Package ‘datasauRus’

February 27, 2024

Title Datasets from the Datasaurus Dozen

Version 0.1.8

Description The Datasaurus Dozen is a set of datasets with the same summary statistics. They retain the same summary statistics despite having radically different distributions. The datasets represent a larger and quirkier object lesson that is typically taught via Anscombe's Quartet (available in the 'datasets' package). Anscombe's Quartet contains four very different distributions with the same summary statistics and as such highlights the value of visualisation in understanding data, over and above summary statistics. As well as being an engaging variant on the Quartet, the data is generated in a novel way. The simulated annealing process used to derive datasets from the original Datasaurus is detailed in "Same Stats, Different Graphs: Generating Datasets with Varied Appearance and Identical Statistics through Simulated Annealing" <doi:10.1145/3025453.3025912>.

License MIT + file LICENSE

URL https://github.com/jumpingrivers/datasauRus,
     https://jumpingrivers.github.io/datasauRus/

BugReports https://github.com/jumpingrivers/datasauRus/issues

Depends R (>= 3.5.0)

Suggests dplyr, ggplot2, knitr, rmarkdown, spelling, testthat

VignetteBuilder knitr

Encoding UTF-8

LazyData true

RoxygenNote 7.2.3

Language en-US

NeedsCompilation no

Author Colin Gillespie [cre, aut],
    Steph Locke [aut],
    Alberto Cairo [dct],
    Rhian Davies [aut],
R topics documented:

box_plots ................................................................. 2
box_plots_long ......................................................... 3
datasaurus_dozen ..................................................... 5
datasaurus_dozen_wide ................................................ 6
simpsons_paradox ...................................................... 8
simpsons_paradox_wide ............................................... 9
twelve_from_slant_alternate_long ................................. 10
twelve_from_slant_alternate_wide ................................. 11
twelve_from_slant_long ............................................... 12
twelve_from_slant_wide .............................................. 13

Index 16

box_plots Box plot data

Description
This dataset is the box plot data produced by Matjeka & Fitzmaurice to demonstrate applicability of their process.

Usage

box_plots

Format
A data frame with 2484 rows and 5 variables:

- left: data pulled to the left
- lines: data with arbitrary spikes along a range
- normal: normally distributed data
- right: data pulled to the right
- split: split data
References

Examples
summary(box.plot)

## base plot
#save current settings
state = par("mar", "mfrow")
par(mfrow = c(5, 2), mar = c(1, 2, 2, 1))
nms = names(box.plot)
for (i in 1:5) {
  nm = nms[i]
  hist(box.plot[[nms[i]]],
       breaks = 100,
       main = nm)
  boxplot(box.plot[[nms[i]]],
          horizontal = TRUE)
}
#reset settings
par(state)

## ggplot
if (require(ggplot2)) {
  ggplot(box.plot, aes(x = left)) + geom_density()
  ggplot(box.plot, aes(x = lines)) + geom_density()
  ggplot(box.plot, aes(x = normal)) + geom_density()
  ggplot(box.plot, aes(x = right)) + geom_density()
  ggplot(box.plot, aes(x = split)) + geom_density()
}

---

box.plot_long

Box plot data
Description

This dataset is the box plot data produced by Matjeka & Fitzmaurice to demonstrate applicability of their process.

Usage

box_plots_long

Format

A data frame with 12420 rows and 2 variables:

- **Plot**: either the left, lines, normal, right or split boxplot
- **Values**: the corresponding values from each dataset

References


Examples

```
summary(box_plots)

## base plot

#save current settings
state = par("mar", "mfrow")

par(mfrow = c(5, 2), mar = c(1, 2, 2, 1))

nms = names(box_plots)

for (i in 1:5) {
  nm = nms[i]
  hist(box_plots[[nms[i]]],
       breaks = 100,
       main = nm)
  boxplot(box_plots[[nms[i]]],
          horizontal = TRUE)
}

#reset settings
par(state)

## ggplot

if (require(ggplot2)) {
```
Description
A dataset demonstrating the utility of visualization. These 12 datasets are equal in standard measures: mean, standard deviation, and Pearson’s correlation.

