>
<td>label_miss_2d</td>
<td>39</td>
</tr>
<tr>
<td>label_shadow</td>
<td>40</td>
</tr>
<tr>
<td>miss-pct-prop-defunct</td>
<td>41</td>
</tr>
<tr>
<td>miss_case_cumsum</td>
<td>41</td>
</tr>
<tr>
<td>miss_case_summary</td>
<td>42</td>
</tr>
<tr>
<td>miss_case_table</td>
<td>43</td>
</tr>
<tr>
<td>miss_prop_summary</td>
<td>44</td>
</tr>
<tr>
<td>miss_scan_count</td>
<td>45</td>
</tr>
<tr>
<td>miss_summary</td>
<td>46</td>
</tr>
<tr>
<td>miss_var_cumsum</td>
<td>47</td>
</tr>
<tr>
<td>miss_var_run</td>
<td>48</td>
</tr>
<tr>
<td>miss_var_span</td>
<td>49</td>
</tr>
<tr>
<td>miss_var_summary</td>
<td>50</td>
</tr>
<tr>
<td>miss_var_table</td>
<td>51</td>
</tr>
<tr>
<td>miss_var_which</td>
<td>52</td>
</tr>
<tr>
<td>n-var-case-complete</td>
<td>53</td>
</tr>
<tr>
<td>n-var-case-miss</td>
<td>53</td>
</tr>
<tr>
<td>nabular</td>
<td>54</td>
</tr>
<tr>
<td>naniar</td>
<td>55</td>
</tr>
<tr>
<td>new_shade</td>
<td>55</td>
</tr>
<tr>
<td>n_complete</td>
<td>56</td>
</tr>
<tr>
<td>n_complete_row</td>
<td>57</td>
</tr>
<tr>
<td>n_miss</td>
<td>57</td>
</tr>
<tr>
<td>n_miss_row</td>
<td>58</td>
</tr>
<tr>
<td>oceanbuoys</td>
<td>59</td>
</tr>
<tr>
<td>pct-miss-complete-case</td>
<td>60</td>
</tr>
<tr>
<td>pct-miss-complete-var</td>
<td>61</td>
</tr>
<tr>
<td>pct_complete</td>
<td>62</td>
</tr>
</tbody>
</table>
add any miss

Description

This adds a column named "any_miss" (by default) that describes whether there are any missings in all of the variables (default), or whether any of the specified columns, specified using variables names or dplyr verbs, starts_with, contains, ends_with, etc. By default the added column will be called "any_miss_all", if no variables are specified, otherwise, if variables are specified, the label will be "any_miss_vars" to indicate that not all variables have been used to create the labels.
add_any_miss

Usage

add_any_miss(
    data,
    ...,  
    label = "any_miss",  
    missing = "missing",  
    complete = "complete"
)

Arguments

data: data.frame

...: Variable names to use instead of the whole dataset. By default this looks at the whole dataset. Otherwise, this is one or more unquoted expressions separated by commas. These also respect the dplyr verbs starts_with, contains, ends_with, etc. By default will add "_all" to the label if left blank, otherwise will add "_vars" to distinguish that it has not been used on all of the variables.

label: label for the column, defaults to "any_miss". By default if no additional variables are listed the label col is "any_miss_all", otherwise it is "any_miss_vars", if variables are specified.

missing: character a label for when values are missing - defaults to "missing"

complete: character character a label for when values are complete - defaults to "complete"

Details

By default the prefix "any_miss" is used, but this can be changed in the label argument.

Value

data.frame with data and the column labelling whether that row (for those variables) has any missing values - indicated by "missing" and "complete".

See Also

bind_shadow() add_any_miss() add_label_missings() add_label_shadow() add_miss_cluster()
add_n_miss() add_prop_miss() add_shadow_shift() cast_shadow()

Examples

airquality %>% add_any_miss()
airquality %>% add_any_miss(Ozone, Solar.R)
add_label_missings

Add a column describing if there are any missings in the dataset

Description
Add a column describing if there are any missings in the dataset

Usage
add_label_missings(data, ..., missing = "Missing", complete = "Not Missing")

Arguments
- data: data.frame
- ...: extra variable to label
- missing: character a label for when values are missing - defaults to "Missing"
- complete: character a label for when values are complete - defaults to "Not Missing"

Value
data.frame with a column "any_missing" that is either "Not Missing" or "Missing" for the purposes of plotting / exploration / nice print methods

See Also
bind_shadow() add_any_miss() add_label_missings() add_label_shadow() add_miss_cluster()
add_n_miss() add_prop_miss() add_shadow_shift() cast_shadow()

Examples
airquality %>% add_label_missings()
airquality %>% add_label_missings(Ozone, Solar.R)
airquality %>% add_label_missings(Ozone, Solar.R, missing = "yes", complete = "no")
add_label_shadow

Description

Instead of focusing on labelling whether there are missings, we instead focus on whether there have been any shadows created. This can be useful when data has been imputed and you need to determine which rows contained missing values when the shadow was bound to the dataset.

Usage

add_label_shadow(data, ..., missing = "Missing", complete = "Not Missing")

Arguments

data	data.frame
...	extra variable to label
missing	character a label for when values are missing - defaults to "Missing"
complete	character a label for when values are complete - defaults to "Not Missing"

Value

data.frame with a column, "any_missing", which describes whether or not there are any rows that have a shadow value.

See Also

bind_shadow() add_any_miss() add_label_missings() add_label_shadow() add_miss_cluster()
add_n_miss() add_prop_miss() add_shadow_shift() cast_shadow()

Examples

```r
airquality %>%
  add_shadow(Ozone, Solar.R) %>%
  add_label_shadow()
```
add_miss_cluster

Add a column that tells us which "missingness cluster" a row belongs to

Description

A way to extract the cluster of missingness that a group belongs to. For example, if you use \texttt{vis_miss(airquality, cluster = TRUE)}, you can see some clustering in the data, but you do not have a way to identify the cluster. Future work will incorporate the seriation package to allow for better control over the clustering from the user.

Usage

\texttt{add_miss_cluster(data, cluster_method = "mcquitty", n_clusters = 2)}

Arguments

- \texttt{data}: a dataframe
- \texttt{cluster_method}: character vector of the agglomeration method to use, the default is "mcquitty". Options are taken from \texttt{stats::hclust} helpfile, and options include: "ward.D", "ward.D2", "single", "complete", "average" (= UPGMA), "mcquitty" (= WPGMA), "median" (= WPGMC) or "centroid" (= UPGMC).
- \texttt{n_clusters}: numeric the number of clusters you expect. Defaults to 2.

See Also

\texttt{bind_shadow()} \texttt{add_any_miss()} \texttt{add_label_missings()} \texttt{add_label_shadow()} \texttt{add_miss_cluster()} \texttt{add_n_miss()} \texttt{add_prop_miss()} \texttt{add_shadow_shift()} \texttt{cast_shadow()}

Examples

\begin{verbatim}
add_miss_cluster(airquality)
add_miss_cluster(airquality, n_clusters = 3)
add_miss_cluster(airquality, cluster_method = "ward.D", n_clusters = 3)
\end{verbatim}

add_n_miss

Add column containing number of missing data values

Description

It can be useful when doing data analysis to add the number of missing data points into your dataframe. \texttt{add_n_miss} adds a column named "n_miss", which contains the number of missing values in that row.
add_prop_miss

Usage

add_n_miss(data, ..., label = "n_miss")

Arguments

data a dataframe

... Variable names to use instead of the whole dataset. By default this looks at the whole dataset. Otherwise, this is one or more unquoted expressions separated by commas. These also respect the dplyr verbs starts_with, contains, ends_with, etc. By default will add "_all" to the label if left blank, otherwise will add "_vars" to distinguish that it has not been used on all of the variables.

label character default is "n_miss".

Value

a dataframe

See Also

bind_shadow() add_any_miss() add_label_missings() add_label_shadow() add_miss_cluster() add_prop_miss() add_shadow_shift() cast_shadow()

Examples

airquality %>% add_n_miss()
airquality %>% add_n_miss(Ozone, Solar.R)
airquality %>% add_n_miss(dplyr::contains("o"))

add_prop_miss

Add column containing proportion of missing data values

Description

It can be useful when doing data analysis to add the proportion of missing data values into your dataframe. add_prop_miss adds a column named "prop_miss", which contains the proportion of missing values in that row. You can specify the variables that you would like to show the missingness for.

Usage

add_prop_miss(data, ..., label = "prop_miss")
Arguments

data a dataframe

... Variable names to use instead of the whole dataset. By default this looks at the whole dataset. Otherwise, this is one or more unquoted expressions separated by commas. These also respect the dplyr verbs `starts_with`, `contains`, `ends_with`, etc. By default will add "_all" to the label if left blank, otherwise will add "_vars" to distinguish that it has not been used on all of the variables.

label character string of what you need to name variable

Value

a dataframe

See Also

`bind_shadow()` `add_any_miss()` `add_label_missings()` `add_label_shadow()` `add_miss_cluster()`
`add_prop_miss()` `add_shadow_shift()` `cast_shadow()`

Examples

```r
airquality %>% add_prop_miss()
airquality %>% add_prop_miss(Solar.R, Ozone)
airquality %>% add_prop_miss(Solar.R, Ozone, label = "testing")
```

# this can be applied to model the proportion of missing data
# as in Tierney et al (doi: 10.1136/bmjopen-2014-007450)
# see "Modelling missingness" in vignette "Getting Started with naniar"
# for details

---

**add_shadow**

*Add a shadow column to dataframe*

Description

As an alternative to `bind_shadow()`, you can add specific individual shadow columns to a dataset. These also respect the dplyr verbs `starts_with`, `contains`, `ends_with`, etc.

Usage

`add_shadow(data, ...)`

Arguments

data data.frame

... One or more unquoted variable names, separated by commas. These also respect the dplyr verbs `starts_with`, `contains`, `ends_with`, etc.
add_shadow_shift

Value
data.frame

See Also
bind_shadow() add_any_miss() add_label_missings() add_label_shadow() add_miss_cluster()
add_n_miss() add_prop_miss() add_shadow_shift() cast_shadow()

Examples

airquality %>% add_shadow(Ozone)
airquality %>% add_shadow(Ozone, Solar.R)

Description
Shadow shift missing values using only the selected variables in a dataset, by specifying variable
names or use dplyr vars and dplyr verbs starts_with, contains, ends_with, etc.

Usage
add_shadow_shift(data, ..., suffix = "shift")

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>data</td>
<td>data.frame</td>
</tr>
<tr>
<td>...</td>
<td>One or more unquoted variable names separated by commas. These also respect the dplyr verbs starts_with, contains, ends_with, etc.</td>
</tr>
<tr>
<td>suffix</td>
<td>suffix to add to variable, defaults to &quot;shift&quot;</td>
</tr>
</tbody>
</table>

Value
data with the added variable shifted named as var_suffix

See Also
bind_shadow() add_any_miss() add_label_missings() add_label_shadow() add_miss_cluster()
add_n_miss() add_prop_miss() add_shadow_shift() cast_shadow()

Examples

airquality %>% add_shadow_shift(Ozone, Solar.R)
add_span_counter  
Add a counter variable for a span of dataframe

Description

Adds a variable, span_counter to a dataframe. Used internally to facilitate counting of missing values over a given span.

Usage

add_span_counter(data, span_size)

Arguments

data  
data.frame
span_size  
integer

Value

data.frame with extra variable "span_counter".

Examples

## Not run:
# add_span_counter(pedestrian, span_size = 100)
## End(Not run)

all-is-miss-complete  
Identify if all values are missing or complete

Description

This is shorthand for all(is.na(x)) and all(!is.na(x))

Usage

all_na(x)
all_miss(x)
all_complete(x)

Arguments

x  
an R object to be tested.
any-na

Examples

misses <- c(NA, NA, NA)
complete <- c(1, 2, 3)
mixture <- c(NA, 1, NA)

all_na(misses)
all_na(complete)
all_na(mixture)
all_complete(misses)
all_complete(complete)
all_complete(mixture)

any-na Identify if there are any missing or complete values

Description

It is useful to search for any instances of missing or complete values. There are two functions
that do this in naniar - any_miss and its alias any_na. These bother under the hood call anyNA.

any_complete is the complement to any_miss - it returns TRUE if there are any complete values.

