

Package: **ibawds** (via r-universe)

September 17, 2024

Type Package

Title Functions and Datasets for the Data Science Course at IBAW

Version 0.6.0.9006

Author Stefan Lanz

Maintainer Stefan Lanz <slanz1137@gmail.com>

Description A collection of useful functions and datasets for the Data Science Course at IBAW.

License MIT + file LICENSE

URL <https://stibu81.github.io/ibawds/>

BugReports <https://github.com/stibu81/ibawds/issues>

Encoding UTF-8

LazyData true

Language en-GB

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.2

Depends R (>= 3.6.0), dslabs

Imports stats, grDevices, rlang, rstudioapi, remotes, ggplot2, scales, dplyr (>= 1.1.0), stringr, magrittr, cli

Suggests tidyverse, rmarkdown, knitr, kableExtra, caret, party, RANN, ranger, reshape2, lubridate, hexbin, patchwork, ggrepel, GGally, ggfortify, deldir, writexl, cowplot, DT, gutenbergr, tidytext, Lahman, HistData, titanic, BiocManager, waldo, rvest, cIValid, usethis, vdiff, testthat (>= 3.0.0), covr, spelling

Config/testthat/edition 3

Repository <https://stibu81.r-universe.dev>

RemoteUrl <https://github.com/stibu81/ibawds>

RemoteRef HEAD

RemoteSha 1eae17ab72f0f855b10e33e875bf0d9d40a75570

Contents

bills	2
breast_cancer	3
check_ibawds_setup	4
cluster_with_centers	4
cran_history	5
define_latex_stats	6
dentition	7
distribution_plot	8
downgrade_packages	9
find_similar_colour	10
galton_sons	11
get_reading_exercise_files	11
grading_tables	12
install_ibawds	13
mtcars2	14
noisy_data	14
n_available_packages	15
protein	16
rand_with_cor	17
rescale	18
seatbelts	19
set_slide_options	19
voronoi_diagram	20
wine_quality	21
Index	23

bills	<i>Summarised Data on Restaurant Bills</i>
-------	--

Description

Summary of data on restaurant bills from the dataset `reshape2::tips`. Labels are in German.

Usage

bills

Format

A data frame with 8 rows and 4 variables:

sex sex of the bill payer

time time of day

smoker whether there were smokers in the party

mean_bill mean of all the bills in dollars

breast_cancer

Wisconsin Breast Cancer Database

Description

Breast cancer database obtained from the University of Wisconsin Hospitals, Madison from Dr. William H. Wolberg. The data were collected in 8 from 1989 to 1991 and are sorted in chronological order.

Usage

breast_cancer

Format

a tibble with 699 rows and 11 variables. All numerical values are integers in the range 1 to 10.

id sample code number

clump_thick clump thickness

unif_cell_size uniformity of cell size

unif_cell_shape uniformity of cell shape

marg_adh marginal adhesion

ep_cell_size single epithelial cell size

bare_nucl bare nuclei

bland_chromat bland chromatin

norm_nucl normal nucleoli

mitoses mitoses

class "benign" (458) or "malignant" (241)

Source

The data is available on the [UC Irvine Machine Learning Repository](#).

O. L. Mangasarian and W. H. Wolberg, *Cancer diagnosis via linear programming*, SIAM News, Volume 23(5) (1990) 1 & 18.

check_ibawds_setup *Check If the User Is Ready for the Course*

Description

Check if the current system is ready for the course by verifying the following:

- R and RStudio are up to date
- the ibawds package is up to date
- all the required packages are installed

The function must be run from RStudio in order to run properly.

Usage

```
check_ibawds_setup()
```

Value

a logical indicating whether the system is up to date (invisibly). Messages inform the user about the status of the system.

cluster_with_centers *Cluster Data According to Centres and Recompute Centres*

Description

For a given dataset and given centres, `cluster_with_centers()` assigns each data point to its closest centre and then recomputes the centres as the mean of all points assigned to each class. An initial set of random cluster centres can be obtained with `init_rand_centers()`. These functions can be used to visualise the mechanism of k-means.

