

# Package ‘naniar’

October 13, 2022

**Type** Package

**Title** Data Structures, Summaries, and Visualisations for Missing Data

**Version** 0.6.1

**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](https://arxiv.org/abs/1809.02264)>.

**License** MIT + file LICENSE

**LazyData** TRUE

**ByteCompile** TRUE

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

**VignetteBuilder** knitr

**Depends** R (>= 3.1.2)

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

**Collate** 'add-cols.R' 'add-n-prop-miss.R' 'cast-shadows.R' 'data-common-na-numbers.R' 'data-common-na-strings.R' 'data-oceanbuoys.R' 'data-pedestrian.R' 'data-riskfactors.R' 'legend-draw.R' 'geom-miss-point.R' 'geom2plotly.R' 'gg-miss-case-cumsum.R' 'gg-miss-case.R' 'gg-miss-fct.R' 'gg-miss-span.R' 'gg-miss-upset.R' 'gg-miss-var-cumsum.R' 'gg-miss-var.R' 'gg-miss-which.R' 'helpers.R' 'impute-median.R' 'impute\_below.R' 'impute\_mean.R' 'label-miss.R' 'mcar-test.R' 'miss-complete-x-pct-prop.R' 'miss-prop-pct-summary.R' 'miss-scan-count.R' 'miss-x-cumsum.R' 'miss-x-run.R' 'miss-x-span.R' 'miss-x-summary.R' 'miss-x-table.R' 'n-prop-miss-complete-rows.R' 'n-prop-miss-complete.R' 'n-var-miss.R' 'nabular.R' 'naniar-ggproto.R' 'naniar-package.R' 'prop-pct-var-case-miss-complete.R'

'replace-to-na.R' 'replace-with-na.R'  
 'scoped-replace-with-na.R' 'shade.R' 'shadow-recode.R'  
 'shadow-shifters.R' 'shadows.R' 'stat-miss-point.R' 'utils.R'  
 'where-na.R'

**URL** <https://github.com/njtierney/naniar>

**BugReports** <https://github.com/njtierney/naniar/issues>

**Encoding** UTF-8

**RoxygenNote** 7.1.1

**Language** en-US

**NeedsCompilation** no

**Author** Nicholas Tierney [aut, cre] (<<https://orcid.org/0000-0003-1460-8722>>),  
 Di Cook [aut] (<<https://orcid.org/0000-0002-3813-7155>>),  
 Miles McBain [aut] (<<https://orcid.org/0000-0003-2865-2548>>),  
 Colin Fay [aut] (<<https://orcid.org/0000-0001-7343-1846>>),  
 Mitchell O'Hara-Wild [ctb],  
 Jim Hester [ctb],  
 Luke Smith [ctb],  
 Andrew Heiss [ctb] (<<https://orcid.org/0000-0002-3948-3914>>)

**Maintainer** Nicholas Tierney <nicholas.tierney@gmail.com>

**Repository** CRAN

**Date/Publication** 2021-05-14 10:20:02 UTC

## R topics documented:

add_any_miss . . . . .	4
add_label_missings . . . . .	6
add_label_shadow . . . . .	7
add_miss_cluster . . . . .	8
add_n_miss . . . . .	8
add_prop_miss . . . . .	9
add_shadow . . . . .	10
add_shadow_shift . . . . .	11
add_span_counter . . . . .	12
all-is-miss-complete . . . . .	12
any-na . . . . .	13
any_row_miss . . . . .	14
as_shadow . . . . .	14
as_shadow_upset . . . . .	15
bind_shadow . . . . .	16
cast_shadow . . . . .	17
cast_shadow_shift . . . . .	18
cast_shadow_shift_label . . . . .	18
common_na_numbers . . . . .	19
common_na_strings . . . . .	20

gather_shadow . . . . .	21
GeomMissPoint . . . . .	22
geom_miss_point . . . . .	22
gg_miss_case . . . . .	24
gg_miss_case_cumsum . . . . .	25
gg_miss_fct . . . . .	26
gg_miss_span . . . . .	27
gg_miss_upset . . . . .	28
gg_miss_var . . . . .	29
gg_miss_var_cumsum . . . . .	30
gg_miss_which . . . . .	30
group_by_fun . . . . .	31
impute_below . . . . .	32
impute_below_all . . . . .	32
impute_below_at . . . . .	33
impute_below_if . . . . .	34
impute_mean . . . . .	35
impute_median . . . . .	36
is_shade . . . . .	37
label_missings . . . . .	38
label_miss_1d . . . . .	39
label_miss_2d . . . . .	39
label_shadow . . . . .	40
mcar_test . . . . .	41
miss-pct-prop-defunct . . . . .	42
miss_case_cumsum . . . . .	43
miss_case_summary . . . . .	43
miss_case_table . . . . .	44
miss_prop_summary . . . . .	45
miss_scan_count . . . . .	46
miss_summary . . . . .	47
miss_var_cumsum . . . . .	48
miss_var_run . . . . .	49
miss_var_span . . . . .	50
miss_var_summary . . . . .	51
miss_var_table . . . . .	53
miss_var_which . . . . .	54
n-var-case-complete . . . . .	54
n-var-case-miss . . . . .	55
nabular . . . . .	56
naniar . . . . .	57
new_shade . . . . .	57
n_complete . . . . .	58
n_complete_row . . . . .	58
n_miss . . . . .	59
n_miss_row . . . . .	60
oceanbuoys . . . . .	60
pct-miss-complete-case . . . . .	62