Usage

any_na(x)

any_miss(x)

any_complete(x)

Arguments

x an R object to be tested

See Also

all_miss() all_complete

Examples

anyNA(airquality)
any_na(airquality)
any_miss(airquality)
any_complete(airquality)
any_row_miss  
*Helper function to determine whether there are any missings*

**Description**

Helper function to determine whether there are any missings

**Usage**

`any_row_miss(x)`

**Arguments**

- `x`  
a vector

**Value**

logical vector TRUE = missing FALSE = complete

---

as_shadow  
*Create shadows*

**Description**

Return a tibble in shadow matrix form, where the variables are the same but have a suffix _NA attached to distinguish them.

**Usage**

`as_shadow(data, ...)`

**Arguments**

- `data`  
dataframe
- `...`  
selected variables to use

**Details**

Representing missing data structure is achieved using the shadow matrix, introduced in Swayne and Buja. The shadow matrix is the same dimension as the data, and consists of binary indicators of missingness of data values, where missing is represented as "NA", and not missing is represented as "!NA". Although these may be represented as 1 and 0, respectively.

**Value**

appended shadow with column names
as_shadow_upset

Examples

as_shadow(airquality)

---

as_shadow_upset  
*Convert data into shadow format for doing an upset plot*

Description

Upset plots are a way of visualising common sets, this function transforms the data into a format that feeds directly into an upset plot.

Usage

`as_shadow_upset(data)`

Arguments

- `data` a data.frame

Value

a data.frame

Examples

```r
## Not run:
library(UpSetR)
airquality %>%
  as_shadow_upset() %>%
  upset()

## End(Not run)
```
**bind_shadow**

**Bind a shadow dataframe to original data**

**Description**

Binding a shadow matrix to a regular dataframe helps visualise and work with missing data.

**Usage**

```r
bind_shadow(data, only_miss = FALSE, ...)
```

**Arguments**

- `data` a dataframe
- `only_miss` logical - if FALSE (default) it will bind a dataframe with all of the variables duplicated with their shadow. Setting this to TRUE will bind variables only those variables that contain missing values. See the examples for more details.
- `...` extra options to pass to `recode_shadow()` - a work in progress.

**Value**

data with the added variable shifted and the suffix _NA

**Examples**

```r
bind_shadow(airquality)
# bind only the variables that contain missing values
bind_shadow(airquality, only_miss = TRUE)

aq_shadow <- bind_shadow(airquality)

## Not run:
# explore missing data visually
library(ggplot2)

# using the bounded shadow to visualise Ozone according to whether Solar # Radiation is missing or not.

ggplot(data = aq_shadow,
       aes(x = Ozone)) +
  geom_histogram() +
  facet_wrap(~Solar.R_NA, ncol = 1)

## End(Not run)
```
Add a shadow column to a dataset

Description

Casting a shadow shifted column performs the equivalent pattern to data `%>%` select(var) `%>%` shadow_shift(). This is a convenience function that makes it easy to perform certain visualisations, in line with the principle that the user should have a way to flexibly return data formats containing information about the missing data. It forms the base building block for the functions cast_shadow_shift, and cast_shadow_shift_label. It also respects the dplyr verbs starts_with, contains, ends_with, etc. to select variables.

Usage

cast_shadow(data, ...)

Arguments

data data.frame

... One or more unquoted variable names separated by commas. These respect the dplyr verbs starts_with, contains, ends_with, etc.

Value

data with the added variable shifted and the suffix _NA

See Also

cast_shadow_shift(), cast_shadow_shift_label(), bind_shadow(), add_any_miss(), add_label_missings(), add_label_shadow(), add_miss_cluster(), add_prop_miss(), add_shadow_shift()

Examples

```r
airquality %>% cast_shadow(Ozone, Solar.R)
## Not run:
library(ggplot2)
library(magrittr)
airquality %>%
cast_shadow(Ozone, Solar.R) %>%
ggplot(aes(x = Ozone,
        colour = Solar.R\_NA)) +
geom_density()
## End(Not run)
```
cast_shadow_shift

Add a shadow and a shadow_shift column to a dataset

Description
Shift the values and add a shadow column. It also respects the dplyr verbs starts_with, contains, ends_with, etc.

Usage
```
cast_shadow_shift(data, ...)
```

Arguments
- **data** data.frame
- **...** One or more unquoted variable names separated by commas. These respect the dplyr verbs starts_with, contains, ends_with, etc.

Value
data.frame with the shadow and shadow_shift vars

See Also
cast_shadow_shift(), cast_shadow_shift_label(), bind_shadow(), add_any_miss(), add_label_missings(), add_label_shadow(), add_miss_cluster(), add_prop_miss(), add_shadow_shift()

Examples
```
airquality %>% cast_shadow_shift(Ozone, Temp)
airquality %>% cast_shadow_shift(dplyr::contains("o"))
```

cast_shadow_shift_label

Add a shadow column and a shadow shifted column to a dataset

Description
Shift the values, add shadow, add missing label

Usage
```
cast_shadow_shift_label(data, ...)
```
Arguments

\begin{itemize}
\item \texttt{data} \hspace{1cm} \texttt{data.frame}
\item \ldots \hspace{1cm} One or more unquoted expressions separated by commas. These also respect the dplyr verbs "starts_with", "contains", "ends_with", etc.
\end{itemize}

Value

\texttt{data.frame} with the shadow and shadow\_shift vars, and missing labels

See Also

\texttt{cast\_shadow\_shift()}, \texttt{cast\_shadow\_shift\_label()} \texttt{bind\_shadow()} \texttt{add\_any\_miss()} \texttt{add\_label\_missings()}
\texttt{add\_label\_shadow()} \texttt{add\_miss\_cluster()} \texttt{add\_prop\_miss()} \texttt{add\_shadow\_shift()}

Examples

\begin{verbatim}
airquality \%\% cast\_shadow\_shift\_label(Ozone, Solar.R)

# replicate the plot generated by geom\_miss\_point()
## Not run:
library(ggplot2)
airquality \%\%
cast\_shadow\_shift\_label(Ozone,Solar.R) \%\%
ggplot(aes(x = Ozone\_shift,
y = Solar.R\_shift,
    colour = any\_missing)) +
  geom\_point()

## End(Not run)
\end{verbatim}

---

common\_na\_numbers \hspace{1cm} Common number values for NA

Description

This vector contains common number values of NA (missing), which is aimed to be used inside na\-niar functions \texttt{miss\_scan\_count()} and \texttt{replace\_with\_na()}. The current list of numbers can be found by printing out \texttt{common\_na\_numbers}. It is a useful way to explore your data for possible miss-ings, but I strongly warn against using this to replace NA values without very carefully looking at the incidence for each of the cases. Common NA strings are in the data object \texttt{common\_na\_strings}.

Usage

\texttt{common\_na\_numbers}
Format

An object of class numeric of length 8.

Note

original discussion here https://github.com/njtierney/naniar/issues/168

Examples

dat_ms <- tibble::tribble(~x, ~y, ~z,
  1,  "A",  -100,
  3,  "N/A",  -99,
   NA,  NA,  -98,
  -99,  "E",  -101,
  -98,  "F",  -1)

miss_scan_count(dat_ms, -99)
miss_scan_count(dat_ms, c("-99", "-98", "N/A"))
common_na_numbers
miss_scan_count(dat_ms, common_na_numbers)

common_na_strings

Common string values for NA

Description

This vector contains common values of NA (missing), which is aimed to be used inside naniar functions miss_scan_count() and replace_with_na(). The current list of strings used can be found by printing out common_na_strings. It is a useful way to explore your data for possible missings, but I strongly warn against using this to replace NA values without very carefully looking at the incidence for each of the cases. Please note that common_na_strings uses \ around the "?", "." and "*" characters to protect against using their wildcard features in grep. Common NA numbers are in the data object common_na_numbers.

Usage

common_na_strings

Format

An object of class character of length 24.

Note

original discussion here https://github.com/njtierney/naniar/issues/168
Examples

dat_ms <- tibble::tribble(~x, ~y, ~z,
1,  "A", -100,
3,  "N/A", -99,
NA, NA, -98,
-99, "E", -101,
-98, "F", -1)

miss_scan_count(dat_ms, -99)
miss_scan_count(dat_ms, c("-99","-98","N/A"))
common_na_strings
miss_scan_count(dat_ms, common_na_strings)

gather_shadow

Long form representation of a shadow matrix

Description

gather_shadow is a long-form representation of binding the shadow matrix to your data, producing variables named case, variable, and missing, where missing contains the missing value representation.

Usage

gather_shadow(data)

Arguments

data a dataframe

Value

dataframe in long, format, containing information about the missings

Examples

gather_shadow(airquality)
Description

These are the stat and geom overrides using ggproto from ggplot2 that make naniar work.

Usage

StatMissPoint

Format

An object of class StatMissPoint (inherits from Stat, ggproto, gg) of length 6.

Description

geom_miss_point provides a way to transform and plot missing values in ggplot2. To do so it uses methods from ggobi to display missing data points on the same axis.

Usage

geom_miss_point(
  mapping = NULL,
  data = NULL,
  prop_below = 0.1,
  jitter = 0.05,
  stat = "miss_point",
  position = "identity",
  colour = ..missing..,
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE,
  ...
)

Arguments

mapping Set of aesthetic mappings created by ggplot2::aes() or ggplot2::aes_. If specified and inherit.aes = TRUE (the default), is combined with the default mapping at the top level of the plot. You only need to supply mapping if there isn’t a mapping defined for the plot.
geom_miss_point

data A data frame. If specified, overrides the default data frame defined at the top level of the plot.

prop_below the degree to shift the values. The default is 0.1

jitter the amount of jitter to add. The default is 0.05

stat The statistical transformation to use on the data for this layer, as a string.

position Position adjustment, either as a string, or the result of a call to a position adjustment function.

colour the colour chosen for the aesthetic

na.rm If FALSE (the default), removes missing values with a warning. If TRUE silently removes missing values.

show.legend logical. Should this layer be included in the legends? NA, the default, includes if any aesthetics are mapped. FALSE never includes, and TRUE always includes.

inherit.aes If FALSE, overrides the default aesthetics, rather than combining with them. This is most useful for helper functions that define both data and aesthetics and shouldn’t inherit behaviour from the default plot specification, e.g. borders.

... other arguments passed on to ggplot2::layer(). There are three types of arguments you can use here:

• Aesthetics: to set an aesthetic to a fixed value, like color = "red" or size = 3.
• Other arguments to the layer, for example you override the default stat associated with the layer.
• Other arguments passed on to the stat.

Details
Plot Missing Data Points

Note
Warning message if na.rm = T is supplied.

See Also
[gg_miss_case()][gg_miss_case_cumsum()][gg_miss_fct()][gg_miss_span()][gg_miss_var()][gg_miss_var_cumsum()][gg_miss_which()]

Examples

## Not run:
library(ggplot2)

# using regular geom_point()
ggplot(airquality,
   aes(x = Ozone,
       y = Solar.R)) +
geom_point()

# using geom_miss_point()
ggplot(airquality, 
   aes(x = Ozone, 
       y = Solar.R)) +
 geom_miss_point()

# using facets

ggplot(airquality, 
   aes(x = Ozone, 
       y = Solar.R)) +
 geom_miss_point() +
 facet_wrap(~Month)

## End(Not run)

---

**gg_miss_case**  
*Plot the number of missings per case (row)*

**Description**

This is a visual analogue to miss_case_summary. It draws a ggplot of the number of missings in each case (row). A default minimal theme is used, which can be customised as normal for ggplot.

**Usage**

```r
gg_miss_case(x, facet, order_cases = TRUE, show_pct = FALSE)
```

**Arguments**

- `x`  
  data.frame

- `facet`  
  (optional) a single bare variable name, if you want to create a faceted plot.

- `order_cases`  
  logical Order the rows by missingness (default is FALSE - no order).

- `show_pct`  
  logical Show the percentage of cases

**Value**

a ggplot object depicting the number of missings in a given case.

