Package ‘visdat’

February 15, 2019

Title Preliminary Visualisation of Data

Version 0.5.3

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

Depends R (>= 3.2.2)

License MIT + file LICENSE

LazyData true

RoxygenNote 6.1.1

Imports ggplot2, tidyrr, dplyr, purrr, readr, magrittr, stats, tibble, glue


BugReports https://github.com/ropensci/visdat/issues

Suggests testthat, plotly (>= 4.5.6), knitr, rmarkdown, vdiffr, gdtools, spelling

VignetteBuilder knitr

Encoding UTF-8

Language en-US

NeedsCompilation no

Author Nicholas Tierney [aut, cre] (https://orcid.org/0000-0003-1460-8722),
Sean Hughes [rev] (https://orcid.org/0000-0002-9409-9405), Sean Hughes reviewed the package for rOpenSci, see https://github.com/ropensci/onboarding/issues/87),
Mara Averick [rev] (Mara Averick reviewed the package for rOpenSci, see https://github.com/ropensci/onboarding/issues/87),
Stuart Lee [ctb],
Earo Wang [ctb],
Nic Crane [ctb]

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

Repository CRAN

Date/Publication 2019-02-15 15:30:03 UTC
add_vis_dat_pal (Internal) Add a specific palette to a visdat plot

Description

(Internal) Add a specific palette to a visdat plot

Usage

add_vis_dat_pal(vis_plot, palette)

Arguments

vis_plot visdat plot created using vis_gather_, vis_extract_value and vis_create_
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 http://colorbrewer2.org/.
**Value**

a visdat plot with a particular palette

**Examples**

```r
## Not run:
# see internal use inside vis_guess and vis_dat

## End(Not run)
```

---

**all_numeric**

(Internal) Are they all numeric columns?

**Description**

(Internal) Are they all numeric columns?

**Usage**

```r
all_numeric(x, ...)
```

**Arguments**

- `x` data.frame
- `...` optional extra inputs

**Value**

logical - TRUE means that there is a column with numerics, FALSE means that there is a column that is not numeric

**Examples**

```r
## Not run:
all_numeric(airquality) # TRUE
all_numeric(iris) # FALSE

## End(Not run)
```
**compare_print**  
*(Internal) A utility function for vis_compare*

**Description**

compare_print is an internal function that takes creates a dataframe with information about where there are differences in the dataframe. This function is used in vis_compare. It evaluates on the data (df1 == df2) and (currently) replaces the "true" (the same) with "Same" and FALSE with "Different", unless it is missing (coded as NA), in which case it leaves it as NA.

**Usage**

```
compare_print(x)
```

**Arguments**

- **x**  
  a vector

---

**expect_frame**  
*Create a dataframe to help visualise 'expected' values*

**Description**

Create a dataframe to help visualise 'expected' values

**Usage**

```
expect_frame(data, expectation)
```

**Arguments**

- **data**  
  data.frame
- **expectation**  
  unquoted conditions or "expectations" to test

**Value**

data.frames where expectation are true

**Author(s)**

Stuart Lee and Earo Wang
**Examples**

```r
## Not run:
dat_test <- tibble::tribble(
  ~x, ~y,
  -1, "A",
  0, "B",
  1, "C"
)

expect_frame(dat_test,
            ~ .x == -1)

## End(Not run)
```

---

**expect_guide_label** *(Internal) Label the legend with the percent of missing data*

**Description**

`miss_guide_label` is an internal function to label the legend of `vis_miss`.

**Usage**

```r
expect_guide_label(x)
```

**Arguments**

- `x` is a dataframe passed from `vis_miss(x)`.

**Value**

A `tibble` with two columns `p_miss_lab` and `p_pres_lab`, containing the labels to use for present and missing. A dataframe is returned because I think it is a good style habit compared to a list.

---

**fingerprint** *Take the fingerprint of a data.frame - find the class or return NA*

**Description**

`fingerprint` is an internal function that takes the "fingerprint" of a dataframe, and currently replaces the contents (x) with the class of a given object, unless it is missing (coded as NA), in which case it leaves it as NA. The name "fingerprint" is taken from the csv-fingerprint, of which the package, `visdat`, is based upon.
Usage

fingerprint(x)

Arguments

x  a vector

gather_cor  (Internal) create a tidy dataframe of correlations suitable for plotting

description

(Internal) create a tidy dataframe of correlations suitable for plotting

Usage

gather_cor(data, cor_method = "pearson",
          na_action = "pairwise.complete.obs")

Arguments

data  data.frame

cor_method  correlation method to use, from cor: "a character string indicating which corre-
            lation 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 "pair-
            wise.complete.obs" (default). This option is taken from the cor function argument use.

Value

tidy dataframe of correlations

Examples

gather_cor(airquality)
guess_type

(Internal) Guess the type of each individual cell in a dataframe

Description

vis_guess uses guess_type to guess cell elements, like fingerprint.

Usage

guess_type(x)

Arguments

x is a vector of values you want to guess

Value

a character vector that describes the suspected class. e.g., "10" is an integer, "20.11" is a double, "text" is character, etc.

Examples

## Not run:
guess_type(1)
guess_type("x")
guess_type(c("1", "0L"))
purrr::map_df(iris, guess_type)
## End(Not run)

label_col_missing_pct

(Internal) Create labels for the columns containing the % missing data

Description

(Internal) Create labels for the columns containing the % missing data

Usage

label_col_missing_pct(x, col_order_index)
Arguments

x data.frame
col_order_index the order of the columns

Value
data.frame containing the missingness percent down to 0.1 percent

---

miss_guide_label Label the legend with the percent of missing data

Description

miss_guide_label is an internal function for vis_miss to label the legend.