Usage
datasaurus_dozen

Format
A data frame with 1846 rows and 3 variables:
• dataset: indicates which dataset the data are from
• x: x-values
• y: y-values

References

Examples
if (require(ggplot2)) {
  ggplot(datasaurus_dozen, aes(x = x, y = y, colour = dataset)) +
  geom_point() +
  theme_void() +
  theme(legend.position = "none") +
  facet_wrap(~dataset, ncol = 3)
}
# Base R plots
state = par("mar", "mfrow")

# plot
par(mfrow = c(5, 3), mar = c(1, 2, 2, 1))

sets = sort(unique(datasaurus_dozen$dataset))

for (s in sets) {
  df = datasaurus_dozen[datasaurus_dozen$dataset == s, ]
  plot(df$x, df$y, pch = 16)
  title(s)
}

# reset settings
par(state)

---

**datasaurus_dozen_wide**  
*Datasaurus Dozen (wide) data*

**Description**

A dataset demonstrating the utility of visualization. These 12 datasets are equal in standard measures: mean, standard deviation, and Pearson’s correlation.

**Usage**

`datasaurus_dozen_wide`

**Format**

A data frame with 142 rows and 26 variables:

- `away_x`: x-values for the `away` dataset
- `away_y`: y-values for the `away` dataset
- `bullseye_x`: x-values for the `bullseye` dataset
- `bullseye_y`: y-values for the `bullseye` dataset
- `circle_x`: x-values for the `circle` dataset
- `circle_y`: y-values for the `circle` dataset
- `dino_x`: x-values for the `dinosaur` dataset!
- `dino_y`: y-values for the `dinosaur` dataset!
- `dots_x`: x-values for the `dots` dataset
- `dots_y`: y-values for the `dots` dataset
- `h_lines_x`: x-values for the `h_lines` dataset
• **h_lines_y**: y-values for the h_lines dataset
• **high_lines_x**: x-values for the high_lines dataset
• **high_lines_y**: y-values for the high_lines dataset
• **slant_down_x**: x-values for the slant_down dataset
• **slant_down_y**: y-values for the slant_down dataset
• **slant_up_x**: x-values for the slant_up dataset
• **slant_up_y**: y-values for the slant_up dataset
• **star_x**: x-values for the star dataset
• **star_y**: y-values for the star dataset
• **v_lines_x**: x-values for the v_lines dataset
• **v_lines_y**: y-values for the v_lines dataset
• **wide_lines_x**: x-values for the wide_lines dataset
• **wide_lines_y**: y-values for the wide_lines dataset
• **x_shape_x**: x-values for the x_shape dataset
• **x_shape_y**: y-values for the x_shape dataset

References


Examples

```r
# Save current settings
state = par("mar", "mfrow")

# Base R Plots
par(mfrow = c(5, 3), mar = c(1, 3, 3, 1))

nms = names(datasaurus_dozen_wide)
for (i in seq(1, 25, by = 2)) {
  nm = substr(nms[i], 1, nchar(nms[i]) - 2)
  plot(datasaurus_dozen_wide[[nms[i]]],
      datasaurus_dozen_wide[[nms[i + 1]]],
      xlab = "", ylab = "", main = nm)
}

#reset settings
par(state)
```
simpsons_paradox  

Simpsons Paradox data

Description
A dataset demonstrating Simpson’s Paradox with a strongly positively correlated dataset (simpson_1) and a dataset with the same positive correlation as simpson_1, but where individual groups have a strong negative correlation (simpson_2).

Usage
simpsons_paradox

Format
A data frame with 444 rows and 3 variables:
- **dataset**: indicates which of the two datasets the data are from, simpson_1 or simpson_2
- **x**: x-values
- **y**: y-values

References

Examples
```r
if (require(ggplot2)) {
  ggplot(simpsons_paradox, aes(x = x, y = y, colour = dataset)) +
  geom_point() +
  theme(legend.position = "none") +
  facet_wrap(~dataset, ncol = 3)
}

# Base R Plots
state = par("mfrow")
par(mfrow = c(1, 2))
sets = unique(datasaurus_dozen$dataset)
for (i in 1:2) {
  df = simpsons_paradox[simpsons_paradox$dataset == paste0("simpson", i), ]
  plot(df$x, df$y, pch = 16, xlab = "", ylab = "")
```
A dataset demonstrating Simpson’s Paradox with a strongly positively correlated dataset (simpson_1) and a dataset with the same positive correlation as simpson_1, but where individual groups have a strong negative correlation (simpson_2).