Usage

```
cluster_with_centers(data, centers)
```

```
init_rand_centers(data, n, seed = sample(1000:9999, 1))
```

Arguments

<code>data</code>	a data.frame containing only the variables to be used for clustering.
<code>centers</code>	a data.frame giving the centres of the clusters. It must have the same number of columns as data.
<code>n</code>	the number of cluster centres to create
<code>seed</code>	a random seed for reproducibility

Value

a list containing two tibbles:

- `centers`: the new centres of the clusters computed after cluster assignment with the given centres
- `cluster`: the cluster assignment for each point in data using the centres that were passed to the function

Examples

```
# demonstrate k-means with iris data
# keep the relevant columns
iris2 <- iris[, c("Sepal.Length", "Petal.Length")]

# initialise the cluster centres
clust <- init_rand_centers(iris2, n = 3, seed = 2435)

# plot the data with the cluster centres
library(ggplot2)
ggplot(iris2, aes(x = Sepal.Length, y = Petal.Length)) +
  geom_point(data = clust$centers, aes(colour = factor(1:3)),
            shape = 18, size = 6) +
  geom_point() +
  scale_colour_brewer(palette = "Set1")

# assign clusters and compute new centres
clust_new <- cluster_with_centers(iris2, clust$centers)

# plot the data with clustering
clust$cluster <- clust_new$cluster
voronoi_diagram(clust, x = "Sepal.Length", y = "Petal.Length",
               data = iris2)

# plot the data with new cluster centres
clust$centers <- clust_new$centers
voronoi_diagram(clust, x = "Sepal.Length", y = "Petal.Length",
               data = iris2, colour_data = FALSE)

# this procedure may be repeated until the algorithm converges
```

Description

Table with the number of packages available on CRAN and the current R version for historic dates back to 21 June 2001.

Usage

```
cran_history
```

Format

A data frame with 69 rows and 4 variables.

date date

n_packages the number of available R packages on CRAN

version the then current version of R

source source of the data (see 'Details')

Details

Data on the number of packages on CRAN between 2001-06-21 and 2014-04-13 is obtained from [CRANpackages](#) from the package [Ecdat](#). This data was collected by John Fox and Spencer Graves. Intervals between data points are irregularly spaced. These data are marked with "John Fox" or "Spencer Graves" in the column source. They are licenced under GPL-2/GPL-3.

Data between 2014-10-01 and 2023-03-06 was collected by the package author from CRAN snapshots on Microsoft's MRAN, which was retired on 1 July 2023. Data was collected on the first day of each quarter. These data are marked with "MRAN" in the column source.

Newer data has been collected in irregular intervals using the functions [n_available_packages\(\)](#) and [available_r_version\(\)](#). These data are marked with "CRAN" in the column source.

Examples

```
library(ggplot2)
ggplot(cran_history, aes(x = date, y = n_packages)) +
  geom_point()
```

define_latex_stats *Define LaTeX commands for statistical symbols*

Description

Add the definitions for various useful LaTeX equation symbols for statistics to an RMarkdown document.

Usage

```
define_latex_stats()
```

Details

Run this function from within a code chunk in a RMarkdown document with options `results = "asis"` and `echo = FALSE` (see "Examples"). It only works for pdf output.

It defines the following macros: `\E`, `\P`, `\Var`, `\Cov`, `\Cor`, `\SD`, `\SE`, `\Xb`, `\Yb`.

Value

The function returns NULL invisibly. The command definitions are output as a side effect.

Examples

```
## Not run:
# add this code chunk to a RMarkdown document
```{r results = "asis", echo = FALSE}
 define_latex_stats()
```

## End(Not run)
```

dentition

Dentition of Mammals

Description

Dental formulas for various mammals. The dental formula describes the number of incisors, canines, premolars and molars per quadrant. Upper and lower teeth may differ and are therefore shown separately. The total number of teeth is twice the number given.

