

# Package: visdat (via r-universe)

September 26, 2026

**Title** Preliminary Visualisation of Data

**Version** 0.6.1.9000

**Description** Create preliminary exploratory data visualisations of an entire dataset to identify problems or unexpected features using 'ggplot2'.

**Depends** R (>= 4.2.0)

**License** MIT + file LICENSE

**LazyData** true

**Imports** ggplot2, tidyr, dplyr, purrr, readr, stats, tibble, glue, forcats, cli, scales, rlang

**URL** <https://docs.ropensci.org/visdat/>,  
<https://github.com/ropensci/visdat>

**BugReports** <https://github.com/ropensci/visdat/issues>

**Suggests** testthat (>= 3.0.0), plotly (>= 4.5.6), knitr, rmarkdown, vdiffr, spelling, covr, stringr

**VignetteBuilder** knitr

**Encoding** UTF-8

**Roxygen** list(markdown = TRUE)

**Language** en-GB

**Config/testthat/edition** 3

**Config/roxygen2/version** 8.0.0

**Config/pak/sysreqs** libicu-dev libx11-dev

**Repository** <https://njtierney.r-universe.dev>

**Date/Publication** 2026-09-26 04:38:36 UTC

**RemoteUrl** <https://github.com/ropensci/visdat>

**RemoteRef** HEAD

**RemoteSha** 9b3006a52a98843dcf2bbff91f27585bb52c000a

Contents

|                              |           |
|------------------------------|-----------|
| abbreviate_vars . . . . .    | 2         |
| dat_bin . . . . .            | 3         |
| data-vis-cor . . . . .       | 3         |
| data-vis-dat . . . . .       | 5         |
| data-vis-miss . . . . .      | 6         |
| typical_data . . . . .       | 7         |
| typical_data_large . . . . . | 7         |
| vis_binary . . . . .         | 9         |
| vis_compare . . . . .        | 10        |
| vis_cor . . . . .            | 11        |
| vis_dat . . . . .            | 12        |
| vis_expect . . . . .         | 13        |
| vis_guess . . . . .          | 15        |
| vis_histogram . . . . .      | 16        |
| vis_miss . . . . .           | 16        |
| vis_value . . . . .          | 18        |
| <b>Index</b>                 | <b>20</b> |

---

|                 |                                                 |
|-----------------|-------------------------------------------------|
| abbreviate_vars | <i>Abbreviate all variables in a data frame</i> |
|-----------------|-------------------------------------------------|

---

Description

It can be useful to abbreviate variable names in a data set to make them easier to plot. This function takes in a data set and some minimum length to abbreviate the data to.

Usage

```
abbreviate_vars(data, min_length = 10)
```

Arguments

|            |                                                    |
|------------|----------------------------------------------------|
| data       | data.frame                                         |
| min_length | minimum number of characters to abbreviate down to |

Value

data frame with abbreviated variable names

**Examples**

```

long_data <- data.frame(
  really_really_long_name = c(NA, NA, 1:8),
  very_quite_long_name = c(-1:-8, NA, NA),
  this_long_name_is_something_else = c(NA, NA,
                                         seq(from = 0, to = 1, length.out = 8))
)

vis_miss(long_data)
long_data |> abbreviate_vars() |> vis_miss()

```

dat\_bin

*A small toy dataset of binary data with missings.***Description**

A dataset containing binary values and missing values. It is created to illustrate the usage of [vis\\_binary\(\)](#).

**Usage**

```
dat_bin
```

**Format**

A data frame with 100 rows and 3 variables:

**x** a binary variable with missing values.

**y** a binary variable with missing values.

**z** a binary variable with **no** missing values.

data-vis-cor

*Return data used to create vis\_cor plot***Description**

Return data used to create vis\_cor plot

Create a tidy dataframe of correlations suitable for plotting

**Usage**

```
data_vis_cor(x, ...)

## Default S3 method:
data_vis_cor(x, ...)

## S3 method for class 'data.frame'
data_vis_cor(
  x,
  cor_method = "pearson",
  na_action = "pairwise.complete.obs",
  ...
)

## S3 method for class 'grouped_df'
data_vis_cor(x, ...)
```