Usage

simpsons_paradox_wide

Format

A data frame with 222 rows and 4 variables:

- **simpson_1_x**: x-values from the simpson_1 dataset
- **simpson_1_y**: y-values from the simpson_1 dataset
- **simpson_2_x**: x-values from the simpson_2 dataset
- **simpson_2_y**: y-values from the simpson_2 dataset

References


Examples

```r
#save current settings
state = par("mar", "mfrow")

par(mfrow = c(1, 2))

plot(simpsons_paradox_wide[["simpson_1_x"]],
     simpsons_paradox_wide[["simpson_1_y"]],
     xlab = "x", ylab = "y", main = "Simpson's Paradox 1")

plot(simpsons_paradox_wide[["simpson_2_x"]],
     simpsons_paradox_wide[["simpson_2_y"]],
     xlab = "x", ylab = "y", main = "Simpson's Paradox 2")
```
Description

A dataset demonstrating the utility of visualization. These 12 datasets are equal in non-parametric measures: median, interquartile range, and Spearman’s rank correlation.

Usage

twelve_from_slant_alternate_long

Format

A data frame with 2184 rows and 3 variables:

- **dataset**: the dataset the data are from
- **x**: x-values
- **y**: y-values

References


Examples

```r
if (require(ggplot2)) {
  ggplot(twelve_from_slant_alternate_long, aes(x = x, y = y, colour = dataset)) +
  geom_point() +
  theme_void() +
  theme(legend.position = "none") +
  facet_wrap(~dataset, ncol = 3)
}
```

# Base R Plots

```r
state = par("mfrow", "mar")
par(mfrow = c(4, 3), mar = c(2, 2, 2, 2))
sets = sort(unique(twelve_from_slant_alternate_long$dataset))
```
for (s in sets) {
    df = twelve_from_slant_alternate_long[twelve_from_slant_alternate_long$dataset == s, ]
    plot(df$x, df$y, pch = 16, xlab = "", ylab = "")
    title(s)
}
par(state)

twelve_from_slant_alternate_wide

Twelve From Slant Alternate (wide) data

Description

A dataset demonstrating the utility of visualization. These 12 datasets are equal in non-parametric measures: median, interquartile range, and Spearman’s rank correlation.

Usage

twelve_from_slant_alternate_wide

Format

A data frame with 182 rows and 24 variables:

- **bullseye_x**: x-values for the bullseye dataset
- **bullseye_y**: y-values for the bullseye dataset
- **circle_x**: x-values for the circle dataset
- **circle_y**: y-values for the circle dataset
- **dots_x**: x-values for the dots dataset
- **dots_y**: y-values for the dots dataset
- **h_lines_x**: x-values for the h_lines dataset
- **h_lines_y**: y-values for the h_lines dataset
- **high_lines_x**: x-values for the high_lines dataset
- **high_lines_y**: y-values for the high_lines dataset
- **slant_x**: x-values for the slant dataset
- **slant_y**: y-values for the slant dataset
- **slant_down_x**: x-values for the slant_down dataset
- **slant_down_y**: y-values for the slant_down dataset
- **slant_up_x**: x-values for the slant_up dataset
- **slant_up_y**: y-values for the slant_up dataset
- **star_x**: x-values for the star dataset
• star_y: y-values for the star dataset
• v_lines_x: x-values for the v_lines dataset
• v_lines_y: y-values for the v_lines dataset
• wide_lines_x: x-values for the wide_lines dataset
• wide_lines_y: y-values for the wide_lines dataset
• x_shape_x: x-values for the x_shape dataset
• x_shape_y: y-values for the x_shape dataset

References

Examples

```r
# save current settings
state = par("mar", "mfrow")

# plot
par(mfrow = c(4, 3), mar = c(1, 3, 3, 1))

nms = names(twelve_from_slant_alternate_wide)
for (i in seq(1, 23, by = 2)) {
  nm = substr(nms[i], 1, nchar(nms[i]) - 2)
  plot(twelve_from_slant_alternate_wide[[nms[i]]],
       twelve_from_slant_alternate_wide[[nms[i + 1]]],
       xlab = "", ylab = "", main = nm)
}

# reset settings
par(state)
```

twelve_from_slant_long

*Twelve From Slant (long) data*

Description

A dataset demonstrating the utility of visualization. These 12 datasets are equal in standard measures: mean, standard deviation, and Pearson’s correlation.