Usage

```
dentition
```

Format

Data frame with 66 rows and 9 variables:

name name of the mammal

I number of top incisors

i number of bottom incisors

C number of top canines

c number of bottom canines

P number of top premolars

p number of bottom premolars

M number of top molars

m number of bottom molars

Source

The data have been downloaded from <https://people.sc.fsu.edu/~jburkardt/datasets/hartigan/file19.txt>

They come from the following textbook:

Hartigan, J. A. (1975). *Clustering Algorithms*, John Wiley, New York.

Table 9.1, page 170.

distribution_plot *Plot Density and Distribution Function With Markings*

Description

Create plots of the density and distribution functions of a probability distribution. It is possible to mark points and shade the area under the curve.

Usage

```
distribution_plot(  
  fun,  
  range,  
  ...,  
  points = NULL,  
  var = "x",  
  title = "Verteilungsfunktion",  
  is_discrete = NULL  
)
```

```
density_plot(  
  fun,  
  range,  
  ...,  
  from = NULL,  
  to = NULL,  
  points = NULL,  
  var = "x",  
  title = "Dichte",  
  is_discrete = NULL  
)
```

Arguments

| | |
|-------|--|
| fun | a density or distribution function that takes quantiles as its first argument. |
| range | numeric vector of length two giving the range of quantiles to be plotted. |
| ... | further arguments that are passed to fun(). |

| | |
|-------------|---|
| points | numeric vector giving quantiles where the function should be marked with a red dot (continuous) or a red bar (discrete). |
| var | character giving the name of the quantile variable. This is only used to label the axes. |
| title | character giving the title of the plot |
| is_discrete | logical indicating whether this is a discrete distribution. For discrete distributions, a bar plot is created. If omitted, the function tries to automatically determine, whether the distributions is discrete. In case this should fail, set this argument explicitly. |
| from, to | numeric values giving start and end of a range where the area under the density will be shaded (continuous) or the bars will be drawn in red (discrete). If only one of the two values is given, the shading will start at negative infinity or go until positive infinity, respectively. |

Value

a ggplot object

Examples

```
# plot density of the normal distribution
density_plot(dnorm, c(-5, 7),
             mean = 1, sd = 2,
             to = 3)

# plot distribution function of the Poisson distribution
distribution_plot(ppois, c(0, 12),
                 lambda = 4,
                 points = c(2, 6, 10),
                 var = "y")
```

downgrade_packages *Downgrade Packages to the Previous Version*

Description

Downgrade packages to the previous version available on CRAN. This is useful in order to prepare the system for a demonstration of package updates.

Usage

```
downgrade_packages(pkg)
```

Arguments

pkg character with the names of the packages to be downgraded.

Details

Downgrading is only possible for packages that are currently installed. For packages that are not installed, a warning is issued.

The function uses `remotes::install_version()` to install a version of a package that is older than the currently installed version.

Value

A character vector with the names of the downgraded packages, invisibly.

find_similar_colour *Find a Named Colour that is Similar to Any Given Colour*

Description

Find the named colour that is most similar to a given colour.

Usage

```
find_similar_colour(  
  colour,  
  distance = c("euclidean", "manhattan"),  
  verbose = interactive()  
)
```

Arguments

| | |
|----------|--|
| colour | a colour specified in one of three forms: a hexadecimal string of the form "#rrggbb" or "#rrggbbaa", a numeric vector of length 3 or a numeric matrix with dimensions <code>c(3, 1)</code> , as it is returned by <code>col2rgb()</code> . Numeric values must be between 0 and 255. |
| distance | character indicating the distance metric to be used. |
| verbose | should additional output be produced? This shows the RGB values for the input colour, the most similar named colour and the difference between the two. |

Value

a character of length one with the name of the most similar named colour.

Examples

```
find_similar_colour("#d339da")  
find_similar_colour(c(124, 34, 201))  
  
# suppress additional output  
find_similar_colour("#85d3a1", verbose = FALSE)
```

```
# use Manhattan distance
find_similar_colour(c(124, 34, 201), distance = "manhattan")
```

galton_sons

Galton's data on the heights of fathers and their children

Description

Two tables of father's heights with heights of one of their sons (`galton_sons`) or daughters (`galton_daughters`), respectively. All heights are given in centimetres. It is created from `HistData::GaltonFamilies` by randomly selecting one son or daughter per family. Since some families consist of only sons or only daughters, not all families are contained in both tables.