pct-miss-complete-var . . . . .	63
pct_complete . . . . .	63
pct_miss . . . . .	64
pedestrian . . . . .	65
plotly_helpers . . . . .	66
prop-miss-complete-case . . . . .	66
prop-miss-complete-var . . . . .	67
prop_complete . . . . .	68
prop_complete_row . . . . .	68
prop_miss . . . . .	69
prop_miss_row . . . . .	70
recode_shadow . . . . .	70
replace_to_na . . . . .	71
replace_with_na . . . . .	72
replace_with_na_all . . . . .	73
replace_with_na_at . . . . .	74
replace_with_na_if . . . . .	75
riskfactors . . . . .	76
scoped-impute_mean . . . . .	79
scoped-impute_median . . . . .	80
shade . . . . .	81
shadow_expand_relevel . . . . .	82
shadow_long . . . . .	83
shadow_shift . . . . .	83
shadow_shift.numeric . . . . .	84
stat_miss_point . . . . .	85
test_if_dataframe . . . . .	86
test_if_missing . . . . .	87
test_if_null . . . . .	87
unbinders . . . . .	88
update_shadow . . . . .	89
what_levels . . . . .	90
where . . . . .	90
where_na . . . . .	91
which_are_shade . . . . .	92
which_na . . . . .	92

**Index****94**


---

 add\_any\_miss

---

*Add a column describing presence of any missing values*


---

**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.

**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 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	<i>Add a column describing whether there is a shadow</i>
------------------	--

---

### Description

Instead of focussing 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 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

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

---

add_miss_cluster	<i>Add a column that tells us which "missingness cluster" a row belongs to</i>
------------------	--

---

### Description

A way to extract the cluster of missingness that a group belongs to. For example, if you use `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

```
add_miss_cluster(data, cluster_method = "mcquitty", n_clusters = 2)
```

### Arguments

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

### 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

```
add_miss_cluster(airquality)
add_miss_cluster(airquality, n_clusters = 3)
add_miss_cluster(airquality, cluster_method = "ward.D", n_clusters = 3)
```

---

add_n_miss	<i>Add column containing number of missing data values</i>
------------	--

---

### Description

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



**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 <code>starts_with</code> , <code>contains</code> , <code>ends_with</code> , etc. By default will add <code>"_all"</code> to the label if left blank, otherwise will add <code>"_vars"</code> to distinguish that it has not been used on all of the variables.
label	character default is <code>"n_miss"</code> .

**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 <code>starts_with</code> , <code>contains</code> , <code>ends_with</code> , etc. By default will add <code>"_all"</code> to the label if left blank, otherwise will add <code>"_vars"</code> 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**

```
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 nanjar"
# 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 <code>starts_with</code> , <code>contains</code> , <code>ends_with</code> , etc.

**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)
```

---

add_shadow_shift	<i>Add a shadow shifted column to a dataset</i>
------------------	---

---

**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**

data	data.frame
...	One or more unquoted variable names separated by commas. These also respect the dplyr verbs starts_with, contains, ends_with, etc.
suffix	suffix to add to variable, defaults to "shift"

**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.

**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 both 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	<i>Helper function to determine whether there are any missings</i>
--------------	--

---

**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	<i>Create shadows</i>
-----------	-----------------------

---

**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

**Examples**

```
as_shadow(airquality)
```

---

as_shadow_upset	<i>Convert data into shadow format for doing an upset plot</i>
-----------------	--

---

**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**

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

---

bind_shadow	<i>Bind a shadow dataframe to original data</i>
-------------	---

---

### Description

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

### Usage

```
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 <a href="#">recode_shadow()</a> - a work in progress.

### Value

data with the added variable shifted and the suffix `_NA`

### Examples

```
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)
```



---

`cast_shadow`*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

<code>data</code>	<code>data.frame</code>
<code>...</code>	One or more unquoted variable names separated by commas. These respect the dplyr verbs <code>starts_with</code> , <code>contains</code> , <code>ends_with</code> , 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

```
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	<i>Add a shadow and a shadow_shift column to a dataset</i>
-------------------	--

---

**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	<i>Add a shadow column and a shadow shifted column to a dataset</i>
-------------------------	---

---

**Description**

Shift the values, add shadow, add missing label

**Usage**

```
cast_shadow_shift_label(data, ...)
```

**Arguments**

data            data.frame

...            One or more unquoted expressions separated by commas. These also respect the dplyr verbs "starts\_with", "contains", "ends\_with", etc.

**Value**

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

**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_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)
```

---

common\_na\_numbers            *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 [miss\\_scan\\_count\(\)](#) and [replace\\_with\\_na\(\)](#). The current list of numbers can be found by printing out `common_na_numbers`. 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. Common NA strings are in the data object `common_na_strings`.

**Usage**

```
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 25.