**Arguments**

|            |                                                                                                                                                                                                                                                   |
|------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| x          | data.frame                                                                                                                                                                                                                                        |
| ...        | extra arguments (currently unused)                                                                                                                                                                                                                |
| cor_method | correlation method to use, from cor: "a character string indicating which correlation coefficient (or covariance) is to be computed. One of "pearson" (default), "kendall", or "spearman": can be abbreviated."                                   |
| na_action  | The method for computing covariances when there are missing values present. This can be "everything", "all.obs", "complete.obs", "na.or.complete", or "pairwise.complete.obs" (default). This option is taken from the cor function argument use. |

**Value**

data frame  
tidy dataframe of correlations

**Examples**

```
data_vis_cor(airquality)

## Not run:
#return vis_dat data for each group
library(dplyr)
airquality |>
  group_by(Month) |>
  data_vis_cor()

## End(Not run)
data_vis_cor(airquality)
```

---

|              |                                                |
|--------------|------------------------------------------------|
| data-vis-dat | <i>Return data used to create vis_dat plot</i> |
|--------------|------------------------------------------------|

---

## Description

Return data used to create vis\_dat plot

## Usage

```
data_vis_dat(x, ...)  
  
## Default S3 method:  
data_vis_dat(x, ...)  
  
## S3 method for class 'data.frame'  
data_vis_dat(x, ...)  
  
## S3 method for class 'grouped_df'  
data_vis_dat(x, ...)
```

## Arguments

|     |                                    |
|-----|------------------------------------|
| x   | data.frame                         |
| ... | extra arguments (currently unused) |

## Value

data frame

## Examples

```
data_vis_dat(airquality)  
  
## Not run:  
#return vis_dat data for each group  
library(dplyr)  
airquality |>  
  group_by(Month) |>  
  data_vis_dat()  
  
## End(Not run)
```

---

|               |                                                 |
|---------------|-------------------------------------------------|
| data-vis-miss | <i>Return data used to create vis_miss plot</i> |
|---------------|-------------------------------------------------|

---

### Description

Return data used to create vis\_miss plot

Create a tidy dataframe of missing data suitable for plotting

### Usage

```
data_vis_miss(x, ...)  
  
## Default S3 method:  
data_vis_miss(x, ...)  
  
## S3 method for class 'data.frame'  
data_vis_miss(x, cluster = FALSE, ...)  
  
## S3 method for class 'grouped_df'  
data_vis_miss(x, ...)
```

### Arguments

|         |                                                             |
|---------|-------------------------------------------------------------|
| x       | data.frame                                                  |
| ...     | extra arguments (currently unused)                          |
| cluster | logical - whether to cluster missingness. Default is FALSE. |

### Value

data frame  
tidy dataframe of missing data

### Examples

```
data_vis_miss(airquality)  
  
## Not run:  
#return vis_dat data for each group  
library(dplyr)  
airquality |>  
  group_by(Month) |>  
  data_vis_miss()  
  
## End(Not run)  
data_vis_miss(airquality)
```

---

|              |                                                |
|--------------|------------------------------------------------|
| typical_data | <i>A small toy dataset of imaginary people</i> |
|--------------|------------------------------------------------|

---

**Description**

A dataset containing information about some randomly generated people, created using the excellent wakefield package. It is created as deliberately messy dataset.

**Usage**

```
typical_data
```

**Format**

A data frame with 5000 rows and 11 variables:

**ID** Unique identifier for each individual, a sequential character vector of zero-padded identification numbers (IDs). see ?wakefield::id

**Race** Race for each individual, "Black", "White", "Hispanic", "Asian", "Other", "Bi-Racial", "Native", and "Hawaiin", see ?wakefield::race

**Age** Age of each individual, see ?wakefield::age

**Sex** Male or female, see ?wakefield::sex

**Height(cm)** Height in centimeters, see ?wakefield::height

**IQ** vector of intelligence quotients (IQ), see ?wakefield::iq

**Smokes** whether or not this person smokes, see ?wakefield::smokes

**Income** Yearly income in dollars, see ?wakefield::income

**Died** Whether or not this person has died yet., see ?wakefield::died

---

|                    |                                                |
|--------------------|------------------------------------------------|
| typical_data_large | <i>A small toy dataset of imaginary people</i> |
|--------------------|------------------------------------------------|

---

**Description**

A wider dataset than typical\_data containing information about some randomly generated people, created using the excellent wakefield package. It is created as deliberately odd / eclectic dataset.