Usage

```
galton_sons
```

```
galton_daughters
```

Format

Two data frames with 179 (`galton_sons`) or 176 (`galton_daughters`)\$ rows, respectively, and 2 variables:

father size of the father in cm.

son/daughter size of the son or daughter, respectively, in cm.

get_reading_exercise_files

Get Files for File Reading Exercise

Description

Copy the files for an exercise for reading files to a directory.

Usage

```
get_reading_exercise_files(path, unzip = TRUE)
```

Arguments

path path where the files should be copied to.

unzip logical indicating whether the files should be unzipped. Set this to FALSE if unzipping fails.

Details

There are 8 files in total. Apart from a few errors that were introduced for the purpose of the exercise, they all contain the same data: information about 100 randomly selected Swiss municipalities. The full file can be downloaded from <https://www.bfs.admin.ch/bfsstatic/dam/assets/7786544/master>.

Value

Logical indicating the success of the copy operation.

| | |
|-----------------------------|---|
| <code>grading_tables</code> | <i>Tables Used for Grading the Papers</i> |
|-----------------------------|---|

Description

These functions create two tables that can be used for the grading of the student's papers.

Usage

```
create_minreq_table(repro, n_tab, n_plot_kinds, n_plots, n_stat)
```

```
create_grading_table(p_text, p_tab, p_plot, p_code, p_stat)
```

Arguments

| | |
|---------------------------|--|
| <code>repro</code> | logical, is the paper reproducible? |
| <code>n_tab</code> | integer, number of tables |
| <code>n_plot_kinds</code> | integer, number of different kinds of plots |
| <code>n_plots</code> | integer, number of plots |
| <code>n_stat</code> | integer, number of statistical computations |
| <code>p_text</code> | numeric between 0 and 3, points given for the text |
| <code>p_tab</code> | numeric between 0 and 3, points given for the tables |
| <code>p_plot</code> | numeric between 0 and 5, points given for the plots |
| <code>p_code</code> | numeric between 0 and 5, points given for the code |
| <code>p_stat</code> | numeric between 0 and 5, points given for the statistic computations |

Details

The tables are created using `knitr::kable()` and `kableExtra::kableExtra` is used for additional styling.

`create_minreq_table()` creates a table that checks that the minimal requirements are satisfied:

- the paper must be reproducible
- there must be at least one table and two kinds of plots

- there must be at least 5 plots and tables
- there must be at least two statistical computations

The table lists for each of those requirement whether it is satisfied or not.

`create_grading_table()` creates a table that gives grades in percent for each of five categories:

- Text
- Tables
- Plots
- Code
- Statistical computations

In each category, up to five points may be awarded. The last row of the table gives the percentage over all categories.

Value

both functions return an object of class `kableExtra`.

| | |
|-----------------------------|---|
| <code>install_ibawds</code> | <i>Install the R-Packages Required for the Course</i> |
|-----------------------------|---|

Description

A number of R-packages are used in the courses and the video lectures. They are also dependencies of this package. Use `install_ibawds()` to install the packages that are not yet installed.

Usage

```
install_ibawds()
```

Details

This function checks whether all the packages that `ibawds` depends on, imports or suggests are installed. In interactive sessions, it either informs the user that all packages are installed or asks to install missing packages. The function relies on `rlang::check_installed()`.

Value

nothing or NULL invisibly

`mtcars2`*Dataset mtcars without row names*

Description

In the `mtcars` dataset, the names of the car models are stored as row names. However, when working with `ggplot2` and other packages from the `tidyverse`, it is convenient to have all data in columns. `mtcars2` is a variant of `mtcars` that contains car models in a column instead of storing them as row names. `mtcars_na` is the same dataset as `mtcars2`, but some of the columns contain missing values.