**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)
```

---

GeomMissPoint	<i>naniar-ggproto</i>
---------------	-----------------------

---

### 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.

---

geom_miss_point	<i>geom_miss_point</i>
-----------------	------------------------

---

### 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 10\ 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 <code>ggplot2::aes()</code> or <code>ggplot2::aes_()</code> . If specified and <code>inherit.aes = TRUE</code> (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.
---------	--

<code>data</code>	A data frame. If specified, overrides the default data frame defined at the top level of the plot.
<code>prop_below</code>	the degree to shift the values. The default is 0.1
<code>jitter</code>	the amount of jitter to add. The default is 0.05
<code>stat</code>	The statistical transformation to use on the data for this layer, as a string.
<code>position</code>	Position adjustment, either as a string, or the result of a call to a position adjustment function.
<code>colour</code>	the colour chosen for the aesthetic
<code>na.rm</code>	If FALSE (the default), removes missing values with a warning. If TRUE silently removes missing values.
<code>show.legend</code>	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.
<code>inherit.aes</code>	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.
<code>...</code>	other arguments passed on to <code>ggplot2::layer()</code> . There are three types of arguments you can use here: <ul style="list-style-type: none"> <li>• Aesthetics: to set an aesthetic to a fixed value, like <code>color = "red"</code> or <code>size = 3</code>.</li> <li>• Other arguments to the layer, for example you override the default <code>stat</code> associated with the layer.</li> <li>• Other arguments passed on to the <code>stat</code>.</li> </ul>

## 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_`

## 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

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

### Arguments

<code>x</code>	data.frame
<code>facet</code>	(optional) a single bare variable name, if you want to create a faceted plot.
<code>order_cases</code>	logical Order the rows by missingness (default is FALSE - no order).
<code>show_pct</code>	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

```
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

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

## Arguments

<code>x</code>	a dataframe
<code>breaks</code>	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

```
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

<code>x</code>	data.frame
<code>fct</code>	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()  
gg_miss_var_cumsum() gg_miss_which()
```

## 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

<code>data</code>	<code>data.frame</code>
<code>var</code>	a bare unquoted variable name from <code>data</code> .
<code>span_every</code>	integer describing the length of the span to be explored
<code>facet</code>	(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)
```

---

`gg_miss_upset`*Plot the pattern of missingness using an upset plot.*

---

## 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 be 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

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

## Arguments

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

## Value

a ggplot visualisation of missing data

## Examples

```
## 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

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

## Arguments

<code>x</code>	a dataframe
<code>facet</code>	(optional) bare variable name, if you want to create a faceted plot.
<code>show_pct</code>	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

```
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)
```

---

group\_by\_fun

*Group By Helper*

---

**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:
miss_case_table.grouped_df <- function(data){
  group_by_fun(data, .fun = miss_case_table)
}
airquality %>%
  group_by(Month) %>%
  miss_case_table()

## End(Not run)
```

---

impute_below	<i>Impute data with values shifted 10 percent below range.</i>
--------------	--

---

### Description

It can be useful in exploratory graphics to impute data outside the range of the data. `impute_below` 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

```
impute_below(...)
```

### Arguments

... extra arguments to pass - see `shadow_shift()` for discussion on this.

---

impute_below_all	<i>Impute data with values shifted 10 percent below range.</i>
------------------	--

---

### 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\ values adds a new string or label.

### Usage

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

### Arguments

<code>.tbl</code>	a data.frame
<code>prop_below</code>	the degree to shift the values. default is
<code>jitter</code>	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() %>%
  # identify where there are missings across rows.
  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	<i>Scoped variants of impute_below</i>
-----------------	--

---

**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, ...)
```

**Arguments**

<code>.tbl</code>	data.frame
<code>.predicate</code>	A predicate function (such as <code>is.numeric</code> )
<code>prop_below</code>	the degree to shift the values. default is
<code>jitter</code>	the amount of jitter to add. default is 0.05
<code>...</code>	extra arguments

**Value**

an dataset with values imputed

**Examples**

```
airquality %>%
  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**

```
impute_mean(x)

## Default S3 method:
impute_mean(x)

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

**Arguments**

<code>x</code>	vector
----------------	--------

**Value**

vector with mean values replaced

**Examples**

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

---

impute_median	<i>Impute the median value into a vector with missing values</i>
---------------	--

---

**Description**

Impute the median value into a vector with missing values

**Usage**

```
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**

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

---

is_shade	<i>Detect if this is a shade</i>
----------	----------------------------------

---

**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	<i>Is there a missing value in the row of a dataframe?</i>
----------------	--

---

**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 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	<i>Label a missing from one column</i>
---------------	--

---

**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	<i>label_miss_2d</i>
---------------	----------------------

---

**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	<i>Label shadow values as missing or not missing</i>
--------------	--

---

**Description**

Powers add\_label\_shadow. For the moment it is an internal function.

**Usage**

```
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 character a label for when values are complete - defaults to "Not Missing"

**Value**

"Missing" or "Not Missing"



---

`mcar_test`*Little's missing completely at random (MCAR) test*

---

### Description

Use Little's (1988) test statistic to assess if data is missing completely at random (MCAR). The null hypothesis in this test is that the data is MCAR, and the test statistic is a chi-squared value. The example below shows the output of `mcar_test(airquality)`. Given the high statistic value and low p-value, we can conclude the `airquality` data is not missing completely at random.