**Usage**

```
typical_data_large
```

**Format**

A data frame with 300 rows and 49 variables:

**Age** Age of each individual, see ?wakefield::age for more info

**Animal** A vector of animals, see ?wakefield::animal

**Answer** A vector of "Yes" or "No"

**Area** A vector of living areas "Suburban", "Urban", "Rural"

**Car** names of cars - see ?mtcars

**Children** vector of number of children - see ?wakefield::children

**Coin** character vector of "heads" and "tails"

**Color** vector of vectors from "colors()"

**Date** vector of "important" dates for an individual

**Death** TRUE / FALSE for whether this person died

**Dice** 6 sided dice result

**DNA** vector of GATC nucleobases

**DOB** birth dates

**Dummy** a 0/1 dummy var

**Education** education attainment level

**Employment** employee status

**Eye** eye colour

**Grade** percent grades

**Grade\_Level** favourite school grade

**Group** control or treatment

**hair** hair colours - "brown", "black", "blonde", or "red"

**Height** height in cm

**Income** yearly income

**Browser** choice of internet browser

**IQ** intelligence quotient

**Language** random language of the world

**Level** levels between 1 and 4

**Likert** likert response - "strongly agree", "agree", and so on

**Lorem\_Ipsum** lorem ipsum text

**Marital** marital status- "married", "divorced", "widowed", "separated", etc

**Military** military branch they are in

**Month** their favourite month

**Name** their name

**Normal** a random normal number

**Political** their favourite political party

**Race** their race  
**Religion** their religion  
**SAT** their SAT score  
**Sentence** an uttered sentence  
**Sex\_1** sex of their first child  
**Sex\_2** sex of their second child  
**Smokes** do they smoke  
**Speed** their median speed travelled in a car  
**State** the last state they visited in the USA  
**String** a random string they smashed out on the keyboard  
**Upper** the last key they hit in upper case  
**Valid** TRUE FALSE answer to a question  
**Year** significant year to that individuals  
**Zip** a zip code they have visited

---

vis\_binary

*Visualise binary values*

---

## Description

Visualise binary values

## Usage

```
vis_binary(  
  data,  
  col_zero = "salmon",  
  col_one = "steelblue2",  
  col_na = "grey90",  
  order = NULL  
)
```

## Arguments

|          |                                                     |
|----------|-----------------------------------------------------|
| data     | a data.frame                                        |
| col_zero | colour for zeroes, default is "salmon"              |
| col_one  | colour for ones, default is "steelblue2"            |
| col_na   | colour for NA, default is "grey90"                  |
| order    | optional character vector of the order of variables |

## Value

a ggplot plot of the binary values

**Examples**

```
vis_binary(dat_bin)

# changing order of variables
# create numeric names
df <- setNames(dat_bin, c("1.1", "8.9", "10.4"))
df

# not ideal
vis_binary(df)
# good - specify the original order
vis_binary(df, order = names(df))
```

vis\_compare

*Visually compare two dataframes and see where they are different.***Description**

vis\_compare, like the other vis\_\* families, gives an at-a-glance ggplot of a dataset, but in this case, hones in on visualising **two** different dataframes of the same dimension, so it takes two dataframes as arguments.

**Usage**

```
vis_compare(df1, df2)
```

**Arguments**

|     |                                               |
|-----|-----------------------------------------------|
| df1 | The first dataframe to compare                |
| df2 | The second dataframe to compare to the first. |

**Value**

ggplot2 object displaying which values in each data frame are present in each other, and which are not.

**See Also**

[vis\\_miss\(\)](#) [vis\\_dat\(\)](#) [vis\\_guess\(\)](#) [vis\\_expect\(\)](#) [vis\\_cor\(\)](#)

**Examples**

```
# make a new dataset of iris that contains some NA values
aq_diff <- airquality
aq_diff[1:10, 1:2] <- NA
vis_compare(airquality, aq_diff)
```

---

vis\_cor*Visualise correlations amongst variables in your data as a heatmap*

---

## Description

Visualise correlations amongst variables in your data as a heatmap

## Usage

```
vis_cor(  
  data,  
  cor_method = "pearson",  
  na_action = "pairwise.complete.obs",  
  facet,  
  ...  
)
```

## Arguments

|            |                                                                                                                                                                                                                                                    |
|------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| data       | data.frame                                                                                                                                                                                                                                         |
| cor_method | correlation method to use, from cor: "a character string indicating which correlation coefficient (or covariance) is to be computed. One of "pearson" (default), "kendall", or "spearman": can be abbreviated."                                    |
| na_action  | The method for computing covariances when there are missing values present. This can be "everything", "all.obs", "complete.obs", "na.or.complete", or "pairwise.complete.obs" (default). This option is taken from the cor function argument use., |
| facet      | bare unquoted variable to use for facetting                                                                                                                                                                                                        |
| ...        | extra arguments you may want to pass to cor                                                                                                                                                                                                        |

