R topics documented:

bills ................................................................. 2
breast_cancer .................................................... 3
cran_history .................................................... 4
define_latex_stats ............................................ 5
dentition ......................................................... 6
distribution_plot .............................................. 7
downgrade_packages ......................................... 8
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 Summarised Data on Restaurant Bills

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
**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](https://archive.ics.uci.edu/ml/datasets/breast+cancer+wisconsin+%28radial+nomal+cell+biopsy+clumps+core+biopsy%29).

**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**

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

**Arguments**

- `data`: a data.frame containing only the variables to be used for clustering.
- `centers`: a data.frame giving the centres of the clusters. It must have the same number of columns as `data`.
- `n`: the number of cluster centres to create
- `seed`: 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**

```r
# 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

cran_history

<table>
<thead>
<tr>
<th>cran_history</th>
<th>History of the Number of Available CRAN Packages</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>Table with the number of packages available on CRAN and the current R version for historic dates back to 21 June 2001.</td>
</tr>
<tr>
<td>Usage</td>
<td>cran_history</td>
</tr>
<tr>
<td>Format</td>
<td>Data frame with 25 rows and 10 variables. The first column (Country) indicates the name of the country, the other columns indicate protein consumption from nine sources in unknown units.</td>
</tr>
<tr>
<td>Details</td>
<td>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. Newer data was obtained using the functions n_available_packages() and available_r_version() which extract the information from CRAN snapshots on MRAN. One data point per quarter is available starting on 2014-10-01. These data are marked with MRAN in the column source.</td>
</tr>
</tbody>
</table>
**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**

```r
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**

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

```r
define_latex_stats()
```

```r```

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


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**

```r
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 "#rrggbbaa", a numeric vector of length 3 or a numeric matrix with dimensions c(3, 1), as it is returned by `col2rgb()`. 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), 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.

grading_tables Tables Used for Grading the Papers

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

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

install_ibawds

Install the R-Packages Required for the Course

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 Polygon*

**Description**

Training and test data create 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.
n_available_packages

References


n_available_packages  Number of Available R Packages and R Versions from MRAN

Description

MRAN has an archive of Snapshots of CRAN dating back to September 17 2014. These functions return the number of available packages and the available R version according to the snapshot of https://cran.r-project.org on MRAN.

Usage

n_available_packages(date = Sys.Date())

available_r_version(date = Sys.Date())

Arguments

date  the date of the snapshot to be used. It can be a Date object or a character in the format %Y-%m-%d.

Details

MRAN has data starting from September 17 2014. Data for a few selected dates before September 17 2014 can be obtained from the dataset CRANpackages from the package Ecdat. A more complete dataset ranging from 2001 until today is included in the package as cran_history.

Note that for some dates there is no snapshot on MRAN. The function will return an error in those cases.

Value

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

See Also

cran_history
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:

<table>
<thead>
<tr>
<th>country</th>
<th>name of the country</th>
</tr>
</thead>
<tbody>
<tr>
<td>red_meat</td>
<td>red meat</td>
</tr>
<tr>
<td>white_meat</td>
<td>white meat</td>
</tr>
<tr>
<td>eggs</td>
<td>eggs</td>
</tr>
<tr>
<td>milk</td>
<td>milk</td>
</tr>
<tr>
<td>fish</td>
<td>fish</td>
</tr>
<tr>
<td>cereals</td>
<td>cereals</td>
</tr>
<tr>
<td>starch</td>
<td>starchy foods</td>
</tr>
<tr>
<td>nuts</td>
<td>pulses, nuts, oil-seeds</td>
</tr>
<tr>
<td>fruit_veg</td>
<td>fruits, vegetables</td>
</tr>
</tbody>
</table>

Source

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

They come from the following book:

Chapter 360, p. 297.

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

Description

`rand_with_cor()` creates a vector of random number that has correlation `rho` 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

- `y`: a numeric vector
- `rho`: numeric value between -1 and 1 giving the desired correlation.
- `mu`: numeric value giving the desired mean
- `sigma`: numeric value giving the desired standard deviation

Value

a vector of the same length as `y` that has correlation `rho` 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**

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

**Arguments**

- `x`: numeric vector
- `mu`: numeric value giving the desired mean
- `sigma`: 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**

```r
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)
```
**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**

```r
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.

---

**Description**

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

**Usage**

```r
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**

```r
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.
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 `show_data = FALSE`.

- `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

Examples
```
cluster <- kmeans(iris[, 1:4], centers = 3)
```

---

**wine_quality**

**Wine Quality**

**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.

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(), 5

bills, 2
breast_cancer, 3

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

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(), 5
noisy_data, 14

options(), 20
protein, 16

rand_with_cor, 17
remotes::install_version(), 9
rescale, 18
reshape2::tips, 2
rlang::check_installed(), 13

Seatbelts, 19
seatbelts, 19
set_slide_options, 19

voronoi_diagram, 20

wine_quality, 21