### Usage

```
mcar_test(data)
```

### Arguments

`data`            A data frame

### Value

A `tibble::tibble()` with one row and four columns:

<code>statistic</code>	Chi-squared statistic for Little's test
<code>df</code>	Degrees of freedom used for chi-squared statistic
<code>p.value</code>	P-value for the chi-squared statistic
<code>missing.patterns</code>	Number of missing data patterns in the data

### Note

Code is adapted from `LittleMCAR()` in the now-orphaned `BaylorEdPsych` package: <https://rdrr.io/cran/BaylorEdPsych/man/LittleMCAR.html>.  
Some of code is adapted from Eric Stemmler - <https://web.archive.org/web/20201120030409/https://stats-bayes.com/post/2020/08/14/r-function-for-little-s-test-for-data-missing-completely-at-random/> using Maximum likelihood estimation from norm.

### Author(s)

Andrew Heiss, <[andrew@andrewheiss.com](mailto:andrew@andrewheiss.com)>

### References

Little, Roderick J. A. 1988. "A Test of Missing Completely at Random for Multivariate Data with Missing Values." *Journal of the American Statistical Association* 83 (404): 1198–1202. doi: [10.1080/01621459.1988.10478722](https://doi.org/10.1080/01621459.1988.10478722).

## Examples

```
mcars_test(airquality)
mcars_test(oceanbuoys)

# If there are non-numeric columns, there will be a warning
mcars_test(riskfactors)
```

---

miss-pct-prop-defunct *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

```
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

```
...           arguments
```

---

miss_case_cumsum	<i>Summarise the missingness in each case</i>
------------------	---

---

**Description**

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

**Usage**

```
miss_case_cumsum(data)
```

**Arguments**

data            a dataframe

**Value**

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

**Examples**

```
miss_case_cumsum(airquality)

## Not run:
library(dplyr)

airquality %>%
  group_by(Month) %>%
  miss_case_cumsum()

## End(Not run)
```

---

miss_case_summary	<i>Summarise the missingness in each case</i>
-------------------	---

---

**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.

**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_case_summary(airquality)

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

## End(Not run)
```

---

miss_case_table	<i>Tabulate missings in cases.</i>
-----------------	------------------------------------

---

**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 dataframe

**Value**

a dataframe

**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_case_table(airquality)
## Not run:
library(dplyr)
airquality %>%
  group_by(Month) %>%
  miss_case_table()

## End(Not run)
```

---

miss\_prop\_summary            *Proportions of missings in data, variables, and cases.*

---

**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**

```
miss_prop_summary(data)
```

**Arguments**

data            a dataframe

**Value**

a dataframe

**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_run() miss_var_span() miss_var_summary()
miss_var_table()
```

**Examples**

```
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**

```
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	<i>Collate summary measures from naniar into one tibble</i>
--------------	---

---

**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\(\)](#)  
[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**

```

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

## Not run:
library(dplyr)
s_miss_group <- group_by(airquality, Month) %>% miss_summary()
s_miss_group$miss_df_prop
s_miss_group$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\(\)](#)  
[miss\\_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**

```
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\\_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_run(pedestrian, hourly_counts)

## 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) %>%
  filter(is_na == "missing") %>%
  count(run_length) %>%
  ggplot(aes(x = run_length,
            y = n)) +
  geom_col()

# look at the number of missing values and the run length of these.
miss_var_run(pedestrian, hourly_counts) %>%
  ggplot(aes(x = is_na,
            y = run_length)) +
  geom_boxplot()

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

## End(Not run)

```

---

miss_var_span	<i>Summarise the number of missings for a given repeating span on a variable</i>
---------------	--

---

**Description**

To summarise the missing values in a time series object it can be useful to calculate the number of missing values in a given time 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. When the number of observations isn't a perfect multiple of the span length, the final span is whatever the last remainder is. For example, the `pedestrian` dataset has 37,700 rows. If the span is set to 4000, then there will be 1700 rows remaining. This can be provided using modulo (%): `nrow(data) %% 4000`. This remainder number is provided in `n_in_span`.

**Usage**

```
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. The final variable, n\_in\_span states how many observations are in the span.