**See Also**

- `geom_miss_point()`  
- `gg_miss_case_cumsum`  
- `gg_miss_fct()`  
- `gg_miss_span()`  
- `gg_miss_var()`  
- `gg_miss_var_cumsum()`  
- `gg_miss_which()`
**Examples**

```r
gg_miss_case(airquality)
## Not run:
library(ggplot2)
gg_miss_case(airquality) + labs(x = "Number of Cases")
gg_miss_case(airquality, show_pct = TRUE)
gg_miss_case(airquality, order_cases = FALSE)
gg_miss_case(airquality, facet = Month)
gg_miss_case(airquality, facet = Month, order_cases = FALSE)
gg_miss_case(airquality, facet = Month, show_pct = TRUE)
## End(Not run)
```

---

**gg_miss_case_cumsum**  
*Plot of cumulative sum of missing for cases*

**Description**

A plot showing the cumulative sum of missing values for cases, reading the rows from the top to bottom. A default minimal theme is used, which can be customised as normal for ggplot.

**Usage**

```r
gg_miss_case_cumsum(x, breaks = 20)
```

**Arguments**

- `x`: a dataframe
- `breaks`: the breaks for the x axis default is 20

**Value**

A ggplot object depicting the number of missings

**See Also**

`geom_miss_point()` `gg_miss_case()` `gg_miss_fct()` `gg_miss_span()` `gg_miss_var()` `gg_miss_var_cumsum()` `gg_miss_which()`

**Examples**

```r
gg_miss_case_cumsum(airquality)
```
gg_miss_fct

Plot the number of missings for each variable, broken down by a factor

Description

This function draws a ggplot plot of the number of missings in each column, broken down by a categorical variable from the dataset. A default minimal theme is used, which can be customised as normal for ggplot.

Usage

gg_miss_fct(x, fct)

Arguments

x data.frame
fct column containing the factor variable to visualise

Value

ggplot object depicting the % missing of each factor level for each variable.

See Also

geom_miss_point() gg_miss_case() gg_miss_case_cumsum gg_miss_span() gg_miss_var()

Examples

gg_miss_fct(x = riskfactors, fct = marital)
## Not run:
library(ggplot2)
gg_miss_fct(x = riskfactors, fct = marital) + labs(title = "NA in Risk Factors and Marital status")

## End(Not run)
gg_miss_span

Plot the number of missings in a given repeating span

Description

gg_miss_span is a replacement function to imputeTS::plotNA.distributionBar(tsNH4, breaksize = 100), which shows the number of missings in a given span, or breaksize. A default minimal theme is used, which can be customised as normal for ggplot.

Usage

gg_miss_span(data, var, span_every, facet)

Arguments

data data.frame
var a bare unquoted variable name from data.
span_every integer describing the length of the span to be explored
facet (optional) a single bare variable name, if you want to create a faceted plot.

Value

ggplot2 showing the number of missings in a span (window, or breaksize)

See Also

geom_miss_point() gg_miss_case() gg_miss_case_cumsum gg_miss_fct() gg_miss_var()
gg_miss_var_cumsum() gg_miss_which()

Examples

miss_var_span(pedestrian, hourly_counts, span_every = 3000)
## Not run:
library(ggplot2)
gg_miss_span(pedestrian, hourly_counts, span_every = 3000)
gg_miss_span(pedestrian, hourly_counts, span_every = 3000, facet = sensor_name)
# works with the rest of ggplot
gg_miss_span(pedestrian, hourly_counts, span_every = 3000) + labs(x = "custom")
gg_miss_span(pedestrian, hourly_counts, span_every = 3000) + theme_dark()

## End(Not run)
**Description**

Upset plots are a way of visualising common sets, `gg_miss_upset` shows the number of missing values for each of the sets of data. The default option of `gg_miss_upset` is taken from `UpSetR::upset` - which is to use up to 5 sets and up to 40 interactions. We also set the ordering to by the frequency of the intersections. Setting `nsets = 5` means to look at 5 variables and their combinations. The number of combinations or rather intersections is controlled by `nintersects`. If there are 40 intersections, there will be 40 combinations of variables explored. The number of sets and intersections can be changed by passing arguments `nsets = 10` to look at 10 sets of variables, and `nintersects = 50` to look at 50 intersections.

**Usage**

```r
gg_miss_upset(data, order.by = "freq", ...)
```

**Arguments**

- `data` data.frame
- `order.by` (from `UpSetR::upset`) How the intersections in the matrix should be ordered by. Options include frequency (entered as "freq"), degree, or both in any order. See `?UpSetR::upset` for more options
- `...` arguments to pass to upset plot - see `?UpSetR::upset`

**Value**

a ggplot visualisation of missing data

**Examples**

```r
## Not run:
gg_miss_upset(airquality)
gg_miss_upset(riskfactors)
gg_miss_upset(riskfactors, nsets = 10)
gg_miss_upset(riskfactors, nsets = 10, nintersects = 10)

## End(Not run)
```
**gg_miss_var**

*Plot the number of missings for each variable*

**Description**

This is a visual analogue to *miss_var_summary*. It draws a ggplot of the number of missings in each variable, ordered to show which variables have the most missing data. A default minimal theme is used, which can be customised as normal for ggplot.

**Usage**

```r
gg_miss_var(x, facet, show_pct = FALSE)
```

**Arguments**

- `x` a dataframe
- `facet` (optional) bare variable name, if you want to create a faceted plot.
- `show_pct` logical shows the number of missings (default), but if set to TRUE, it will display the proportion of missings.

**Value**

a ggplot object depicting the number of missings in a given column

**See Also**

`geom_miss_point()` `gg_miss_case()` `gg_miss_case_cumsum` `gg_miss_fct()` `gg_miss_span()`

`gg_miss_var()` `gg_miss_var_cumsum()` `gg_miss_which()`

**Examples**

```r
gg_miss_var(airquality)
## Not run:
library(ggplot2)
gg_miss_var(airquality) + labs(y = "Look at all the missing ones")
gg_miss_var(airquality, Month)
gg_miss_var(airquality, Month, show_pct = TRUE)
gg_miss_var(airquality, Month, show_pct = TRUE) + ylim(0, 100)
## End(Not run)
```
gg_miss_var_cumsum  
Plot of cumulative sum of missing value for each variable

Description
A plot showing the cumulative sum of missing values for each variable, reading columns from the left to the right of the initial dataframe. A default minimal theme is used, which can be customised as normal for ggplot.

Usage
gg_miss_var_cumsum(x)

Arguments
x  
a data.frame

Value
a ggplot object showing the cumulative sum of missings over the variables

See Also
geom_miss_point() gg_miss_case() gg_miss_case_cumsum gg_miss_fct() gg_miss_span() gg_miss_var() gg_miss_which()

Examples

gg_miss_var_cumsum(airquality)

---

gg_miss_which  
Plot which variables contain a missing value

Description
This plot produces a set of rectangles indicating whether there is a missing element in a column or not. A default minimal theme is used, which can be customised as normal for ggplot.

Usage
gg_miss_which(x)

Arguments
x  
a dataframe
Value

a ggplot object of which variables contains missing values

See Also

geom_miss_point() gg_miss_case() gg_miss_case_cumsum gg_miss_fct() gg_miss_span()
gg_miss_var() gg_miss_var_cumsum() gg_miss_which()

Examples

gg_miss_which(airquality)

Description

This is a wrapper to facilitate the grouped_df S3 method.

Usage

group_by_fun(data, .fun, ...)

Arguments

  data                   data.frame, which will be grouped
  .fun                   a function to apply
  ...                   additional arguments to be passed to map

Value

a dataframe with the function applied to each group

Examples

## Not run:
mis_case_table.grouped_df <- function(data){
group_by_fun(data,.fun = miss_case_table)
}
airquality %>%
group_by(Month) %>%
mis_case_table()

## End(Not run)
**impute_below_all**

*Impute data with values shifted 10 percent below range.*

**Description**

It can be useful in exploratory graphics to impute data outside the range of the data. `impute_below_all` imputes all variables with missings to have values 10 percent below the range for numeric values, and for character or factor values adds a new string or label. It is powered by `shadow_shift`, so please see the documentation for `shadow_shift()` to full details on the different implementations.

**Usage**

```r
impute_below_all(.tbl, prop_below = 0.1, jitter = 0.05, ...)
```

**Arguments**

- `.tbl`: a data.frame
- `prop_below`: the degree to shift the values. default is 0.1
- `jitter`: the amount of jitter to add. default is 0.05
- `...`: additional arguments

**Value**

an dataset with values imputed
Examples

# you can impute data like so:
airquality %>%
  impute_below_all()

# However, this does not show you WHERE the missing values are.
# to keep track of them, you want to use `bind_shadow()` first.

airquality %>%
  bind_shadow() %>%
  impute_below_all()

# This identifies where the missing values are located, which means you
# can do things like this:

## Not run:
library(ggplot2)
airquality %>%
  bind_shadow() %>%
  impute_below_all() %>%
  add_label_shadow() %>%
  ggplot(aes(x = Ozone,
             y = Solar.R,
             colour = any_missing)) +
  geom_point()
# Note that this ^^ is a long version of `geom_miss_point()`.

## End(Not run)

---

impute_below_at  Scoped variants of impute_below

Description

impute_below operates on all variables. To only impute variables that satisfy a specific condition, use the scoped variants, impute_below_at, and impute_below_if. To use _at effectively, you must know that _at` affects variables selected with a character vector, or with vars()`.

Usage

impute_below_at(.tbl, .vars, prop_below = 0.1, jitter = 0.05, ...)

Arguments

- **.tbl**: a data.frame
- **.vars**: variables to impute
prop_below  the degree to shift the values. default is
jitter      the amount of jitter to add. default is 0.05
...         extra arguments

Value

an dataset with values imputed

Examples

# select variables starting with a particular string.
impute_below_at(airquality,
    .vars = c("Ozone", "Solar.R"))

impute_below_at(airquality, .vars = 1:2)

## Not run:
library(dplyr)
impute_below_at(airquality,
    .vars = vars(Ozone))

library(ggplot2)
airquality %>%
    bind_shadow() %>%
    impute_below_at(vars(Ozone, Solar.R)) %>%
    add_label_shadow() %>%
    ggplot(aes(x = Ozone,
              y = Solar.R,
              colour = any_missing)) +
    geom_point()

## End(Not run)

impute_below_if  Scoped variants of impute_below

Description

impute_below operates on all variables. To only impute variables that satisfy a specific condition, use the scoped variants, impute_below_at, and impute_below_if.

Usage

impute_below_if(.tbl, .predicate, prop_below = 0.1, jitter = 0.05, ...)
**impute_mean**

**Arguments**

- `.tbl` : data.frame
- `.predicate` : A predicate function (such as `is.numeric`)
- `prop_below` : the degree to shift the values. default is 0.05
- `jitter` : the amount of jitter to add. default is 0.05
- `...` : extra arguments

**Value**

an dataset with values imputed

**Examples**

```r
gapminder %>% 
impute_below_if(.predicate = is.numeric)
```

---

**impute_mean**

*Impute the mean value into a vector with missing values*

**Description**

Impute the mean value into a vector with missing values

**Usage**

```r
impute_mean(x)
```

```r
## Default S3 method:
impute_mean(x)
```

```r
## S3 method for class 'factor'
impute_mean(x)
```

**Arguments**

- `x` : vector

**Value**

vector with mean values replaced
Examples

```r
vec <- rnorm(10)
vec[sample(1:10, 3)] <- NA
impute_mean(vec)
```

```r
impute_median(x)
```

Description

Impute the median value into a vector with missing values

Usage

```r
impute_median(x)
```

## Default S3 method:
impute_median(x)

## S3 method for class 'factor'
impute_median(x)

Arguments

- `x` vector

Value

vector with median values replaced

Examples

```r
vec <- rnorm(10)
vec[sample(1:10, 3)] <- NA
impute_median(vec)
```
is_shade

Detect if this is a shade

Description

This tells us if this column is a shade

Usage

is_shade(x)
are_shade(x)
any_shade(x)

Arguments

x a vector you want to test if is a shade

Value

logical - is this a shade?