Usage`mtcars2``mtcars2_na`**Format**

A data frame with 32 rows and 12 variables. The format is identical to `mtcars` and details can be found in its documentation. The only difference is that the car model names are stored in the column `model` instead of the row names.

`noisy_data`*Noisy Data From a Tenth Order Polynomial*

Description

Training and test data created from a tenth order polynomial with added noise. The polynomial is given by

$$f(x) = 2x - 10x^5 + 15x^{10}$$

The noise follows a standard normal distribution. The data can be used to demonstrate overfitting. It is inspired by section II. B. in [A high-bias, low-variance introduction to Machine Learning for physicists](#)

Usage`noisy_data`**Format**

a list of two tibbles with two columns each. x stands for the independent, y for the dependent variable. The training data (`noisy_data$train`) contains 1000 rows, the test data (`noisy_data$test`) 20 rows.

References

P. Mehta et al., *A high-bias, low-variance introduction to Machine Learning for physicists* Phys. Rep. 810 (2019), 1-124. [arXiv:1803.08823](https://arxiv.org/abs/1803.08823) [doi:10.1016/j.physrep.2019.03.001](https://doi.org/10.1016/j.physrep.2019.03.001)

n_available_packages *Number of Available R Packages and R Versions from CRAN*

Description

Obtain the number of available packages on CRAN and the current R version.

Usage

```
n_available_packages(cran = getOption("repos"))
```

```
available_r_version(cran = getOption("repos"))
```

Arguments

cran character vector giving the base URL of the CRAN server to use.

Details

The number of packages on CRAN and the R version can be obtained for selected dates in the past from the dataset [cran_history](#).

Note: Previously, these functions could obtain the number of packages on CRAN and the then current R version also for past dates by using snapshots from Microsoft's MRAN. However, MRAN shut down on 1 July 2023 such that this functionality is no longer available.

Value

the number of available packages as an integer or the R version number as a character

See Also

[cran_history](#)

protein

Protein Consumption in European Countries

Description

Protein Consumption from various sources in European countries in unspecified units. The exact year of data collection is not known but the oldest known publication of the data is from 1973.

Usage

protein

Format

Data frame with 25 rows and 10 variables:

country name of the country

red_meat red meat

white_meat white meat

eggs eggs

milk milk

fish fish

cereals cereals

starch starchy foods

nuts pulses, nuts, oil-seeds

fruit_veg fruits, vegetables

Source

The data have been downloaded from <https://raw.githubusercontent.com/jgscott/STA380/master/data/protein.csv>

They come from the following book:

Hand, D. J. et al. (1994). *A Handbook of Small Data Sets*, Chapman and Hall, London.

Chapter 360, p. 297.

In the book, it is stated that the data have first been published in

Weber, A. (1973). *Agrarpolitik im Spannungsfeld der internationalen Ernährungspolitik*, Institut für Agrarpolitik und Marktlehre, Kiel.

`rand_with_cor`*Create a Random Vector With Fixed Correlation With Another Vector*

Description

`rand_with_cor()` creates a vector of random number that has correlation ρ with a given vector y . Also mean and standard deviation of the random vector can be fixed by the user. By default, they will be equal to the mean and standard deviation of y , respectively.

Usage

```
rand_with_cor(y, rho, mu = mean(y), sigma = sd(y))
```

Arguments

| | |
|--------------------|--|
| <code>y</code> | a numeric vector |
| <code>rho</code> | numeric value between -1 and 1 giving the desired correlation. |
| <code>mu</code> | numeric value giving the desired mean |
| <code>sigma</code> | numeric value giving the desired standard deviation |

Value

a vector of the same length as y that has correlation ρ with y .

Source

This solution is based on an [answer](#) by [whuber](#) on [Cross Validated](#).