**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**

```
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)
```

---

miss_var_summary	<i>Summarise the missingness in each variable</i>
------------------	---

---

**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

**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)
```

---

miss_var_table	<i>Tabulate the missings in the variables</i>
----------------	---

---

### 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

### 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_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**

```
n_var_complete(data)
```

```
n_case_complete(data)
```

**Arguments**

`data`            data.frame

**Value**

integer number of complete values

**See Also**

[n\\_var\\_miss\(\)](#)

**Examples**

```
# 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**

```
n_var_miss(data)
```

```
n_case_miss(data)
```

**Arguments**

data            data.frame

**Value**

integer, number of missings

**See Also**

[n\\_var\\_complete\(\)](#)

**Examples**

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

---

`nabular`*Convert data into nabular form by binding shade to it*

---

### 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

<code>data</code>	a dataframe
<code>only_miss</code>	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.
<code>...</code>	extra options to pass to <a href="#">recode_shadow()</a> - a work in progress.

### Value

data with the added variable shifted and the suffix `_NA`

### See Also

[bind\\_shadow\(\)](#)

### Examples

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

all.equal(aq_nab, aq_s)
```



---

naniar	<i>naniar</i>
--------	---------------

---

## 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\(\)](#) [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\(\)](#) [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	<i>Create a new shade factor</i>
-----------	----------------------------------

---

## Description

Create a new shade factor

## Usage

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

## Arguments

x	a factor to convert into a shade object
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

---

n_complete	<i>Return the number of complete values</i>
------------	---

---

**Description**

A complement to n\_miss

**Usage**

```
n_complete(x)
```

**Arguments**

x                    a vector

**Value**

numeric number of complete values

**Examples**

```
n_complete(airquality)
n_complete(airquality$Ozone)
```

---

n_complete_row	<i>Return a vector of the number of complete values in each row</i>
----------------	---

---

**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()`  
`pct_miss()` `prop_complete()` `prop_complete_row()` `prop_miss()`

**Examples**

```
n_complete_row(airquality)
```

---

n_miss	<i>Return the number of missing values</i>
--------	--

---

**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	<i>Return a vector of the number of missing values in each row</i>
------------	--

---

**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)
```

---

oceanbuoys	<i>West Pacific Tropical Atmosphere Ocean Data, 1993 &amp; 1997.</i>
------------	--

---

**Description**

Real-time data from moored ocean buoys for improved detection, understanding and prediction of El Ni'no 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

## 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

```
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()  
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)
```

---

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  $\text{mean}(\text{is.na}(x)) * 100$

**Usage**

```
pct_miss(x)
```

**Arguments**

x                    vector or data.frame

**Value**

numeric the percent of missing values in x

**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

<https://data.melbourne.vic.gov.au/Transport-Movement/Pedestrian-volume-updated-monthly-b2ak-trbp>

## 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.GeoMissPoint(data, prestats_data, layout, params, p, ...)
```

### Arguments

data	the data returned by <code>ggplot2::ggplot_build()</code> .
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\(\)](#)  
[miss\\_case\\_table\(\)](#) [miss\\_summary\(\)](#) [miss\\_var\\_prop\(\)](#) [miss\\_var\\_run\(\)](#) [miss\\_var\\_span\(\)](#) [miss\\_var\\_summary\(\)](#)  
[miss\\_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	<i>Return the proportion of complete values</i>
---------------	---

---

**Description**

The complement to prop\_miss

**Usage**

```
prop_complete(x)
```

**Arguments**

x                    vector or data.frame

**Value**

numeric proportion of complete values

**Examples**

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

---

prop_complete_row	<i>Return a vector of the proportion of missing values in each row</i>
-------------------	--

---

**Description**

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

**Usage**

```
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\(\)](#)  
[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**

```
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	<i>Return a vector of the proportion of missing values in each row</i>
---------------	--

---

**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**

```
prop_miss_row(airquality)
```

---

recode_shadow	<i>Add special missing values to the shadow matrix</i>
---------------	--

---

**Description**

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

**Usage**

```
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**

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

dfs <- bind_shadow(df)

dfs

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

recode_shadow(dfs,
              temp = .where(wind == -99 ~ "bananas")) %>%
recode_shadow(wind = .where(wind == -99 ~ "apples"))
```

---

replace_to_na	<i>Replace values with missings</i>
---------------	-------------------------------------

---

**Description**

This function is Defunct, please see [replace\\_with\\_na\(\)](#).

**Usage**

```
replace_to_na(...)
```

**Arguments**

...            additional arguments for methods.

**Value**

values replaced by NA

---

replace_with_na	<i>Replace values with missings</i>
-----------------	-------------------------------------

---

### Description

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

### Usage

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

### Arguments

<code>data</code>	A data.frame
<code>replace</code>	A named list given the NA to replace values for each column
<code>...</code>	additional arguments for methods. Currently unused

### 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: <code>~.x {condition}</code> . For example, writing <code>~.x &lt; 20</code> 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: <code>~.x {condition}</code> . For example, writing <code>~.x &lt; 20</code> would mean "where a variable value is less than 20, replace with NA".

### Value

a 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_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	<i>Replace values with NA based on some condition, for variables that meet some predicate</i>
--------------------	---

---

### 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: <code>~.x {condition}</code> . For example, writing <code>~.x &lt; 20</code> 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)))
```

---

riskfactors	<i>The Behavioral Risk Factor Surveillance System (BRFSS) Survey Data, 2009.</i>
-------------	--