Examples

xs <- shade(c(NA, 1, 2, "3"))

is_shade(xs)
are_shade(xs)
any_shade(xs)

aq_s <- as_shadow(airquality)

is_shade(aq_s)
are_shade(aq_s)
any_shade(aq_s)
any_shade(airquality)
label_missings Is there a missing value in the row of a dataframe?

Description

Creates a character vector describing presence/absence of missing values

Usage

label_missings(data, ..., missing = "Missing", complete = "Not Missing")

Arguments

data a dataframe or set of vectors of the same length
... extra variable to label
missing character a label for when values are missing - defaults to "Missing"
complete character a label for when values are complete - defaults to "Not Missing"

Value

character vector of "Missing" and "Not Missing".

See Also

bind_shadow() add_any_miss() add_label_missings() add_label_shadow() add_miss_cluster()
add_n_miss() add_prop_miss() add_shadow_shift() cast_shadow()

Examples

label_missings(airquality)

## Not run:
library(dplyr)

airquality %>%
  mutate(is_missing = label_missings(airquality)) %>%
  head()

airquality %>%
  mutate(is_missing = label_missings(airquality, 
                                        missing = "definitely missing", 
                                        complete = "absolutely complete")) %>%
  head()

## End(Not run)
**label_miss_1d**

| label_miss_1d | Label a missing from one column |

**Description**

Label whether a value is missing in a row of one columns.

**Usage**

`label_miss_1d(x1)`

**Arguments**

- **x1**: a variable of a dataframe

**Value**

a vector indicating whether any of these rows had missing values

**Note**

can we generalise label_miss to work for any number of variables?

**See Also**

`add_any_miss()` `add_label_missings()` `add_label_shadow()`

**Examples**

`label_miss_1d(airquality$Ozone)`

---

**label_miss_2d**

| label_miss_2d | label_miss_2d |

**Description**

Label whether a value is missing in either row of two columns.

**Usage**

`label_miss_2d(x1, x2)`
Arguments

x1  a variable of a dataframe
x2  another variable of a dataframe

Value

a vector indicating whether any of these rows had missing values

Examples

label_miss_2d(airquality$Ozone, airquality$Solar.R)

label_shadow(data, ..., missing = "Missing", complete = "Not Missing")

Arguments

data  data.frame
...  extra variable to label
missing  character a label for when values are missing - defaults to "Missing"
complete  character a label for when values are complete - defaults to "Not Missing"

Value

"Missing" or "Not Missing"
Proportion of variables containing missings or complete values

Description

Defunct. Please see `prop_miss_var()`, `prop_complete_var()`, `pct_miss_var()`, `pct_complete_var()`, `prop_miss_case()`, `prop_complete_case()`, `pct_miss_case()`, `pct_complete_case()`.

Usage

```r
miss_var_prop(...)
complete_var_prop(...)
miss_var_pct(...)
complete_var_pct(...)
miss_case_prop(...)
complete_case_prop(...)
miss_case_pct(...)
complete_case_pct(...)
```

Arguments

```r
...
```

Summarise the missingness in each case

Description

Provide a data.frame containing each case (row), the number and percent of missing values in each case.

Usage

```r
miss_case_cumsum(data)
```

Arguments

```r
data a dataframe
```
Value

a tibble containing the number and percent of missing data in each case

Examples

```
miss_case_cumsum(airquality)
```

```r
# Not run:
library(dplyr)
airquality %>%
  group_by(Month) %>%
  miss_case_cumsum()
```

```
# End(Not run)
```

---

**miss_case_summary**  
*Summarise the missingness in each case*

Description

Provide a summary for each case in the data of the number, percent missings, and cumulative sum of missings of the order of the variables. By default, it orders by the most missings in each variable.

Usage

```
miss_case_summary(data, order = TRUE, add_cumsum = FALSE, ...)
```

Arguments

- **data**: a data.frame
- **order**: a logical indicating whether or not to order the result by n_miss. Defaults to TRUE. If FALSE, order of cases is the order input.
- **add_cumsum**: logical indicating whether or not to add the cumulative sum of missings to the data. This can be useful when exploring patterns of nonresponse. These are calculated as the cumulative sum of the missings in the variables as they are first presented to the function.
- **...**: extra arguments

Value

a tibble of the percent of missing data in each case.
miss_case_table

See Also

pct_miss_case() prop_miss_case() pct_miss_var() prop_miss_var() pct_complete_case()
prop_complete_case() pct_complete_var() prop_complete_var() miss_prop_summary() miss_case_summary()
mis_var_table() miss_summary() miss_var_prop() miss_var_run() miss_var_span() miss_var_summary()
mis_var_table() n_complete() n_complete_row() n_miss() n_miss_row() pct_complete()
pct_miss() prop_complete() prop_complete_row() prop_miss()

Examples

miss_case_summary(airquality)

## Not run:
# works with group_by from dplyr
library(dplyr)
airquality %>%
group_by(Month) %>%
  miss_case_summary()

## End(Not run)

---

missing_case_table  
Tabulate missings in cases.

Description

Provide a tidy table of the number of cases with 0, 1, 2, up to n, missing values and the proportion of the number of cases those cases make up.

Usage

miss_case_table(data)

Arguments

data  a data frame

Value

a data frame

See Also

pct_miss_case() prop_miss_case() pct_miss_var() prop_miss_var() pct_complete_case()
pct_complete_case() pct_complete_var() prop_complete_var() miss_prop_summary() miss_case_summary()
mis_var_table() miss_summary() miss_var_prop() miss_var_run() miss_var_span() miss_var_summary()
mis_var_table() n_complete() n_complete_row() n_miss() n_miss_row() pct_complete()
pct_miss() prop_complete() prop_complete_row() prop_miss()
**Examples**

```r
miss_case_table(airquality)
## Not run:
library(dplyr)
airquality %>%
group_by(Month) %>%
miss_case_table()
## End(Not run)
```

---

**Description**

Return missing data info about the dataframe, the variables, and the cases. Specifically, returning how many elements in a dataframe contain a missing value, how many elements in a variable contain a missing value, and how many elements in a case contain a missing.

**Usage**

```r
miss_prop_summary(data)
```

**Arguments**

- `data` a dataframe

**Value**

a dataframe

**See Also**

`pct_miss_case()` `propmiss_case()` `pct_miss_var()` `propmiss_var()` `pct_complete_case()` `prop_complete_case()` `pct_complete_var()` `prop_complete_var()` `miss_prop_summary()` `miss_case_summary()` `miss_case_table()` `miss_summary()` `miss_var_run()` `miss_var_span()` `miss_var_summary()` `miss_var_table()`

**Examples**

```r
miss_prop_summary(airquality)
## Not run:
library(dplyr)
# respects dplyr::group_by
airquality %>% group_by(Month) %>% miss_prop_summary()
## End(Not run)
```
**miss_scan_count**

**Search and present different kinds of missing values**

**Description**

Searching for different kinds of missing values is really annoying. If you have values like -99 in your data, when they shouldn’t be there, or they should be encoded as missing, it can be difficult to ascertain if they are there, and if so, where they are. miss_scan_count makes it easier for users to search for particular occurrences of these values across their variables.

**Usage**

```
miss_scan_count(data, search)
```

**Arguments**

- `data`: data
- `search`: values to search for

**Value**

a dataframe of the occurrences of the values you searched for

**See Also**

- `pct_miss_case()`
- `prop_miss_case()`
- `pct_miss_var()`
- `prop_miss_var()`
- `pct_complete_case()`
- `prop_complete_case()`
- `pct_complete_var()`
- `prop_complete_var()`
- `miss_prop_summary()`
- `miss_case_summary()`
- `miss_case_table()`
- `miss_summary()`
- `miss_var_prop()`
- `miss_var_run()`
- `miss_var_span()`
- `miss_var_summary()`
- `miss_var_table()`

**Examples**

```r
dat_ms <- tibble::tribble(~x, ~y, ~z,
                          1, "A", -100,
                          3, "N/A", -99,
                          NA, NA, -98,
                          -99, "E", -101,
                          -98, "F", -1)

miss_scan_count(dat_ms,-99)
miss_scan_count(dat_ms,c(-99,-98))
miss_scan_count(dat_ms,c("-99","-98","N/A"))
miss_scan_count(dat_ms,common_na_strings)
```
miss_summary

Collate summary measures from naniar into one tibble

Description

miss_summary performs all of the missing data helper summaries and puts them into lists within a tibble

Usage

miss_summary(data, order = TRUE)

Arguments

data a dataframe
order whether or not to order the result by n_miss

Value

a tibble of missing data summaries

See Also

pct_miss_case() prop_miss_case() pct_miss_var() prop_miss_var() pct_complete_case()
prop_complete_case() pct_complete_var() prop_complete_var() miss_prop_summary() miss_case_summary()
mis_case_table() miss_summary() miss_var_prop() miss_var_run() miss_var_span() miss_var_summary()
mis_var_table() n_complete() n_complete_row() n_miss() n_miss_row() pct_complete()
pct_miss() prop_complete() prop_complete_row() prop_miss()

Examples

s_miss <- miss_summary(airquality)
s_miss$miss_df_prop
ds_miss$miss_case_table
ds_miss$miss_var_summary
# etc, etc, etc.

## Not run:
library(dplyr)
s_miss_group <- group_by(airquality, Month) %>% miss_summary()
s_miss_group$miss_df_prop
ds_miss_group$miss_case_table
# etc, etc, etc.

## End(Not run)
miss_var_cumsum

Cumulative sum of the number of missings in each variable

Description

Calculate the cumulative sum of number & percentage of missingness for each variable.

Usage

miss_var_cumsum(data)

Arguments

data a data.frame

Value

a tibble of the cumulative sum of missing data in each variable

See Also

pct_miss_case() prop_miss_case() pct_miss_var() prop_miss_var() pct_complete_case()
prop_complete_case() pct_complete_var() prop_complete_var() miss_prop_summary() miss_case_summary()
mis_case_table() miss_summary() miss_var_prop() miss_var_run() miss_var_span() miss_var_summary() miss_var_table()

Examples

miss_var_cumsum(airquality)
## Not run:
library(dplyr)

# respects dplyr::group_by

airquality %>%
group_by(Month) %>%
miss_var_cumsum()

## End(Not run)
**miss_var_run**

*Find the number of missing and complete values in a single run*

**Description**

It is useful to find the number of missing values that occur in a single run. The function `miss_var_run()`, returns a dataframe with the column names "run_length" and "is_na", which describe the length of the run, and whether that run describes a missing value.

**Usage**

```r
miss_var_run(data, var)
```

**Arguments**

- `data` data.frame
- `var` a bare variable name

**Value**

dataframe with column names "run_length" and "is_na", which describe the length of the run, and whether that run describes a missing value.

**See Also**

`pct_miss_case()` `prop_miss_case()` `pct_miss_var()` `prop_miss_var()` `pct_complete_case()` `prop_complete_case()` `pct_complete_var()` `prop_complete_var()` `miss_case_summary()` `miss_var_summary()` `miss_case_table()` `miss_var_prop()` `miss_var_run()` `miss_var_span()` `miss_case_summary()` `miss_var_table()` `n_complete()` `n_complete_row()` `n_miss()` `n_miss_row()` `pct_complete()` `prop_complete()` `prop_complete_row()` `prop_miss()`

**Examples**

```r
miss_var_run(pedestrian, hourly_counts)
```

```r
# Not run:
# find the number of runs missing/complete for each month
library(dplyr)

pedestrian %>%
  group_by(month) %>%
  miss_var_run(hourly_counts)

library(ggplot2)

# explore the number of missings in a given run
miss_var_run(pedestrian, hourly_counts) %>%
```
miss_var_span

To summarise the missing values in a time series object it can be useful to calculate the number of missing values in a given period. `miss_var_span` takes a `data.frame` object, a variable, and a `span_every` argument and returns a dataframe containing the number of missing values within each span.