## Value

ggplot2 object

## Examples

```
vis_cor(airquality)  
vis_cor(airquality, facet = Month)  
vis_cor(mtcars)  
## Not run:  
# this will error  
vis_cor(iris)  
  
## End(Not run)
```

vis\_dat

*Visualises a data.frame to tell you what it contains.***Description**

vis\_dat gives you an at-a-glance ggplot object of what is inside a dataframe. Cells are coloured according to what class they are and whether the values are missing. As vis\_dat returns a ggplot object, it is very easy to customize and change labels, and customize the plot

**Usage**

```
vis_dat(
  x,
  sort_type = TRUE,
  palette = "default",
  warn_large_data = TRUE,
  large_data_size = 9e+05,
  facet
)
```

**Arguments**

|                 |                                                                                                                                                                                                                                                                                                                                                                                                                            |
|-----------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| x               | a data.frame object                                                                                                                                                                                                                                                                                                                                                                                                        |
| sort_type       | logical TRUE/FALSE. When TRUE (default), it sorts by the type in the column to make it easier to see what is in the data                                                                                                                                                                                                                                                                                                   |
| palette         | character "default", "qual" or "cb_safe". "default" (the default) provides the stock ggplot scale for separating the colours. "qual" uses an experimental qualitative colour scheme for providing distinct colours for each Type. "cb_safe" is a set of colours that are appropriate for those with colourblindness. "qual" and "cb_safe" are drawn from <a href="http://colorbrewer2.org/">http://colorbrewer2.org/</a> . |
| warn_large_data | logical - warn if there is large data? Default is TRUE see note for more details                                                                                                                                                                                                                                                                                                                                           |
| large_data_size | integer default is 900000 (given by 'nrow(data.frame) * ncol(data.frame)'). This can be changed. See note for more details.                                                                                                                                                                                                                                                                                                |
| facet           | bare variable name for a variable you would like to facet by. By default there is no faceting. Only one variable can be faceted. You can get the data structure using data_vis_dat and the faceted structure by using group_by and then data_vis_dat.                                                                                                                                                                      |

**Value**

ggplot2 object displaying the type of values in the data frame and the position of any missing values.

**Note**

Some datasets might be too large to plot, sometimes creating a blank plot - if this happens, I would recommend downsampling the data, either looking at the first 1,000 rows or by taking a random sample. This means that you won't get the same "look" at the data, but it is better than a blank plot! See example code for suggestions on doing this.

**See Also**

`vis_miss()` `vis_guess()` `vis_expect()` `vis_cor()` `vis_compare()`

**Examples**

```
vis_dat(airquality)

# experimental colourblind safe palette
vis_dat(airquality, palette = "cb_safe")
vis_dat(airquality, palette = "qual")

# if you have a large dataset, you might want to try downsampling:
## Not run:
library(nycflights13)
library(dplyr)
flights |>
  sample_n(1000) |>
  vis_dat()

flights |>
  slice(1:1000) |>
  vis_dat()

## End(Not run)
```

---

vis\_expect

---

*Visualise whether a value is in a data frame*


---

**Description**

`vis_expect` visualises certain conditions or values in your data. For example, If you are not sure whether to expect -1 in your data, you could write: `vis_expect(data, \ (x) x == -1)`, and you can see if there are times where the values in your data are equal to -1. You could also, for example, explore a set of bad strings, or possible NA values and visualise where they are using `vis_expect(data, \ (x) x %in% bad_strings)` where `bad_strings` is a character vector containing bad strings like N A N/A etc.

**Usage**

```
vis_expect(data, expectation, show_perc = TRUE)
```

**Arguments**

|                          |                                                                                                                                                                                                                                                                                |
|--------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>data</code>        | a data.frame                                                                                                                                                                                                                                                                   |
| <code>expectation</code> | a function, e.g., <code>\(x) {condition}</code> . For example, writing <code>\(x) x &lt; 20</code> would mean "where a variable value is less than 20, replace with NA", and <code>\(x) %in% {vector}</code> would mean "where a variable has values that are in that vector". |
| <code>show_perc</code>   | logical. TRUE now adds in the \ TRUE or FALSE in the whole dataset into the legend. Default value is TRUE.                                                                                                                                                                     |