Usage
twelve_from_slant_long
Format

A data frame with 2184 rows and 3 variables:

- **dataset**: the dataset the data are from
- **x**: x-values
- **y**: y-values

References


Examples

```r
if (require(ggplot2)) {
  ggplot(twelve_from_slant_long, aes(x = x, y = y, colour = dataset)) +
  geom_point() +
  theme_void() +
  theme(legend.position = "none") +
  facet_wrap(~dataset, ncol = 3)
}

# Base R Plots
state = par("mfrow", "mar")
par(mfrow = c(4, 3), mar = c(3, 2, 2, 2))
sets = sort(unique(twelve_from_slant_long$dataset))
for (s in sets) {
  df = twelve_from_slant_long[twelve_from_slant_long$dataset == s, ]
  plot(df$x, df$y, pch = 16, xlab = "", ylab = "")
  title(s)
}
par(state)
```

**twelve_from_slant_wide**

*Twelve From Slant (wide) data*

Description

A dataset demonstrating the utility of visualization. These 12 datasets are equal in standard measures: mean, standard deviation, and Pearson’s correlation.
Usage
twelve_from_slant_wide

Format

A data frame with 182 rows and 24 variables:

- bullseye_x: x-values for the bullseye dataset
- bullseye_y: y-values for the bullseye dataset
- circle_x: x-values for the circle dataset
- circle_y: y-values for the circle dataset
- dots_x: x-values for the dots dataset
- dots_y: y-values for the dots dataset
- h_lines_x: x-values for the h_lines dataset
- h_lines_y: y-values for the h_lines dataset
- high_lines_x: x-values for the high_lines dataset
- high_lines_y: y-values for the high_lines dataset
- slant_x: x-values for the slant dataset
- slant_y: y-values for the slant dataset
- slant_down_x: x-values for the slant_down dataset
- slant_down_y: y-values for the slant_down dataset
- slant_up_x: x-values for the slant_up dataset
- slant_up_y: y-values for the slant_up dataset
- star_x: x-values for the star dataset
- star_y: y-values for the star dataset
- v_lines_x: x-values for the v_lines dataset
- v_lines_y: y-values for the v_lines dataset
- wide_lines_x: x-values for the wide_lines dataset
- wide_lines_y: y-values for the wide_lines dataset
- x_shape_x: x-values for the x_shape dataset
- x_shape_y: y-values for the x_shape dataset

References

Examples

```r
# save current settings
state = par("mar", "mfrow")

# plot
par(mfrow = c(4, 3), mar = c(1, 3, 3, 1))

nms = names(twelve_from_slant_wide)
for (i in seq(1, 23, by = 2)) {
  nm = substr(nms[i], 1, nchar(nms[i]) - 2)
  plot(twelve_from_slant_wide[[nms[i]]],
       twelve_from_slant_wide[[nms[i + 1]]],
       xlab = "", ylab = "", main = nm)
}

# reset settings
par(state)
```
Index

* datasets
  box_plots, 2
  box_plots_long, 3
  datasaurus_dozen, 5
  datasaurus_dozen_wide, 6
  simpsons_paradox, 8
  simpsons_paradox_wide, 9
  twelve_from_slant_alternate_long, 10
  twelve_from_slant_alternate_wide, 11
  twelve_from_slant_long, 12
  twelve_from_slant_wide, 13

box_plots, 2
box_plots_long, 3

datasaurus_dozen, 5
datasaurus_dozen_wide, 6

simpsons_paradox, 8
simpsons_paradox_wide, 9

twelve_from_slant_alternate_long, 10
twelve_from_slant_alternate_wide, 11
twelve_from_slant_long, 12
twelve_from_slant_wide, 13