**Usage**

```r
miss_var_span(data, var, span_every)
```

**Arguments**

- **data**: `data.frame`
- **var**: bare unquoted variable name of interest.
- **span_every**: integer describing the length of the span to be explored

**Value**

`dataframe` with variables `n_miss`, `n_complete`, `prop_miss`, and `prop_complete`, which describe the number, or proportion of missing or complete values within that given time span.
miss_var_summary

See Also

pct_miss_case() prop_miss_case() pct_miss_var() prop_miss_var() pct_complete_case()
prop_complete_case() pct_complete_var() prop_complete_var() miss_prop_summary() miss_case_summary()
mis_case_table() miss_summary() miss_var_prop() miss_var_run() miss_var_span() miss_var_summary()
mis_var_table()

Examples

miss_var_span(data = pedestrian,
              var = hourly_counts,
              span_every = 168)

## Not run:
library(dplyr)
pedestrian %>%
  group_by(month) %>%
  miss_var_span(var = hourly_counts,
               span_every = 168)

## End(Not run)

Description

Provide a summary for each variable of the number, percent missings, and cumulative sum of missings of the order of the variables. By default, it orders by the most missings in each variable.

Usage

miss_var_summary(data, order = FALSE, add_cumsum = FALSE, ...)

Arguments

data a data.frame
order a logical indicating whether to order the result by n_miss. Defaults to TRUE. If FALSE, order of variables is the order input.
add_cumsum logical indicating whether or not to add the cumulative sum of missings to the data. This can be useful when exploring patterns of nonresponse. These are calculated as the cumulative sum of the missings in the variables as they are first presented to the function.
... extra arguments

Value

a tibble of the percent of missing data in each variable
miss_var_table

Note

n_miss_cumsum is calculated as the cumulative sum of missings in the variables in the order that
they are given in the data when entering the function

See Also

pct_miss_case() prop_miss_case() pct_miss_var() prop_miss_var() pct_complete_case()
prop_complete_case() pct_complete_var() prop_complete_var() miss_prop_summary() miss_case_summary()
miss_case_table() miss_summary() miss_var_prop() miss_var_run() miss_var_span() miss_var_summary()
miss_var_table() n_complete() n_complete_row() n_miss() n_miss_row() pct_complete()
pct_miss() prop_complete() prop_complete_row() prop_miss()

Examples

miss_var_summary(airquality)
miss_var_summary(oceanbuoys, order = TRUE)

## Not run:
# works with group_by from dplyr
library(dplyr)
airquality %>%
  group_by(Month) %>%
  miss_var_summary()

## End(Not run)

---

**Description**

Provide a tidy table of the number of variables with 0, 1, 2, up to n, missing values and the proportion
of the number of variables those variables make up.

**Usage**

miss_var_table(data)

**Arguments**

data a dataframe

**Value**

a dataframe
miss_var_which

See Also

pct_miss_case() prop_miss_case() pct_miss_var() prop_miss_var() pct_complete_case() prop_complete_case() pct_miss_var() prop_miss_var() pct_complete_case() prop_complete_case() pct_miss_var() prop_miss_var() pct_complete_case() prop_complete_case() miss_prop_summary() miss_case_summary() miss_var_table() miss_summary() miss_var_prop() miss_var_run() miss_var_span() miss_var_summary() miss_var_table() n_complete() n_complete_row() n_miss() n_miss_row() pct_complete() pct_miss() prop_complete() prop_complete_row() prop_miss()

Examples

miss_var_table(airquality)
## Not run:
library(dplyr)
airquality %>%
  group_by(Month) %>%
  miss_var_table()
## End(Not run)

miss_var_which

Which variables contain missing values?

Description

It can be helpful when writing other functions to just return the names of the variables that contain missing values. miss_var_which returns a vector of variable names that contain missings. It will return NULL when there are no missings.

Usage

miss_var_which(data)

Arguments

data a data.frame

Value

character vector of variable names

Examples

miss_var_which(airquality)
miss_var_which(mtcars)
**n-var-case-complete**  
*The number of variables with complete values*

**Description**
This function calculates the number of variables that contain a complete value.

**Usage**
```r
n_var_complete(data)
n_case_complete(data)
```

**Arguments**
- `data`: data.frame

**Value**
integer number of complete values

**See Also**
- `n_var_miss()`

**Examples**
```r
# how many variables contain complete values?
n_var_complete(airquality)
n_case_complete(airquality)
```

---

**n-var-case-miss**  
*The number of variables or cases with missing values*

**Description**
This function calculates the number of variables or cases that contain a missing value.

**Usage**
```r
n_var_miss(data)
n_case_miss(data)
```

**Examples**
```r
# how many variables contain complete values?
n_var_miss(airquality)
n_case_miss(airquality)
```
nabular

Arguments

- data: data.frame

Value

- integer, number of missings

See Also

- n_var_complete()

Examples

```r
# how many variables contain missing values?
n_var_miss(airquality)
n_case_miss(airquality)
```

Description

Binding a shadow matrix to a regular dataframe converts it into nabular data, which makes it easier to visualise and work with missing data.

Usage

```
nabular(data, only_miss = FALSE, ...)
```

Arguments

- data: a dataframe
- only_miss: logical - if FALSE (default) it will bind a dataframe with all of the variables duplicated with their shadow. Setting this to TRUE will bind variables only those variables that contain missing values. See the examples for more details.
- ...: extra options to pass to recode_shadow() - a work in progress.

Value

- data with the added variable shifted and the suffix _NA

See Also

- bind_shadow()
Examples

```r
aq_nab <- nabular(airquality)
aq_s <- bind_shadow(airquality)

all.equal(aq_nab, aq_s)
```

Description

naniar is a package to make it easier to summarise and handle missing values in R. It strives to do this in a way that is as consistent with tidyverse principles as possible.

See Also

add_any_miss() add_label_missings() add_label_shadow() add_miss_cluster() add_n_miss()
add_prop_miss() add_shadow() add_shadow_shift() as_shadow() bind_shadow() cast_shadow()
cast_shadow_shift() cast_shadow_shift_label() draw_key_missing_point() gather_shadow()
geom_miss_point() gg_miss_case() gg_miss_case_cumsum() gg_miss_fct() gg_miss_span()
gg_miss_var() gg_miss_var_cumsum() gg_miss_which() label_miss_1d() label_miss_2d()
label_missings() pct_miss_case() prop_miss_case() pct_miss_var() prop_miss_var()
pct_complete_case() prop_complete_case() pct_complete_var() prop_complete_var() miss_prop_summary()
mis_case_summary() mis_case_table() miss_summary() miss_var_prop() miss_var_run()
mis_var_summary() mis_var_table() n_complete() n_complete_row()
n_miss() n_miss_row() n_pct_complete() n_pct_miss() prop_complete() prop_complete_row()
prop_miss() prop_miss_row() replace_to_na() replace_with_na() replace_with_na_all()
replace_with_na_at() replace_with_na_if() shadow_shift() stat_miss_point() vis_miss()
where_na()

new_shade  Create a new shade factor

Description

Create a new shade factor

Usage

```r
new_shade(x, extra_levels = NULL)
```
Arguments

\( x \)  

a factor to convert into a shade object

\( \text{extra\_levels} \)  

the extra levels to give to shade objects, such as "broken\_machine" and so on, which get converted into "NA\_broken\_machine".

Value

a new shade, which is built upon a factor

\[
\begin{array}{ll}
n\_\text{complete} & \text{Return the number of complete values} \\
\end{array}
\]

Description

A complement to \( n\_\text{miss} \)

Usage

\[
n\_\text{complete}(x)
\]

Arguments

\( x \)  

a vector

Value

numeric number of complete values

Examples

\[
\begin{align*}
n\_\text{complete}(\text{airquality}) \\
n\_\text{complete}(\text{airquality}\$Ozone)
\end{align*}
\]
n_complete_row

Return a vector of the number of complete values in each row

Description

Substitute for rowSums(!is.na(data)) but it also checks if input is NULL or is a dataframe

Usage

n_complete_row(data)

Arguments

data a dataframe

Value

numeric vector of the number of complete values in each row

See Also

pct_miss_case() prop_miss_case() pct_miss_var() prop_miss_var() pct_complete_case() prop_complete_case() pct_complete_var() prop_complete_var() miss_prop_summary() miss_case_summary() miss_case_table() miss_summary() miss_var_prop() miss_var_run() miss_var_span() miss_var_summary() miss_var_table() n_complete() n_complete_row() n_miss() n_miss_row() pct_complete() prop_complete() prop_complete_row() prop_miss()

Examples

n_complete_row(airquality)

n_miss

Return the number of missing values

Description

Substitute for sum(is.na(data))

Usage

n_miss(x)

Arguments

x a vector
Value

numeric the number of missing values

Examples

n_miss(airquality)
n_miss(airquality$Ozone)

---

**n_miss_row**

*Return a vector of the number of missing values in each row*

Description

Substitute for `rowSums(is.na(data))`, but it also checks if input is NULL or is a dataframe

Usage

\[
n\_miss\_row(data)\]

Arguments

- **data**: a dataframe

Value

numeric vector of the number of missing values in each row

See Also

`pct_miss_case()` `prop_miss_case()` `pct_miss_var()` `prop_miss_var()` `pct_complete_case()`
`prop_complete_case()` `pct_complete_var()` `prop_complete_var()` `miss_prop_summary()` `miss_case_summary()`
`miss_case_table()` `miss_summary()` `miss_var_prop()` `miss_var_run()` `miss_var_span()` `miss_var_summary()`
`miss_var_table()` `n_complete()` `n_complete_row()` `n_miss()` `n_miss_row()` `pct_complete()`
`pct_miss()` `prop_complete()` `prop_complete_row()` `prop_miss()`

Examples

\[
n\_miss\_row(airquality)\]
Real-time data from moored ocean buoys for improved detection, understanding and prediction of El Ni’o and La Ni’a. The data is collected by the Tropical Atmosphere Ocean project (https://www.pmel.noaa.gov/gtmba/pmel-theme/pacific-ocean-tao).

Usage
data(oceanbuoys)

Format
An object of class tbl_df (inherits from tbl, data.frame) with 736 rows and 8 columns.

Details
Format: a data frame with 736 observations on the following 8 variables.

year A numeric with levels 1993 1997.
latitude A numeric with levels -5 -2 0.
longitude A numeric with levels -110 -95.
sea_temp_c Sea surface temperature(degree Celsius), measured by the TAO buoys at one meter below the surface.
air_temp_c Air temperature(degree Celsius), measured by the TAO buoys three meters above the sea surface.
humidity Relative humidity(%), measured by the TAO buoys 3 meters above the sea surface.
wind_ew The East-West wind vector components(M/s). TAO buoys measure the wind speed and direction four meters above the sea surface. If it is positive, the East-West component of the wind is blowing towards the East. If it is negative, this component is blowing towards the West.
wind_ns The North-South wind vector components(M/s). TAO buoys measure the wind speed and direction four meters above the sea surface. If it is positive, the North-South component of the wind is blowing towards the North. If it is negative, this component is blowing towards the South.

Source
https://www.pmel.noaa.gov/tao/drupal/disdel/

See Also
library(MissingDataGUI) (data named "tao")
Examples

vis_miss(oceanbuoys)

# Look at the missingness in the variables
miss_var_summary(oceanbuoys)
## Not run:
# Look at the missingness in air temperature and humidity
library(ggplot2)
p <-
ggplot(oceanbuoys,
aes(x = air_temp_c,
y = humidity)) +
geom_miss_point()

p

# for each year?
p + facet_wrap(~year)

# this shows that there are more missing values in humidity in 1993, and
# more air temperature missing values in 1997

# see more examples in the vignette, "getting started with naniar".
## End(Not run)

---

pct-miss-complete-case

Percentage of cases that contain a missing or complete values.

Description

Calculate the percentage of cases (rows) that contain a missing or complete value.

Usage

pct_miss_case(data)

pct_complete_case(data)

Arguments

data a dataframe

Value

numeric the percentage of cases that contain a missing or complete value
pct-miss-complete-var

See Also

pct_miss_case() prop_miss_case() pct_miss_var() prop_miss_var() pct_complete_case()
pct_complete_var() prop_complete_var() miss_prop_summary() miss_case_summary()
miss_case_table() miss_summary() miss_var_prop() miss_var_run() miss_var_span() miss_var_summary()
miss_var_table()

Examples

pct_miss_case(airquality)
pct_complete_case(airquality)

---

pct-miss-complete-var  Percentage of variables containing missings or complete values

Description

Calculate the percentage of variables that contain a single missing or complete value.