Examples

```
x <- runif(1000, 5, 8)

# create a random vector with positive correlation
y1 <- rand_with_cor(x, 0.8)
all.equal(cor(x, y1), 0.8)

# create a random vector with negative correlation
# and fixed mean and standard deviation
y2 <- rand_with_cor(x, -0.3, 2, 3)
all.equal(cor(x, y2), -0.3)
all.equal(mean(y2), 2)
all.equal(sd(y2), 3)
```

`rescale`*Rescale Mean And/Or Standard Deviation of a Vector*

Description

Rescale Mean And/Or Standard Deviation of a Vector

Usage

```
rescale(x, mu = mean(x), sigma = sd(x))
```

Arguments

| | |
|--------------------|---|
| <code>x</code> | numeric vector |
| <code>mu</code> | numeric value giving the desired mean |
| <code>sigma</code> | numeric value giving the desired standard deviation |

Details

By default, mean and standard deviation are not changed, i.e., `rescale(x)` is identical to `x`. Only if a value is specified for `mu` and/or `sigma` the mean and/or the standard deviation are rescaled.

Value

a numeric vector with the same length as `x` with mean `mu` and standard deviation `sigma`.

Examples

```
x <- runif(1000, 5, 8)

# calling rescale without specifying mu and sigma doesn't change anything
all.equal(x, rescale(x))

# change the mean without changing the standard deviation
x1 <- rescale(x, mu = 3)
all.equal(mean(x1), 3)
all.equal(sd(x1), sd(x))

# rescale mean and standard deviation
x2 <- rescale(x, mu = 3, sigma = 2)
all.equal(mean(x2), 3)
all.equal(sd(x2), 2)
```

`seatbelts`*Road Casualties in Great Britain 1969-84*

Description

Extract of the data in the [Seatbelts](#) dataset as a data frame. The original dataset is a multiple time series (class `mts`). Labels are in German.

Usage

```
seatbelts
```

Format

A data frame with 576 rows and 3 variables:

date data of the first data of the month for which the data was collected.

seat seat where the persons that were killed or seriously injured were seated. One of "Fahrer" (driver's seat), "Beifahrer" (front seat), "Rücksitz" (rear seat).

victims number of persons that were killed or seriously injured.

`set_slide_options`*Set Options for Slides*

Description

Set options for ggplot plots and tibble outputs for IBAW slides.

Usage

```
set_slide_options(  
  ggplot_text_size = 22,  
  ggplot_margin_pt = rep(10, 4),  
  tibble_print_max = 12,  
  tibble_print_min = 8  
)
```

Arguments

`ggplot_text_size`

Text size to be used in ggplot2 plots. This applies to all texts in the plots.

`ggplot_margin_pt`

numeric vector of length 4 giving the sizes of the top, right, bottom, and left margins in points.

`tibble_print_max`
Maximum number of rows printed for a tibble. Set to `Inf` to always print all rows.

`tibble_print_min`
Number of rows to be printed if a tibble has more than `tibble_print_max` rows.

Details

The function uses `ggplot2::theme_update()` to modify the default theme for ggplot and `options()` to set base R options that influence the printing of tibbles.

Note that if you make changes to these options in a R Markdown file, you may have to delete the knitr cache in order for the changes to apply.

Value

a named list (invisibly) with two elements containing the old values of the options for the ggplot theme and the base R options, respectively. These can be used to reset the ggplot theme and the base R options to their previous values.

`voronoi_diagram` *Create a Voronoi Diagram for a Clustering*

Description

Create a Voronoi diagram for a given clustering object.

Usage

```
voronoi_diagram(
  cluster,
  x,
  y,
  data = NULL,
  show_data = !is.null(data),
  colour_data = TRUE,
  legend = TRUE,
  point_size = 2,
  linewidth = 0.7
)
```

Arguments

`cluster` an object containing the result of a clustering, e.g., created by `kmeans()`. It must contain the fields `cluster` and `centers`.

`x, y` character giving the names of the variables to be plotted on the x- and y-axis.

| | |
|-------------|--|
| data | The data that has been used to create the clustering. If this is provided, the extension of the plot is adapted to the data and the data points are plotted unless this is suppressed by specifying <code>show_data = FALSE</code> . |
| show_data | should the data points be plotted? This is TRUE by default if data is given. |
| colour_data | should the data points be coloured according to the assigned cluster? |
| legend | should a colour legend for the clusters be plotted? |
| point_size | numeric indicating the size of the data points and the cluster centres. |
| linewidth | numeric indicating the width of the lines that separate the areas for the clusters. Set to 0 to show no lines at all. |

Details

The function uses the `deldir` package to create the polygons for the Voronoi diagram. The code has been inspired by `ggvoronoi`, which can handle more complex situations.