**Value**

a ggplot2 object

**See Also**

[vis\\_miss\(\)](#) [vis\\_dat\(\)](#) [vis\\_guess\(\)](#) [vis\\_cor\(\)](#) [vis\\_compare\(\)](#)

**Examples**

```
dat_test <- tibble::tribble(
  ~x, ~y,
  -1, "A",
  0, "B",
  1, "C",
  NA, NA
)

vis_expect(dat_test, \(x) x == -1)

vis_expect(airquality, \(x) x == 5.1)

# explore some common NA strings

common_nas <- c(
  "NA",
  "N A",
  "N/A",
  "na",
  "n a",
  "n/a"
)

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

vis_expect(dat_ms, \(x) x %in% common_nas)
```

vis\_guess

*Visualise type guess in a data.frame***Description**

vis\_guess visualises the class of every single individual cell in a dataframe and displays it as ggplot object, similar to vis\_dat. Cells are coloured according to what class they are and whether the values are missing. vis\_guess estimates the class of individual elements using readr::guess\_parser. It may be currently slow on larger datasets.

**Usage**

```
vis_guess(x, palette = "default")
```

**Arguments**

|         |                                                                                                                                                                                                                                                                                                                                                                                                                            |
|---------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| x       | a data.frame                                                                                                                                                                                                                                                                                                                                                                                                               |
| palette | character "default", "qual" or "cb_safe". "default" (the default) provides the stock ggplot scale for separating the colours. "qual" uses an experimental qualitative colour scheme for providing distinct colours for each Type. "cb_safe" is a set of colours that are appropriate for those with colourblindness. "qual" and "cb_safe" are drawn from <a href="http://colorbrewer2.org/">http://colorbrewer2.org/</a> . |

**Value**

ggplot2 object displaying the guess of the type of values in the data frame and the position of any missing values.

**See Also**

[vis\\_miss\(\)](#) [vis\\_dat\(\)](#) [vis\\_expect\(\)](#) [vis\\_cor\(\)](#) [vis\\_compare\(\)](#)

**Examples**

```
messy_vector <- c(TRUE,
  "TRUE",
  "T",
  "01/01/01",
  "01/01/2001",
  NA,
  NaN,
  "NA",
  "Na",
  "na",
  "10",
  10,
  "10.1",
  10.1,
```

```

      "abc",
      "$%TG")
set.seed(1114)
messy_df <- data.frame(var1 = messy_vector,
                       var2 = sample(messy_vector),
                       var3 = sample(messy_vector))
vis_guess(messy_df)

```

---

|               |                                                               |
|---------------|---------------------------------------------------------------|
| vis_histogram | <i>Visualise histogram of numeric columns in a data.frame</i> |
|---------------|---------------------------------------------------------------|

---

### Description

vis\_histogram visualises the distribution of every numeric column in a dataframe and displays it using a faceted ggplot object.

### Usage

```
vis_histogram(x, ...)
```

### Arguments

|     |                                                         |
|-----|---------------------------------------------------------|
| x   | a data.frame                                            |
| ... | Other arguments are passed as geom_histogram arguments. |

### Value

ggplot2 object displaying the guess of the type of values in the data frame and the position of any missing values.

### Examples

```
vis_histogram(airquality, bins = 30)
```

---

|          |                                                       |
|----------|-------------------------------------------------------|
| vis_miss | <i>Visualise a data.frame to display missingness.</i> |
|----------|-------------------------------------------------------|

---

### Description

vis\_miss provides an at-a-glance ggplot of the missingness inside a dataframe, colouring cells according to missingness, where black indicates a missing cell and grey indicates a present cell. As it returns a ggplot object, it is very easy to customize and change labels.

**Usage**

```
vis_miss(
  x,
  cluster = FALSE,
  sort_miss = FALSE,
  show_perc = TRUE,
  show_perc_col = TRUE,
  large_data_size = 9e+05,
  warn_large_data = TRUE,
  facet
)
```