Usage

pct_miss_var(data)
pct_complete_var(data)

Arguments

data  a dataframe

Value

numeric the percent of variables that contain missing or complete data

See Also

pct_miss_case() prop_miss_case() pct_miss_var() prop_miss_var() pct_complete_case()
pct_complete_var() prop_complete_var() miss_prop_summary() miss_case_summary()
miss_case_table() miss_summary() miss_var_prop() miss_var_run() miss_var_span() miss_var_summary()
miss_var_table()

Examples

prop_miss_var(airquality)
prop_complete_var(airquality)
pct_complete

Return the percent of complete values

Description
The complement to pct_miss

Usage
pct_complete(x)

Arguments
x vector or data.frame

Value
numeric percent of complete values

Examples

pct_complete(airquality)
pct_complete(airquality$Ozone)

pct_miss

Return the percent of missing values

Description
This is shorthand for mean(is.na(x)) * 100

Usage
pct_miss(x)

Arguments
x vector or data.frame

Value
numeric the percent of missing values in x
pedestrian

Examples

pct_miss(airquality)
pct_miss(airquality$Ozone)

---

pedestrian  Pedestrian count information around Melbourne for 2016

Description

This dataset contains hourly counts of pedestrians from 4 sensors around Melbourne: Birrarung Marr, Bourke Street Mall, Flagstaff station, and Spencer St-Collins St (south), recorded from January 1st 2016 at 00:00:00 to December 31st 2016 at 23:00:00. The data is made free and publicly available from https://data.melbourne.vic.gov.au/Transport-Movement/Pedestrian-volume-updated-monthly-b2ak-trbp

Usage

data(pedestrian)

Format

A tibble with 37,700 rows and 9 variables:

- **hourly_counts** (integer) the number of pedestrians counted at that sensor at that time
- **date_time** (POSIXct, POSIXt) The time that the count was taken
- **year** (integer) Year of record
- **month** (factor) Month of record as an ordered factor (1 = January, 12 = December)
- **month_day** (integer) Full day of the month
- **week_day** (factor) Full day of the week as an ordered factor (1 = Sunday, 7 = Saturday)
- **hour** (integer) The hour of the day in 24 hour format
- **sensor_id** (integer) the id of the sensor
- **sensor_name** (character) the full name of the sensor

Source

Examples

# explore the missingness with vis_miss
vis_miss(pedestrian)

# Look at the missingness in the variables
miss_var_summary(pedestrian)

## Not run:
# There is only missingness in hourly_counts
# Look at the missingness over a rolling window
library(ggplot2)
gg_miss_span(pedestrian, hourly_counts, span_every = 3000)

## End(Not run)

plotly_helpers

Plotly helpers (Convert a geom to a "basic" geom.)

Description

Helper functions to make it easier to automatically create plotly charts. This function makes it possible to convert ggplot2 geoms that are not included with ggplot2 itself. Users shouldn’t need to use this function. It exists purely to allow other package authors to write their own conversion method(s).

Usage

to_basic.GeomMissPoint(data, prestats_data, layout, params, p, ...)

Arguments

data the data returned by ggplot2::ggplot_build().
prestats_data the data before statistics are computed.
layout the panel layout.
params parameters for the geom, statistic, and 'constant' aesthetics
p a ggplot2 object (the conversion may depend on scales, for instance).
... currently ignored
prop-miss-complete-case

Proportion of cases that contain a missing or complete values.

Description
Calculate the proportion of cases (rows) that contain missing or complete values.

Usage
prop_miss_case(data)
prop_complete_case(data)

Arguments
data a dataframe

Value
numeric the proportion of cases that contain a missing or complete value

See Also
pct_miss_case() prop_miss_case() pct_miss_var() prop_miss_var() pct_complete_case()
prop_complete_case() pct_complete_var() prop_complete_var() miss_prop_summary() miss_case_summary()
mis_case_table() miss_summary() miss_var_prop() miss_var_run() miss_var_span() miss_var_summary()
mis_var_table()

Examples
prop_miss_case(airquality)
prop_complete_case(airquality)

prop-miss-complete-var

Proportion of variables containing missings or complete values

Description
Calculate the proportion of variables that contain a single missing or complete values.
Usage

prop_miss_var(data)

prop_complete_var(data)

Arguments

data a dataframe

Value

numeric the proportion of variables that contain missing or complete data

See Also

pct_miss_case() prop_miss_case() pct_miss_var() prop_miss_var() pct_complete_case()
prop_complete_case() pct_complete_var() prop_complete_var() miss_prop_summary() miss_case_summary()
miss_case_table() miss_summary() miss_var_prop() miss_var_run() miss_var_span() miss_var_summary()
miss_var_table()

Examples

prop_miss_var(airquality)
prop_complete_var(airquality)

prop_complete Return the proportion of complete values

Description

The complement to prop_miss

Usage

prop_complete(x)

Arguments

x vector or data.frame

Value

numeric proportion of complete values
prop_complete_row

Examples

```r
prop_complete(airquality)
prop_complete(airquality$Ozone)
```

prop_complete_row

Return a vector of the proportion of missing values in each row

Description

Substitute for `rowMeans(!is.na(data))`, but it also checks if input is NULL or is a dataframe

Usage

```r
prop_complete_row(data)
```

Arguments

data a dataframe

Value

numeric vector of the proportion of missing values in each row

See Also

```
pct_miss_case() prop_miss_case() pct_miss_var() prop_miss_var() pct_complete_case()
prop_complete_case() pct_complete_var() prop_complete_var() miss_prop_summary() miss_case_summary()
mis_case_table() miss_summary() miss_var_prop() miss_var_run() miss_var_span() miss_var_summary()
mis_var_table() n_complete() n_complete_row() n_miss() n_miss_row() pct_complete()
pct_miss() prop_complete() prop_complete_row() prop_miss()
```

Examples

```r
prop_complete_row(airquality)
```
prop_miss

Return the proportion of missing values

Description

This is shorthand for mean(is.na(x))

Usage

prop_miss(x)

Arguments

x vector or data.frame

Value

numeric the proportion of missing values in x

Examples

prop_miss(airquality)
prop_miss(airquality$Ozone)

prop_miss_row

Return a vector of the proportion of missing values in each row

Description

Substitute for rowMeans(is.na(data)), but it also checks if input is NULL or is a dataframe

Usage

prop_miss_row(data)

Arguments

data a dataframe

Value

numeric vector of the proportion of missing values in each row
**See Also**

pct_miss_case() prop_miss_case() pct_miss_var() prop_miss_var() pct_complete_case()
prop_complete_case() pct_complete_var() prop_complete_var() miss_prop_summary() miss_case_summary()
miss_case_table() miss_summary() miss_var_prop() miss_var_run() miss_var_span() miss_var_summary()
miss_var_table() n_complete() n_complete_row() n_miss() n_miss_row() pct_complete()
pct_miss() prop_complete() prop_complete_row() prop_miss()

**Examples**

```r
drop_miss_row(airquality)
```

---

**recode_shadow**

*Add special missing values to the shadow matrix*

**Description**

It can be useful to add special missing values, naniar supports this with the `recode_shadow` function.

**Usage**

```r
recode_shadow(data, ...)
```

**Arguments**

- `data` data.frame
- `...` A sequence of two-sided formulas as in dplyr::case_when, but when a wrapper function .where written around it.

**Value**

a dataframe with altered shadows

**Examples**

```r
df <- tibble::tribble(
  ~wind, ~temp,
  -99, 45,
  68, NA,
  72, 25
)
dfs <- bind_shadow(df)
dfs
```
replace_to_na

Replace values with missings

Description
This function is Defunct, please see `replace_with_na()`.

Usage
```r
replace_to_na(...)```

Arguments

... additional arguments for methods.

Value
values replaced by NA

replace_with_na

Replace values with missings

Description
Specify variables and their values that you want to convert to missing values. This is a complement to `tidyr::replace_na`.

Usage
```r
replace_with_na(data, replace = list(), ...)```

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>data</td>
<td>A data.frame</td>
</tr>
<tr>
<td>replace</td>
<td>A named list given the NA to replace values for each column</td>
</tr>
<tr>
<td>...</td>
<td>additional arguments for methods. Currently unused</td>
</tr>
</tbody>
</table>
replace_with_na_all

Value

Dataframe with values replaced by NA.

See Also

replace_with_na() replace_with_na_all() replace_with_na_at() replace_with_na_if()

Examples

dat_ms <- tibble::tribble(~x, ~y, ~z,
  1, "A", -100,
  3, "N/A", -99,
  NA, NA, -98,
  -99, "E", -101,
  -98, "F", -1)

replace_with_na(dat_ms,
               replace = list(x = -99))

replace_with_na(dat_ms,
               replace = list(x = c(-99, -98)))

replace_with_na(dat_ms,
               replace = list(x = c(-99, -98),
                              y = c("N/A"),
                              z = c(-101)))

replace_with_na_all

Replace all values with NA where a certain condition is met

Description

This function takes a dataframe and replaces all values that meet the condition specified as an NA value, following a special syntax.

Usage

replace_with_na_all(data, condition)

Arguments

data A dataframe
condition A condition required to be TRUE to set NA. Here, the condition is specified with a formula, following the syntax: ~.x {condition}. For example, writing ~.x < 20 would mean "where a variable value is less than 20, replace with NA".
Examples

dat_ms <- tibble::tribble(~x, ~y, ~z,
  1, "A", -100,
  3, "N/A", -99,
  NA, NA, -98,
  -99, "E", -101,
  -98, "F", -1)

dat_ms
# replace all instances of -99 with NA
replace_with_na_all(data = dat_ms,
  condition = ~.x == -99)

# replace all instances of -99 or -98, or "N/A" with NA
replace_with_na_all(dat_ms,
  condition = ~.x %in% c(-99, -98, "N/A"))

# replace all instances of common na strings
replace_with_na_all(dat_ms,
  condition = ~.x %in% common_na_strings)

# where works with functions
replace_with_na_all(airquality, ~ sqrt(.x) < 5)

---

**replace_with_na_at**  
Replace specified variables with NA where a certain condition is met

**Description**

Replace specified variables with NA where a certain condition is met

**Usage**

`replace_with_na_at(data, .vars, condition)`

**Arguments**

- `data`  
  dataframe

- `.vars`  
  A character string of variables to replace with NA values

- `condition`  
  A condition required to be TRUE to set NA. Here, the condition is specified with a formula, following the syntax: `~.x {condition}`. For example, writing `~.x < 20` would mean "where a variable value is less than 20, replace with NA".

**Value**

a dataframe
replace_with_na_if

Examples

dat_ms <- tibble::tribble(~x, ~y, ~z,
  1, "A", -100,
  3, "N/A", -99,
  NA, NA, -98,
  -99, "E", -101,
  -98, "F", -1)

dat_ms

replace_with_na_at(data = dat_ms,
  .vars = "x",
  condition = ~.x == -99)

replace_with_na_at(data = dat_ms,
  .vars = c("x","z"),
  condition = ~.x == -99)

# replace using values in common_na_strings
replace_with_na_at(data = dat_ms,
  .vars = c("x","z"),
  condition = ~.x %in% common_na_strings)

---

replace_with_na_if  Replace values with NA based on some condition, for variables that meet some predicate

Description

Replace values with NA based on some condition, for variables that meet some predicate

Usage

replace_with_na_if(data, .predicate, condition)

Arguments

data  Dataframe

.predicate  A predicate function to be applied to the columns or a logical vector.

condition  A condition required to be TRUE to set NA. Here, the condition is specified with a formula, following the syntax: ~.x {condition}. For example, writing ~.x < 20 would mean "where a variable value is less than 20, replace with NA".