Usage

miss_guide_label(x)

Arguments

x is a dataframe passed from vis_miss(x).

Value

a tibble with two columns p_miss_lab and p_pres_lab, containing the labels to use for present and missing. A dataframe is returned because I think it is a good style habit compared to a list.

---

test_if_dataframe Test if input is a data.frame

Description

Test if input is a data.frame

Usage

test_if_dataframe(x)

Arguments

x object
Value

an error if input (x) is not a data.frame

Examples

## Not run:
# success
test_if_dataframe(airquality)
# fail
test_if_dataframe(AirPassengers)

## End(Not run)

typical_data

A small toy dataset of imaginary people

Description

A dataset containing information about some randomly generated people, created using the excel-
ent `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", "Na-
  tive", 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  A small toy dataset of imaginary people

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 favorite 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
**Description**

visdat is a package that helps with the preliminary visualisation of data. visdat makes it easy to visualise your whole dataset so that you can visually identify problems.
**See Also**

It's main functions are:

- `vis_dat()`
- `vis_miss()`
- `vis_guess()`
- `vis_compare()`
- `vis_expect()`

Learn more about visdat at [www.njtierney.com/visdat/articles/using_visdat.html](http://www.njtierney.com/visdat/articles/using_visdat.html)

---

**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**

```r
# 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)
```
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", ...)

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.

... extra arguments you may want to pass to cor

Value

ggplot2 object

Examples

vis_cor(airquality)
## Not run:
vis_cor(mtcars)
vis_cor(iris)

## End(Not run)
vis_create_  

(Internal) Create a boilerplate for visualisations of the vis_family

Description

(Internal) Create a boilerplate for visualisations of the vis_family

Usage

vis_create_(x)

Arguments

x  a dataframe in longformat as transformed by vis_gather_ and vis_extract_value.

Value

a ggplot object

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)

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 http://colorbrewer2.org/.

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, this can be changed. See note for more details
vis_expect

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)

## Not run:
# 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:
library(nycflights13)
library(dplyr)
flights %>%
sample_n(1000) %>%
vis_dat()

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

## End(Not run)
vis_expect(data, ~.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

data a data.frame

expectation a formula following the syntax: ~.x {condition}. For example, writing ~.x < 20 would mean "where a variable value is less than 20, replace with NA", and ~.x %in% {vector} would mean "where a variable has values that are in that vector".

show_perc logical. TRUE now adds in the % of expectations are 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 == -1)

## Not run:  
vis_expect(airquality, ~.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(~xL ~yL ~zL
1, "A", -100,
3, "N/A", -99,
NA, NA, -98,
"N A", "E", -101,
"na", "F", -1)

vis_expect(dat_ms, ~x %in% common_nas)

## End(Not run)

---

**vis_extract_value_** *(Internal) Add values of each row as a column*

**Description**

This adds information about each row, so that when called by plotly, the values are made visible on hover. Warnings are suppressed because tidyr gives a warning about type coercion, which is fine.

**Usage**

```r
vis_extract_value_(x)
```

**Arguments**

- `x` dataframe created from `vis_gather_`

**Value**

the x dataframe with the added column value.

---

**vis_gather_** *(Internal) Gather rows into a format appropriate for grid visualisation*

**Description**

(Internal) Gather rows into a format appropriate for grid visualisation

**Usage**

```r
vis_gather_(x)
```

**Arguments**

- `x` a dataframe
Value
data.frame gathered to have columns "variables", "valueType", and a row id called "rows".

---

**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 http://colorbrewer2.org/.

**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**

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

"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_miss

Visualise a data.frame to display missingness.

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)

Arguments

x
  a data.frame
cluster
  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.
sort_miss
  logical. TRUE arranges the columns in order of missingness. Default value is FALSE.
show_perc
  logical. TRUE now adds in the % of missing/complete data in the whole dataset into the legend. Default value is TRUE.
show_perc_col
  logical. TRUE adds in the % missing data in a given column into the x axis. Can be disabled with FALSE. Default value is TRUE.
large_data_size
  integer default is 900000, this can be changed. See note for more details
warn_large_data
  logical - warn if there is large data? Default is TRUE see note for more details

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)

## Not run:
vis_miss(airquality, cluster = TRUE)

vis_miss(airquality, sort_miss = TRUE)

# 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)
Index

*Topic datasets
   typical_data, 9
   typical_data_large, 10

add_vis_dat_pal, 2
all_numeric, 3

compare_print, 4

expect_frame, 4
expect_guide_label, 5

fingerprint, 5

gather_cor, 6
guess_type, 7

label_col_missing_pct, 7

miss_guide_label, 8

test_if_dataframe, 8
   typical_data, 9
   typical_data_large, 10

vis_compare, 12
   vis_compare(), 12, 15, 16, 18, 20
vis_cor, 13
   vis_cor(), 12, 15, 16, 18, 20
vis_create, 14
vis_dat, 14
   vis_dat(), 12, 16, 18, 20
vis_expect, 15
   vis_expect(), 12, 15, 18, 20
vis_extract_value, 17
vis_gather, 17
vis_guess, 18
   vis_guess(), 12, 15, 16, 20
vis_miss, 19
   vis_miss(), 12, 15, 16, 18
visdat, 11
visdat-package (visdat), 11