---

**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](https://www.cdc.gov/brfss/annual_data/annual_2009.htm)

**See Also**

the codebook: [https://www.cdc.gov/brfss/annual\\_data/annual\\_2009.htm](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,  
 11:District of Columbia,12:Florida, 13:Georgia, 15:Hawaii, 16:Idaho, 1 :Illinois,18:Indiana,  
 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,  
 33:New Hampshire, 34:New Jersey, 35:NewMexico, 36:New York, 37:North Carolina, 38:North  
 Dakota, 39:Ohio,40:Oklahoma, 41:Oregon, 42:Pennsylvania, 44:Rhode Island, 45:SouthCarolina,  
 46:South Dakota, 47:Tennessee, 48:Texas, 49:Utah, 50:Vermont, 51:Virginia, 53:Washington,  
 54:West Virginia,55:Wisconsin, 56:Wyoming, 66:Guam, 72:Puerto Rico, 78:Virgin Islands

`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 Dontknow 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, pre-paid 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

```
vis_miss(riskfactors)

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

# and now as a plot
gg_miss_var(riskfactors)

## 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)
```

```
## 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

## Usage

```
impute_mean_all(.tbl)

impute_mean_at(.tbl, .vars)

impute_mean_if(.tbl, .predicate)
```

## Arguments

<code>.tbl</code>	a data.frame
<code>.vars</code>	variables to impute
<code>.predicate</code>	variables to impute

## Value

an dataset with values imputed

## Examples

```
# 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

## Usage

```

impute_median_all(.tbl)

impute_median_at(.tbl, .vars)

impute_median_if(.tbl, .predicate)

```

## Arguments

<code>.tbl</code>	a data.frame
<code>.vars</code>	variables to impute
<code>.predicate</code>	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() %>%
  ggplot(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

```
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

```
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

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

### 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	<i>Reshape shadow data into a long format</i>
-------------	---

---

### 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?

### Value

data in long format, with columns variable, value, variable\_NA, and value\_NA.

### Examples

```
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	<i>Shift missing values to facilitate missing data exploration/visualisation</i>
--------------	--

---

### 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**

```
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**

```
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**

```
## 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	<i>stat_miss_point</i>
-----------------	------------------------

---

## Description

stat\_miss\_point adds a geometry for displaying missingness to geom\_point

## Usage

```
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 <code>ggplot2::aes()</code> or <code>ggplot2::aes_()</code> . If specified and <code>inherit.aes = TRUE</code> (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 <code>geom_point</code> and <code>stat_point</code> .
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. <code>borders</code> .
...	other arguments passed on to <code>ggplot2::layer()</code> . 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	<i>Test if input is a data.frame</i>
-------------------	--------------------------------------

---

### 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	<i>Test if the input is Missing</i>
-----------------	-------------------------------------

---

**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	<i>Test if the input is NULL</i>
--------------	----------------------------------

---

**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**

```
# 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	<i>Expand all shadow levels</i>
---------------	---------------------------------

---

### Description

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

### Usage

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

### Arguments

data	data.frame
suffix	character vector

### Value

data.frame with adjusted levels

### Examples

```
## 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	<i>check the levels of many things</i>
-------------	--

---

**Description**

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

**Usage**

```
what_levels(x)
```

**Arguments**

x                    data.frame, usually

**Value**

a list containing the levels of everything

---

where	<i>Split a call into two components with a useful verb name</i>
-------	---

---

**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

**Usage**

```
.where(...)
```

**Arguments**

...                    case\_when style formula

**Value**

a list of "condition" and "suffix" arguments

## Examples

```
## 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)
```

---

where\_na

*Which rows and cols contain missings?*

---

## Description

Internal function that is short for `which(is.na(x), arr.ind = TRUE)`. Creates array index locations of missing values in a dataframe.

## Usage

```
where_na(x)
```

## Arguments

x                    a dataframe

## 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)
```

---

which_are_shade	<i>Which variables are shades?</i>
-----------------	------------------------------------

---

**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	<i>Which elements contain missings?</i>
----------	---

---

**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.

`which_na`

93

**See Also**

[where\\_na\(\)](#)

**Examples**

```
which_na(airquality)
```

# Index

- \* **datasets**
  - common\_na\_numbers, 19
  - common\_na\_strings, 20
  - GeomMissPoint, 22
  - oceanbuoys, 60
  - pedestrian, 65
  - riskfactors, 76
- .where (where), 90
  
- add\_any\_miss, 4
- add\_any\_miss(), 5–11, 17–19, 38, 39, 57
- add\_label\_missings, 6
- add\_label\_missings(), 5–11, 17–19, 38, 39, 57
- add\_label\_shadow, 7
- add\_label\_shadow(), 5–11, 17–19, 38, 39, 57
- add\_miss\_cluster, 8
- add\_miss\_cluster(), 5–11, 17–19, 38, 57
- add\_n\_miss, 8
- add\_n\_miss(), 5–8, 11, 38, 57
- add\_prop\_miss, 9
- add\_prop\_miss(), 5–11, 17–19, 38, 57
- add\_shadow, 10
- add\_shadow(), 57
- add\_shadow\_shift, 11
- add\_shadow\_shift(), 5–11, 17–19, 38, 57, 84
  