Value

Dataframe
Examples

dat_ms <- tibble::tribble(~x, ~y, ~z,
1, "A", -100,
3, "N/A", -99,
NA, NA, -98,
-99, "E", -101,
-98, "F", -1)

dat_ms

replace_with_na_if(data = dat_ms,
  .predicate = is.character,
  condition = ~.x == "N/A")
replace_with_na_if(data = dat_ms,
  .predicate = is.character,
  condition = ~.x %in% common_na_strings)

replace_with_na(dat_ms,
  to_na = list(x = c(-99, -98),
               y = c("N/A"),
               z = c(-101)))

---


Description

The data is a subset of the 2009 survey from BRFSS, an ongoing data collection program designed to measure behavioral risk factors for the adult population (18 years of age or older) living in households.

Usage

data(riskfactors)

Format

An object of class tbl_df (inherits from tbl.data.frame) with 245 rows and 34 columns.

Source

https://www.cdc.gov/brfss/annual_data/annual_2009.htm
See Also
the codebook: https://www.cdc.gov/brfss/annual_data/annual_2009.htm
Format: a data frame with 245 observations on the following 34 variables.

state A factor with 52 levels. The labels and states corresponding to the labels are as follows:
1:Alabama, 2:Alaska, 4:Arizona, 5:Arkansas, 6:California, 8:Colorado, 9:Connecticut, 10:Delaware,
19:Iowa, 20:Kansas, 21:Kentucky, 22:Louisiana, 23:Maine, 24:Maryland, 25:Massachusetts,
26:Michigan, 27:Minnesota, 28:Mississippi, 2:Missouri, 30:Montana, 31:Nebraska, 32:Nevada,
Dakota, 39:Ohio, 40:Oklahoma, 41:Oregon, 42:Pennsylvania, 44:Rhode Island, 45:South Carolina,
sex A factor with levels Male Female.
age A numeric vector from 7 to 97.
weight_lbs The weight without shoes in pounds.
height_inch The weight without shoes in inches.
bmi Body Mass Index (BMI). Computed by weight in Kilogram / (height in Meters * height in
Meters). Missing if any of weight or height is missing.
marital A factor with levels Married Divorced Widowed Separated NeverMarried UnmarriedCouple.
pregnant Whether pregnant now with two levels Yes and No.
children A numeric vector giving the number of children less than 18 years of age in household.
education A factor with the education levels 1 2 3 4 5 6 as 1: Never attended school or only
kindergarten; 2: Grades 1 through 8 (Elementary); 3: Grades 9 through 11 (Some high school);
4: Grade 12 or GED (High school graduate); 5: College 1 year to 3 years (Some college or
technical school); 6: College 4 years or more (College graduate).
employment A factor showing the employment status with levels 1 2 3 4 5 7 8. The labels mean
– 1: Employed for wages; 2: Self-employed; 3: Out of work for more than 1 year; 4: Out of
work for less that 1 year; 5: A homemaker; 6: A student; 7: Retired; 8: Unable to work.
income The annual household income from all sources with levels <10k 10-15k 15-20k 20-25k
25-35k 35-50k 50-75k >75k Don't know Refused.
veteran A factor with levels 1 2 3 4 5. The question for this variable is: Have you ever served on
active duty in the United States Armed Forces, either in the regular military or in a National
Guard or military reserve unit? Active duty does not include training for the Reserves or
National Guard, but DOES include activation, for example, for the Persian Gulf War. And
the labels are meaning: 1: Yes, now on active duty; 2: Yes, on active duty during the last 12
months, but not now; 3: Yes, on active duty in the past, but not during the last 12 months; 4:
No, training for Reserves or National Guard only; 5: No, never served in the military.
hispanic A factor with levels Yes No corresponding to the question: are you Hispanic or Latino?
health_general Answer to question “in general your health is” with levels Excellent VeryGood
Good Fair Poor Refused.
health_physical The number of days during the last 30 days that the respondent’s physical health
was not good. -7 is for "Don’t know/Not sure”, and -9 is for "Refused".
health_mental  The number of days during the last 30 days that the respondent’s mental health was not good. -7 is for "Don’t know/Not sure", and -9 is for "Refused".

health_poor  The number of days during the last 30 days that poor physical or mental health keep the respondent from doing usual activities, such as self-care, work, or recreation. -7 is for "Don’t know/Not sure", and -9 is for "Refused".

health_cover  Whether having any kind of health care coverage, including health insurance, prepaid plans such as HMOs, or government plans such as Medicare. The answer has two levels: Yes and No.

provide_care  Whether providing any such care or assistance to a friend or family member during the past month, with levels Yes and No.

activity_limited  Whether being limited in any way in any activities because of physical, mental, or emotional problems, with levels Yes and No.

drink_any  Whether having had at least one drink of any alcoholic beverage such as beer, wine, a malt beverage or liquor during the past 30 days, with levels Yes and No.

drink_days  The number of days during the past 30 days that the respondent had at least one drink of any alcoholic beverage. -7 is for "Don’t know/Not sure", and -9 is for "Refused".

drink_avg  The number of drinks on the average the respondent had on the days when he/she drank, during the past 30 days. -7 is for "Don’t know/Not sure", and -9 is for "Refused".

smoke_100  Whether having smoked at least 100 cigarettes in the entire life, with levels Yes and No.

smoke_days  The frequency of days now smoking, with levels Everyday Somedays and NotAtAll(not at all).

smoke_stop  Whether having stopped smoking for one day or longer during the past 12 months because the respondent was trying to quit smoking, with levels Yes and No.

smoke_last  A factor with levels 3 4 5 6 7 8 corresponding to the question: how long has it been since last smoking cigarettes regularly? The labels mean: 3: Within the past 6 months (3 months but less than 6 months ago); 4: Within the past year (6 months but less than 1 year ago); 5: Within the past 5 years (1 year but less than 5 years ago); 6: Within the past 10 years (5 years but less than 10 years ago); 7: 10 years or more; 8: Never smoked regularly.

diet_fruit  The number of fruit the respondent eat every year, not counting juice. -7 is for "Don’t know/Not sure", and -9 is for "Refused".

diet_salad  The number of servings of green salad the respondent eat every year. -7 is for "Don’t know/Not sure", and -9 is for "Refused".

diet_potato  The number of servings of potatoes, not including french fries, fried potatoes, or potato chips, that the respondent eat every year. -7 is for "Don’t know/Not sure", and -9 is for "Refused".

diet_carrot  The number of carrots the respondent eat every year. -7 is for "Don’t know/Not sure", and -9 is for "Refused".

diet_vegetable  The number of servings of vegetables the respondent eat every year, not counting carrots, potatoes, or salad. -7 is for "Don’t know/Not sure", and -9 is for "Refused".

diet_juice  The number of fruit juices such as orange, grapefruit, or tomato that the respondent drink every year. -7 is for "Don’t know/Not sure", and -9 is for "Refused".

library(MissingDataGUI) (named brfss)
Examples

```r
vis_miss(riskfactors)
# Look at the missingness in the variables
miss_var_summary(riskfactors)
# and now as a plot
gg_miss_var(riskfactors)
```

```r
## Not run:
# Look at the missingness in bmi and poor health
library(ggplot2)
p <- ggplot(riskfactors, 
  aes(x = health_poor, 
       y = bmi)) + 
  geom_miss_point()

p

# for each sex?
p + facet_wrap(~sex)
# for each education bracket?
p + facet_wrap(~education)
```

```r
## End(Not run)
```

---

**scoped_impute_mean**

**Scoped variants of impute_mean**

**Description**

`impute_mean` imputes the mean for a vector. To get it to work on all variables, use `impute_mean_all`. To only impute variables that satisfy a specific condition, use the scoped variants, `impute_below_at`, and `impute_below_if`. To use `at` effectively, you must know that `at` affects variables selected with a character vector, or with `vars()`.

**Usage**

```r
impute_mean_all(.tbl)
impute_mean_at(.tbl, .vars)
impute_mean_if(.tbl, .predicate)
```
Arguments

- `.tbl` a data.frame
- `.vars` variables to impute
- `.predicate` variables to impute

Value

- an dataset with values imputed

Examples

```r
# select variables starting with a particular string.
impute_mean_all(airquality)

impute_mean_at(airquality,
  .vars = c("Ozone", "Solar.R"))

## Not run:
library(dplyr)
impute_mean_at(airquality,
  .vars = vars(Ozone))
impute_mean_if(airquality,
  .predicate = is.numeric)

library(ggplot2)
airquality %>%
  bind_shadow() %>%
impute_mean_all() %>%
  add_label_shadow() %>%
  ggplot(aes(x = Ozone,
    y = Solar.R,
    colour = any_missing)) +
  geom_point()

## End(Not run)
```

scoped-impute_median

Scoped variants of `impute_median`

Description

`impute_median` imputes the median for a vector. To get it to work on all variables, use `impute_median_all`. To only impute variables that satisfy a specific condition, use the scoped variants, `impute_below_at` and `impute_below_if`. To use `at` effectively, you must know that `at` affects variables selected with a character vector, or with `vars()`.
Usage

impute_median_all(.tbl)
impute_median_at(.tbl, .vars)
impute_median_if(.tbl, .predicate)

Arguments

.tbl a data.frame
.vars variables to impute
.predicate variables to impute

Value

an dataset with values imputed

Examples

# select variables starting with a particular string.
impute_median_all(airquality)
impute_median_at(airquality,
   .vars = c("Ozone", "Solar.R"))

## Not run:
library(dplyr)
impute_median_at(airquality,
   .vars = vars(Ozone))
impute_median_if(airquality,
   .predicate = is.numeric)

library(ggplot2)
airquality %>% bind_shadow() %>%
impute_median_all() %>%
add_label_shadow() %>%
ggpplot(aes(x = Ozone,
   y = Solar.R,
   colour = any_missing)) +
geom_point()

## End(Not run)
**shade**  
*Create new levels of missing*

**Description**

Returns (at least) factors of !NA and NA, where !NA indicates a datum that is not missing, and NA indicates missingness. It also allows you to specify some new missings, if you like. This function is what powers the factor levels in `as_shadow()`.

**Usage**

```r
shade(x, ..., extra_levels = NULL)
```

**Arguments**

- `x`: a vector
- `...`: additional levels of missing to add
- `extra_levels`: extra levels you might to specify for the factor.

**Examples**

```r
df <- tibble::tribble(
  ~wind, ~temp,
  -99, 45,
  68, NA,
  72, 25
)
shade(df$wind)
shade(df$wind, inst_fail = -99)
```

---

**shadow_expand_relevel**  
*Expand and relevel a shadow column with a new suffix*

**Description**

Internal function to handle appropriate expansion and releveling of shadow variables.

**Usage**

```r
shadow_expand_relevel(.var, suffix)
```
shadow_long

Arguments

.var a variable in a data.frame
suffix a character suffix to add to NA_, e.

Value

a factor with expanded levels

Examples

df <- tibble::tribble(
  ~wind, ~temp,
  -99, 45,
  68, NA,
  72, 25
)

dfs <- bind_shadow(df)
test_shade <- dfs$wind_NA

# shadow_expand_relevel(test_shade, "weee")

# dfs %>%
# mutate(temp_NA = shadow_expand_relevel(temp_NA, "weee"))

# test that this breaks
# shadow_expand_relevel(airquality, "weee")

shadow_long

Reshape shadow data into a long format

Description

Once data is in nabular form, where the shadow is bound to the data, it can be useful to reshape it into a long format with the columns

Usage

shadow_long(shadow_data, ..., only_main_vars = TRUE)

Arguments

shadow_data a data.frame
... bare name of variables that you want to focus on
only_main_vars logical - do you want to filter down to main variables?
shadow_shift

Value
data in long format, with columns variable, value, variable_NA, and value_NA.

Examples

```r
aq_shadow <- bind_shadow(airquality)
shadow_long(aq_shadow)
# then filter only on Ozone
shadow_long(aq_shadow, Ozone)
shadow_long(aq_shadow, Ozone, Solar.R)
```

---

**shadow_shift**

*Shift missing values to facilitate missing data exploration/visualisation*

Description

*shadow_shift* transforms missing values to facilitate visualisation, and has different behaviour for different types of variables. For numeric variables, the values are shifted to 10% below the minimum value for a given variable plus some jittered noise, to separate repeated values, so that missing values can be visualised along with the rest of the data.