References

Garrett et al., *ggvoronoi: Voronoi Diagrams and Heatmaps with ggplot2*, Journal of Open Source Software 3(32) (2018) 1096, doi:[10.21105/joss.01096](https://doi.org/10.21105/joss.01096)

Examples

```
cluster <- kmeans(iris[, 1:4], centers = 3)
voronoi_diagram(cluster, "Sepal.Length", "Sepal.Width", iris)
```

| | |
|--------------|---------------------|
| wine_quality | <i>Wine Quality</i> |
|--------------|---------------------|

Description

Physicochemical data and quality ratings for red and white Portuguese **Vinho Verde** wines.

Usage

```
wine_quality
```

Format

a tibble with 6497 rows and 13 variables:

colour colour of the wine; "red" (1'599) or "white" (4'898)

fixed_acidity tartaric acid per volume in g/dm^3

volatile_acidity acetic acid per volume in g/dm^3

citric_acid citric acid per volume in g/dm^3

residual_sugar residual sugar per volume in g/dm^3
chlorides sodium chloride per volume in g/dm^3
free_sulfur_dioxide free sulphur dioxide per volume in mg/dm^3
total_sulfur_dioxide total sulphur dioxide per volume in mg/dm^3
density density in g/dm^3
pH pH value
sulphates potassium sulphate per volume in g/dm^3
alcohol alcohol content per volume in %
quality quality score between 0 (worst) and 10 (best) determined by sensory analysis.

Source

The data is available on the [UC Irvine Machine Learning Repository](#).

P. Cortez, A. Cerdeira, F. Almeida, T. Matos and J. Reis, *Modeling wine preferences by data mining from physicochemical properties*, Decision Support Systems 47(4) (2009), 547-553.

Index

* datasets

- bills, [2](#)
- breast_cancer, [3](#)
- cran_history, [5](#)
- dentition, [7](#)
- galton_sons, [11](#)
- mtcars2, [14](#)
- noisy_data, [14](#)
- protein, [16](#)
- seatbelts, [19](#)
- wine_quality, [21](#)

available_r_version
(n_available_packages), [15](#)

available_r_version(), [6](#)

bills, [2](#)

breast_cancer, [3](#)

check_ibawds_setup, [4](#)

cluster_with_centers, [4](#)

col2rgb(), [10](#)

cran_history, [5](#), [15](#)

create_grading_table (grading_tables),
[12](#)

create_minreq_table (grading_tables), [12](#)

define_latex_stats, [6](#)

density_plot (distribution_plot), [8](#)

dentition, [7](#)

distribution_plot, [8](#)

downgrade_packages, [9](#)

find_similar_colour, [10](#)

galton_daughters (galton_sons), [11](#)

galton_sons, [11](#)

get_reading_exercise_files, [11](#)

ggplot2::theme_update(), [20](#)

grading_tables, [12](#)

HistData::GaltonFamilies, [11](#)

init_rand_centers
(cluster_with_centers), [4](#)

install_ibawds, [13](#)

kableExtra::kableExtra, [12](#)

kmeans(), [20](#)

knitr::kable(), [12](#)

mtcars, [14](#)

mtcars2, [14](#)

mtcars2_na (mtcars2), [14](#)

n_available_packages, [15](#)

n_available_packages(), [6](#)

noisy_data, [14](#)

options(), [20](#)

protein, [16](#)

rand_with_cor, [17](#)

remotes::install_version(), [10](#)

rescale, [18](#)

reshape2::tips, [2](#)

rlang::check_installed(), [13](#)

Seatbelts, [19](#)

seatbelts, [19](#)

set_slide_options, [19](#)

voronoi_diagram, [20](#)

wine_quality, [21](#)