- 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(), 57
- as\_shadow\_upset, 15
  
- bind\_shadow, 16
- bind\_shadow(), 5–11, 17–19, 38, 56, 57, 88
  
- cast\_shadow, 17
- cast\_shadow(), 5–11, 38, 57
- cast\_shadow\_shift, 18
- cast\_shadow\_shift(), 17–19, 57, 84
- cast\_shadow\_shift\_label, 18
- cast\_shadow\_shift\_label(), 17–19, 57, 84
- common\_na\_numbers, 19
- common\_na\_strings, 20
- complete\_case\_pct
  - (miss-pct-prop-defunct), 42
- complete\_case\_prop
  - (miss-pct-prop-defunct), 42
- complete\_var\_pct
  - (miss-pct-prop-defunct), 42
- complete\_var\_prop
  - (miss-pct-prop-defunct), 42
  
- draw\_key\_missing\_point(), 57
  
- gather\_shadow, 21
- gather\_shadow(), 57
- geom\_miss\_point, 22
- geom\_miss\_point(), 24–27, 29–31, 57
- GeomMissPoint, 22
- gg\_miss\_case, 24
- gg\_miss\_case(), 25–27, 29–31, 57
- gg\_miss\_case\_cumsum, 24, 25, 26, 27, 29–31
- gg\_miss\_case\_cumsum(), 57
- gg\_miss\_fct, 26
- gg\_miss\_fct(), 24, 25, 27, 29–31, 57
- gg\_miss\_span, 27

- gg\_miss\_span(), 24–26, 29–31, 57
- gg\_miss\_upset, 28
- gg\_miss\_var, 29
- gg\_miss\_var(), 24–27, 29–31, 57
- gg\_miss\_var\_cumsum, 30
- gg\_miss\_var\_cumsum(), 24–27, 29, 31, 57
- gg\_miss\_which, 30
- gg\_miss\_which(), 24–27, 29–31, 57
- ggplot2::aes(), 22, 85
- ggplot2::aes\_(), 22, 85
- ggplot2::layer(), 23, 85
- group\_by\_fun, 31
  
- impute\_below, 32
- impute\_below\_all, 32
- impute\_below\_at, 33
- impute\_below\_if, 34
- impute\_mean, 35
- impute\_mean\_all (scoped-impute\_mean), 79
- impute\_mean\_at (scoped-impute\_mean), 79
- impute\_mean\_if (scoped-impute\_mean), 79
- impute\_median, 36
- impute\_median\_all
  - (scoped-impute\_median), 80
- impute\_median\_at
  - (scoped-impute\_median), 80
- impute\_median\_if
  - (scoped-impute\_median), 80
- is\_shade, 37
  
- label\_miss\_1d, 39
- label\_miss\_1d(), 57
- label\_miss\_2d, 39
- label\_miss\_2d(), 57
- label\_missings, 38
- label\_missings(), 57
- label\_shadow, 40
  
- mcar\_test, 41
- miss-pct-prop-defunct, 42
- miss\_case\_cumsum, 43
- miss\_case\_pct (miss-pct-prop-defunct), 42
- miss\_case\_prop (miss-pct-prop-defunct), 42
- miss\_case\_summary, 43
- miss\_case\_summary(), 44–49, 51–53, 57, 59, 60, 62, 63, 67, 69, 70
- miss\_case\_table, 44
- miss\_case\_table(), 44–49, 51–53, 57, 59, 60, 62, 63, 67, 69, 70
- miss\_prop\_summary, 45
- miss\_prop\_summary(), 44–49, 51–53, 57, 59, 60, 62, 63, 67, 69, 70
- miss\_scan\_count, 46
- miss\_scan\_count(), 19, 20
- miss\_summary, 47
- miss\_summary(), 44–49, 51–53, 57, 59, 60, 62, 63, 67, 69, 70
- miss\_var\_cumsum, 48
- miss\_var\_pct (miss-pct-prop-defunct), 42
- miss\_var\_prop (miss-pct-prop-defunct), 42
- miss\_var\_prop(), 44–49, 51–53, 57, 59, 60, 62, 63, 67, 69, 70
- miss\_var\_run, 49
- miss\_var\_run(), 44–49, 51–53, 57, 59, 60, 62, 63, 67, 69, 70
- miss\_var\_span, 50
- miss\_var\_span(), 44–49, 51–53, 57, 59, 60, 62, 63, 67, 69, 70
- miss\_var\_summary, 51
- miss\_var\_summary(), 44–49, 51–53, 57, 59, 60, 62, 63, 67, 69, 70
- miss\_var\_table, 53
- miss\_var\_table(), 44–49, 51–53, 57, 59, 60, 62, 63, 67, 69, 70
- miss\_var\_which, 54
  