Usage

```r
shadow_shift(x, ...)
```

Arguments

- `x` a variable of interest to shift
- `...` extra arguments to pass

See Also

`add_shadow_shift()` `cast_shadow_shift()` `cast_shadow_shift_label()`

Examples

```r
airquality$Ozone
shadow_shift(airquality$Ozone)
## Not run:
library(dplyr)
airquality %>%
  mutate(Ozone_shift = shadow_shift(Ozone))
## End(Not run)
```
### shadow_shift.numeric

Shift (impute) numeric values for graphical exploration

#### Description

Shift (impute) numeric values for graphical exploration

#### Usage

```r
## S3 method for class 'numeric'
shadow_shift(
  x,
  prop_below = 0.1,
  jitter = 0.05,
  seed_shift = 2017 - 7 - 1 - 1850,
  ...
)
```

#### Arguments

- **x**: a variable of interest to shift
- **prop_below**: the degree to shift the values. default is
- **jitter**: the amount of jitter to add. default is 0.05
- **seed_shift**: a random seed to set, if you like
- **...**: extra arguments to pass

### stat_miss_point

#### Description

stat_miss_point adds a geometry for displaying missingness to geom_point

#### Usage

```r
stat_miss_point(
  mapping = NULL,
  data = NULL,
  prop_below = 0.1,
  jitter = 0.05,
  geom = "point",
  position = "identity",
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE,
  ...
)
```
Arguments

- **mapping**: Set of aesthetic mappings created by `ggplot2::aes()` or `ggplot2::aes_()`. If specified and `inherit.aes = TRUE` (the default), is combined with the default mapping at the top level of the plot. You only need to supply mapping if there isn’t a mapping defined for the plot.

- **data**: A data frame. If specified, overrides the default data frame defined at the top level of the plot.

- **prop_below**: the degree to shift the values. The default is 0.1

- **jitter**: the amount of jitter to add. The default is 0.05

- **geom, stat**: Override the default connection between `geom_point` and `stat_point`.

- **position**: Position adjustment, either as a string, or the result of a call to a position adjustment function

- **na.rm**: If `FALSE` (the default), removes missing values with a warning. If `TRUE` silently removes missing values.

- **show.legend**: logical. Should this layer be included in the legends? `NA`, the default, includes if any aesthetics are mapped. `FALSE` never includes, and `TRUE` always includes.

- **inherit.aes**: If `FALSE`, overrides the default aesthetics, rather than combining with them. This is most useful for helper functions that define both data and aesthetics and shouldn’t inherit behaviour from the default plot specification, e.g. borders.

- **...**: other arguments passed on to `ggplot2::layer()`. There are three types of arguments you can use here:
  - Aesthetics: to set an aesthetic to a fixed value, like `color = "red"` or `size = 3`.
  - Other arguments to the layer, for example you override the default `stat` associated with the layer.
  - Other arguments passed on to the stat.

---

**test_if_dataframe**

*Test if input is a data.frame*

**Description**

Test if input is a data.frame

**Usage**

`test_if_dataframe(x)`

**Arguments**

- **x**: object
Value

an error if input (x) is a data.frame

Examples

## Not run:
# success
test_if_dataframe(airquality)
# fail
my_test <- matrix(10)
test_if_dataframe(my_test)

## End(Not run)

test_if_missing Test if the input is Missing

Description

Test if the input is Missing

Usage

test_if_missing(x)

Arguments

x object

Value

an error if input (x) is not specified

Examples

## Not run:
# success
my_test <- x
test_if_null(my_test)
# fail
test_if_missing()

## End(Not run)
test_if_null  

Test if the input is NULL

Description
Test if the input is NULL

Usage

test_if_null(x)

Arguments

x  object

Value

an error if input (x) is NULL

Examples

## Not run:
# success
test_if_null(airquality)
#fail
my_test <- NULL
test_if_null(my_test)

## End(Not run)

unbinders  

Unbind (remove) shadow from data, and vice versa

Description
Remove the shadow variables (which end in _NA) from the data, or vice versa. This will also remove the nabular class from the data.

Usage

unbind_shadow(data)
unbind_data(data)

Arguments

data  data.frame containing shadow columns (created by bind_shadow())
Value

data.frame without shadow columns if using `unbind_shadow()`, or without the original data, if using `unbind_data()`.

Examples

```r
# bind shadow columns
aq_sh <- bind_shadow(airquality)

# print data
aq_sh

# remove shadow columns
unbind_shadow(aq_sh)

# remove data
unbind_data(aq_sh)

# errors when you don't use data with shadows
## Not run:
unbind_data(airquality)
unbind_shadow(airquality)
## End(Not run)
```

---

**update_shadow**

*Expand all shadow levels*

**Description**

Internal function to appropriately expand and relevel all shadow variables to include a new suffix

**Usage**

```r
update_shadow(data, suffix)
```

**Arguments**

- `data`: data.frame
- `suffix`: character vector

**Value**

data.frame with adjusted levels
Examples

```r
# Not run:
df <- tibble::tribble(~wind, ~temp,
                      -99, 45,
                      68, NA,
                      72, 25)

dfs <- bind_shadow(df)

# update_shadow(dfs, "weee")
# update_shadow(dfs, "weee") %>% what_levels()

## End(Not run)
```

---

**what_levels**

*check the levels of many things*

---

**Description**

This function is used internally to check what the levels are of the dataframe.

**Usage**

```r
what_levels(x)
```

**Arguments**

- `x` data.frame, usually

**Value**

a list containing the levels of everything

---

**where**

*Split a call into two components with a useful verb name*

---

**Description**

This function is used inside `recode_shadow` to help evaluate the formula call effectively. `where` is a special function designed for use in `recode_shadow`, and you shouldn’t use it outside of it.
where_na

Usage

.where(...)

Arguments

... case_when style formula

Value

a list of "condition" and "suffix" arguments

Examples

```r
## Not run:
df <- tibble::tribble(
  ~wind, ~temp, 
  -99, 45, 
  68, NA, 
  72, 25   
)

dfs <- bind_shadow(df)
recode_shadow(dfs, 
  temp = .where(wind == -99 ~ "bananas"))

## End(Not run)
```
Value

A matrix with columns "row" and "col", which refer to the row and column that identify the position of a missing value in a dataframe.

See Also

which_na()

Examples

where_na(airquality)
where_na(oceanbuoys$sea_temp_c)

Description

This function tells us which variables contain shade information.

Usage

which_are_shade(.tbl)

Arguments

.tbl: a data.frame or tbl.

Value

numeric - which column numbers contain shade information.

Examples

df_shadow <- bind_shadow(airquality)
which_are_shade(df_shadow)
### which_na

**Which elements contain missings?**

**Description**

Equivalent to `which(is.na())` - returns integer locations of missing values.

**Usage**

`which_na(x)`

**Arguments**

- `x` a dataframe

**Value**

integer locations of missing values.

**See Also**

`where_na()`

**Examples**

`which_na(airquality)`
Index

* datasets
  common_na_numbers, 19
  common_na_strings, 20
  GeomMissPoint, 22
  oceanbuoys, 59
  pedestrian, 63
  riskfactors, 74
  where (where), 88
  add_any_miss, 4
  add_any_miss(), 5–11, 17–19, 38, 39, 55
  add_label_missings, 6
  add_label_missings(), 5–11, 17–19, 38, 39, 55
  add_label_shadow, 7
  add_label_shadow(), 5–11, 17–19, 38, 39, 55
  add_miss_cluster, 8
  add_miss_cluster(), 5–11, 17–19, 38, 55
  add_n_miss, 8
  add_n_miss(), 5–8, 11, 38, 55
  add_prop_miss, 9
  add_prop_miss(), 5–11, 17–19, 38, 55
  add_shadow, 10
  add_shadow(), 55
  add_shadow_shift, 11
  add_shadow_shift(), 5–11, 17–19, 38, 55, 82
  add_span_counter, 12
  all-is-miss-complete, 12
  all_complete, 13
  all_complete(all-is-miss-complete), 12
  all_miss(all-is-miss-complete), 12
  all_miss(), 13
  all_na(all-is-miss-complete), 12
  any-na, 13
  any_complete(any-na), 13
  any_miss(any-na), 13
  any_na(any-na), 13
  any_row_miss, 14
  any_shade(is_shade), 37
  are_shade(is_shade), 37
  as_shadow, 14
  as_shadow(), 55
  as_shadow_upset, 15
  bind_shadow, 16
  bind_shadow(), 5–11, 17–19, 38, 54, 55, 86
  cast_shadow, 17
  cast_shadow(), 5–11, 38, 55
  cast_shadow_shift, 18
  cast_shadow_shift(), 17–19, 55, 82
  cast_shadow_shift_label, 18
  cast_shadow_shift_label(), 17–19, 55, 82
  common_na_numbers, 19
  common_na_strings, 20
  complete_case_pct
    (miss-pct-prop-defunct), 41
  complete_case_prop
    (miss-pct-prop-defunct), 41
  complete_var_pct
    (miss-pct-prop-defunct), 41
  complete_var_prop
    (miss-pct-prop-defunct), 41
  draw_key_missing_point(), 55
  gather_shadow, 21
  gather_shadow(), 55
  geom_miss_point, 22
  geom_miss_point(), 24–27, 29–31, 55
  GeomMissPoint, 22
  gg_miss_case, 24
  gg_miss_case(), 25–27, 29–31, 55
  gg_miss_case_cumsum, 24, 25, 26, 27, 29–31
  gg_miss_case_cumsum(), 55
  gg_miss_fct, 26
  gg_miss_fct(), 24, 25, 27, 29–31, 55
  gg_miss_span, 27
impute_mean_at
impute_mean_all
impute_mean_if
impute_below_if
impute_below_at
impute_below_all
impute_below

group_by_fun

miss_case_prop
miss_case_pct
miss_case_cumsum

miss-pct-prop-defunct

label_shadow

ggplot2::layer()

miss_case_summary

label_missings()

label_missings

label_miss_2d()

label_miss_1d()

is_shade

impute_median_at

impute_median_all
impute_median

impute_median_all(scoped-impute_median)
impute_median_at(scoped-impute_median)
impute_median_if(scoped-impute_median)
impute_median

impute_median_all(scoped-impute_median)
impute_median_at(scoped-impute_median)
impute_median_if(scoped-impute_median)

is_shade

miss_case_table()

miss_var_table()
miss_var_prop()
miss_var_cumsum
miss_var_span()
mis-var-case-miss

n_complete_row

n_complete()

n_case_complete(n-var-case-complete)

n_case_miss(n-var-case-miss)

n_complete()

n_miss_row

n_var_complete(n-var-case-complete)
n_var_complete()
n_var_miss(n-var-case-miss)
n_var_miss(), naniar, naniar-ggproto(naniar), nabular, nboyoos, oceanbuoys, new_shade, nvar_miss(), prop_complete_var(), prop_complete_row(), prop_complete_case()

prop_miss(), prop_miss_var(), prop_miss_row(), prop_miss_case(), prop_miss() (prop-miss-complete-case), recode_shadow, recode_shadow(), replace_to_na, replace_to_na(), replace_with_na, replace_with_na(), replace_with_na_all, replace_with_na_all(), replace_with_na_at, replace_with_na_at(), replace_with_na_if, replace_with_na_if(), riskfactors, scoped-impute_mean, scoped-impute_median, shade, shade_shift, shade_shift(), shade_long, shade_long.numeric, stat_miss_point, stat_miss_point(), StatMissPoint, StatMissPoint(GeomMissPoint), test_if_dataframe, test_if_dataframe(), test_if_missing, test_if_null, test_if_null(), to_basic.GeomMissPoint, to_basic.GeomMissPoint(plotly_helpers), unbind_data, unbind_data(), unbind_data(unbinders), unbind_data(), unbind_shadow, unbind_shadow(unbinders), unbinders, unbinders()
unbind_shadow(), 87
unbinders, 86
update_shadow, 87
vis_miss(), 55
what_levels, 88
where, 88
where_na, 89
where_na(), 55, 91
which_are_shade, 90
which_na, 91
which_na(), 90