**Arguments**

|                              |                                                                                                                                                                                                                                                                                       |
|------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>x</code>               | a data.frame                                                                                                                                                                                                                                                                          |
| <code>cluster</code>         | logical. TRUE specifies that you want to use hierarchical clustering (mcquitty method) to arrange rows according to missingness. FALSE specifies that you want to leave it as is. Default value is FALSE.                                                                             |
| <code>sort_miss</code>       | logical. TRUE arranges the columns in order of missingness. Default value is FALSE.                                                                                                                                                                                                   |
| <code>show_perc</code>       | logical. TRUE now adds in the \ in the whole dataset into the legend. Default value is TRUE.                                                                                                                                                                                          |
| <code>show_perc_col</code>   | logical. TRUE adds in the \ column into the x axis. Can be disabled with FALSE. Default value is TRUE. No missingness percentage column information will be presented when facet argument is used. Please see the nanian package to provide missingness summaries over groups.        |
| <code>large_data_size</code> | integer default is 900000 (given by <code>'nrow(data.frame) * ncol(data.frame)'</code> ). This can be changed. See note for more details.                                                                                                                                             |
| <code>warn_large_data</code> | logical - warn if there is large data? Default is TRUE see note for more details                                                                                                                                                                                                      |
| <code>facet</code>           | (optional) bare variable name, if you want to create a faceted plot, with one plot per level of the variable. No missingness percentage column information will be presented when facet argument is used. Please see the nanian package to provide missingness summaries over groups. |

**Details**

The missingness summaries in the columns are rounded to the nearest integer. For more detailed summaries, please see the summaries in the nanian R package, specifically, `nanian::miss_var_summary()`.

**Value**

ggplot2 object displaying the position of missing values in the dataframe, and the percentage of values missing and present.

**Note**

Some datasets might be too large to plot, sometimes creating a blank plot - if this happens, I would recommend downsampling the data, either looking at the first 1,000 rows or by taking a random sample. This means that you won't get the same "look" at the data, but it is better than a blank plot! See example code for suggestions on doing this.

**See Also**

`vis_dat()` `vis_guess()` `vis_expect()` `vis_cor()` `vis_compare()`

**Examples**

```
vis_miss(airquality)

vis_miss(airquality, cluster = TRUE)

vis_miss(airquality, sort_miss = TRUE)

vis_miss(airquality, facet = Month)

## Not run:
# if you have a large dataset, you might want to try downsampling:
library(nycflights13)
library(dplyr)
flights |>
  sample_n(1000) |>
  vis_miss()

flights |>
  slice(1:1000) |>
  vis_miss()

## End(Not run)
```

---

vis\_value

*Visualise the value of data values*


---

**Description**

Visualise all of the values in the data on a 0 to 1 scale. Only works on numeric data - see examples for how to subset to only numeric data.

**Usage**

```
vis_value(data, na_colour = "grey90", viridis_option = "D")
```

**Arguments**

|                |                                                                                                                                                                                                               |
|----------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| data           | a data.frame                                                                                                                                                                                                  |
| na_colour      | a character vector of length one describing what colour you want the NA values to be. Default is "grey90"                                                                                                     |
| viridis_option | A character string indicating the colormap option to use. Four options are available: "magma" (or "A"), "inferno" (or "B"), "plasma" (or "C"), "viridis" (or "D", the default option) and "cividis" (or "E"). |

**Value**

a ggplot plot of the values

**Examples**

```
vis_value(airquality)
vis_value(airquality, viridis_option = "A")
vis_value(airquality, viridis_option = "B")
vis_value(airquality, viridis_option = "C")
vis_value(airquality, viridis_option = "E")
## Not run:
library(dplyr)
diamonds |>
  select_if(is.numeric) |>
  vis_value()

## End(Not run)
```

# Index

## \* datasets

- dat\_bin, [3](#)
- typical\_data, [7](#)
- typical\_data\_large, [7](#)

abbreviate\_vars, [2](#)

dat\_bin, [3](#)  
data-vis-cor, [3](#)  
data-vis-dat, [5](#)  
data-vis-miss, [6](#)  
data\_vis\_cor (data-vis-cor), [3](#)  
data\_vis\_dat (data-vis-dat), [5](#)  
data\_vis\_miss (data-vis-miss), [6](#)

typical\_data, [7](#)  
typical\_data\_large, [7](#)

vis\_binary, [9](#)  
vis\_binary(), [3](#)  
vis\_compare, [10](#)  
vis\_compare(), [13–15, 18](#)  
vis\_cor, [11](#)  
vis\_cor(), [10, 13–15, 18](#)  
vis\_dat, [12](#)  
vis\_dat(), [10, 14, 15, 18](#)  
vis\_expect, [13](#)  
vis\_expect(), [10, 13, 15, 18](#)  
vis\_guess, [15](#)  
vis\_guess(), [10, 13, 14, 18](#)  
vis\_histogram, [16](#)  
vis\_miss, [16](#)  
vis\_miss(), [10, 13–15](#)  
vis\_value, [18](#)