- n-var-case-complete, 54
- n-var-case-miss, 55
- n\_case\_complete (n-var-case-complete), 54
- n\_case\_miss (n-var-case-miss), 55
- n\_complete, 58
- n\_complete(), 44, 45, 47, 49, 52, 53, 57, 59, 60, 69, 70
- n\_complete\_row, 58
- n\_complete\_row(), 44, 45, 47, 49, 52, 53, 57, 59, 60, 69, 70
- n\_miss, 59
- n\_miss(), 44, 45, 47, 49, 52, 53, 57, 59, 60, 69, 70
- n\_miss\_row, 60
- n\_miss\_row(), 44, 45, 47, 49, 52, 53, 57, 59, 60, 69, 70
- n\_var\_complete (n-var-case-complete), 54
- n\_var\_complete(), 55

- n\_var\_miss (n-var-case-miss), 55
- n\_var\_miss(), 55
- nabular, 56
- naniar, 57
- naniar-ggproto (GeomMissPoint), 22
- new\_shade, 57
  
- oceanbuoys, 60
  
- pct-miss-complete-case, 62
- pct-miss-complete-var, 63
- pct\_complete, 63
- pct\_complete(), 44, 45, 47, 49, 52, 53, 57, 59, 60, 69, 70
- pct\_complete\_case
  - (pct-miss-complete-case), 62
- pct\_complete\_case(), 42, 44–49, 51–53, 57, 59, 60, 62, 63, 67, 69, 70
- pct\_complete\_var
  - (pct-miss-complete-var), 63
- pct\_complete\_var(), 42, 44–49, 51–53, 57, 59, 60, 62, 63, 67, 69, 70
- pct\_miss, 64
- pct\_miss(), 44, 45, 47, 49, 52, 53, 57, 59, 60, 69, 70
- pct\_miss\_case (pct-miss-complete-case), 62
- pct\_miss\_case(), 42, 44–49, 51–53, 57, 59, 60, 62, 63, 67, 69, 70
- pct\_miss\_var (pct-miss-complete-var), 63
- pct\_miss\_var(), 42, 44–49, 51–53, 57, 59, 60, 62, 63, 67, 69, 70
- pedestrian, 65
- plotly\_helpers, 66
- prop-miss-complete-case, 66
- prop-miss-complete-var, 67
- prop\_complete, 68
- prop\_complete(), 44, 45, 47, 49, 52, 53, 57, 59, 60, 69, 70
- prop\_complete\_case
  - (prop-miss-complete-case), 66
- prop\_complete\_case(), 42, 44–49, 51–53, 57, 59, 60, 62, 63, 67, 69, 70
- prop\_complete\_row, 68
- prop\_complete\_row(), 44, 45, 47, 49, 52, 53, 57, 59, 60, 69, 70
- prop\_complete\_var
  - (prop-miss-complete-var), 67
- prop\_complete\_var(), 42, 44–49, 51–53, 57, 59, 60, 62, 63, 67, 69, 70
- prop\_miss, 69
- prop\_miss(), 44, 45, 47, 49, 52, 53, 57, 59, 60, 69, 70
- prop\_miss\_case
  - (prop-miss-complete-case), 66
- prop\_miss\_case(), 42, 44–49, 51–53, 57, 59, 60, 62, 63, 67, 69, 70
- prop\_miss\_row, 70
- prop\_miss\_row(), 57
- prop\_miss\_var (prop-miss-complete-var), 67
- prop\_miss\_var(), 42, 44–49, 51–53, 57, 59, 60, 62, 63, 67, 69, 70
  
- recode\_shadow, 70
- recode\_shadow(), 16, 56
- replace\_to\_na, 71
- replace\_to\_na(), 57
- replace\_with\_na, 72
- replace\_with\_na(), 19, 20, 57, 71, 72
- replace\_with\_na\_all, 73
- replace\_with\_na\_all(), 57, 72
- replace\_with\_na\_at, 74
- replace\_with\_na\_at(), 57, 72
- replace\_with\_na\_if, 75
- replace\_with\_na\_if(), 57, 72
- riskfactors, 76
  
- scoped-impute\_mean, 79
- scoped-impute\_median, 80
- shade, 81
- shadow\_expand\_relevel, 82
- shadow\_long, 83
- shadow\_shift, 83
- shadow\_shift(), 32, 57
- shadow\_shift.numeric, 84
- stat\_miss\_point, 85
- stat\_miss\_point(), 57
- StatMissPoint (GeomMissPoint), 22
  
- test\_if\_dataframe, 86
- test\_if\_missing, 87
- test\_if\_null, 87
- tibble::tibble(), 41
- to\_basic.GeoMissPoint
  - (plotly\_helpers), 66
- unbind\_data (unbinders), 88



`unbind_data()`, [88](#)  
`unbind_shadow (unbinders)`, [88](#)  
`unbind_shadow()`, [88](#)  
`unbinders`, [88](#)  
`update_shadow`, [89](#)

`vis_miss()`, [57](#)

`what_levels`, [90](#)  
`where`, [90](#)  
`where_na`, [91](#)  
`where_na()`, [57](#), [93](#)  
`which_are_shade`, [92](#)  
`which_na`, [92](#)  
`which_na()`, [91](#)