Package ‘ds4psy’

May 6, 2020

Type Package
Title Data Science for Psychologists
Version 0.2.1
Date 2020-05-06
Author Hansjoerg Neth [aut, cre]
Maintainer Hansjoerg Neth <h.neth@uni.kn>
Description All datasets and functions required for the examples and exercises of the book "Data Science for Psychologists" (by Hansjoerg Neth, Konstanz University, 2020), available at <https://bookdown.org/hneth/ds4psy/>. The book and course introduce principles and methods of data science to students of psychology and other biological or social sciences. The 'ds4psy' package primarily provides datasets, but also functions for data generation and manipulation (e.g., of text and time data) and graphics that are used in the book and its exercises. All functions included in 'ds4psy' are designed to be instructive and entertaining, rather than elegant or efficient.
Depends R (>= 3.4.0)
Imports ggplot2, cowplot, unikn
Suggests knitr, rmarkdown, spelling
Collate 'util_fun.R' 'color_fun.R' 'data.R' 'data_fun.R' 'text_fun.R' 'time_fun.R' 'theme_fun.R' 'plot_fun.R' 'start.R'
Encoding UTF-8
LazyData true
License CC BY-SA 4.0
BugReports https://github.com/hneth/ds4psy/issues
VignetteBuilder knitr
RoxygenNote 7.1.0
Language en-US
NeedsCompilation no
Repository CRAN
Date/Publication 2020-05-06 15:50:02 UTC
R topics documented:

- capitalize .................................................. 3
- caseflip ................................................... 4
- coin .......................................................... 5
- count_char .................................................. 6
- cur_date ..................................................... 6
- cur_time ..................................................... 7
- data_1 ....................................................... 8
- data_2 ....................................................... 9
- data_t1 ..................................................... 9
- data_t1_de ............................................... 10
- data_t1_tab ............................................. 10
- data_t2 ..................................................... 11
- data_t3 ..................................................... 11
- data_t4 ..................................................... 12
- dice ........................................................ 13
- dice_2 ..................................................... 14
- ds4psy_guide ........................................... 15
- expWide ................................................... 15
- falsePosPsy_all ......................................... 16
- fame ....................................................... 17
- is.wholenumber .......................................... 18
- l33t_rul35 ................................................ 19
- make_grid ................................................ 20
- num_as_char ............................................. 20
- num_as_ordinal ......................................... 22
- outliers ................................................... 23
- pal_ds4psy ............................................... 23
- pal_n_sq .................................................. 24
- pi_100k .................................................... 25
- plot_fn ..................................................... 25
- plot_fun ................................................... 27
- plot_n ...................................................... 28
- plot_text ................................................ 30
- plot_tiles ................................................ 32
- posPsy_AHI_CESD ....................................... 34
- posPsy_long ............................................. 35
- posPsy_p_info .......................................... 36
- posPsy_wide ............................................. 37
- read_ascii ............................................... 38
- sample_dates ........................................... 39
- sample_times ........................................... 40
- t3 .......................................................... 41
- t4 .......................................................... 42
- table6 ..................................................... 42
- table7 ..................................................... 43
- table8 ..................................................... 43
capitalize

capitalize converts the case of each word’s n initial characters (typically to upper) in a string of text x.

Description

capitalize converts the case of each word’s n initial characters (typically to upper) in a string of text x.

Usage

capitalize(x, n = 1, upper = TRUE, as_text = TRUE)

Arguments

x A string of text (required).
n Number of initial characters to convert. Default: n = 1.
upper Convert to uppercase? Default: upper = TRUE.
as_text Return word vector as text (i.e., one character string)? Default: as_text = TRUE.

See Also

caseflip for converting the case of all letters.
Other text functions: caseflip(), count_char(), l33t_rul35, read_ascii(), transl33t()
caseflip

caseflip flips the case of characters in a string of text \texttt{x}.

Description

caseflip flips the case of characters in a string of text \texttt{x}.

Usage

caseflip(x)

Arguments

\texttt{x} \\
A string of text (required).

See Also

\texttt{capitalize} for converting the case of initial letters.

Other text functions: \texttt{capitalize(), count_char(), l33t_rul35, read_ascii(), transl33t()}

Examples

\texttt{x <- c("Hello world!", "This is a 1st sentence.", "This is the 2nd sentence.", "The end.")} \\
\texttt{caseflip(x)}
Flip a fair coin (with 2 sides "H" and "T") n times.

Description

coin generates a sequence of events that represent the results of flipping a fair coin n times.

Usage

```r
coin(n = 1, events = c("H", "T"))
```

Arguments

- **n**: Number of coin flips. Default: \(n = 1\).
- **events**: Possible outcomes (as a vector). Default: \(events = c("H", "T")\).

Details

By default, the 2 possible events for each flip are "H" (for "heads") and "T" (for "tails").

See Also

Other random functions: `dice_2()`, `dice()`, `sample_dates()`, `sample_times()`

Examples

```r
# Basics:
coin()
table(coin(n = 100))
table(coin(n = 100, events = LETTERS[1:3]))

# Note an oddity:
coin(10, events = 8:9)  # works as expected, but
coin(10, events = 9:9)  # odd: see sample() for an explanation.

# Limits:
coin(2:3)
coin(NA)
coin(0)
coin(1/2)
coin(3, events = "X")
coin(3, events = NA)
coin(NULL, NULL)
```
count_char

count_char counts the frequency of characters in a string of text x.

Description

count_char counts the frequency of characters in a string of text x.

Usage

count_char(x, case_sense = TRUE, rm_specials = TRUE, sort_freq = TRUE)

Arguments

x A string of text (required).

case_sense Boolean: Distinguish lower- vs. uppercase characters? Default: case_sense = TRUE.

rm_specials Boolean: Remove special characters? Default: rm_specials = TRUE.

sort_freq Boolean: Sort output by character frequency? Default: sort_freq = TRUE.

See Also

plot_text for a corresponding plot function.

Other text functions: capitalize(), caseflip(), l33t_rul35, read_ascii(), transl33t()

Examples

# Default:
x <- c("Hello!", "This is a 1st sentence.", "This is the 2nd sentence.", "The end.")
count_char(x)

# Options:
count_char(x, case_sense = FALSE)
count_char(x, rm_specials = FALSE)
count_char(x, sort_freq = FALSE)

cur_date

Current date (in yyyy-mm-dd or dd-mm-yyyy format).

Description

cur_date provides a relaxed version of Sys.time() that is sufficient for most purposes.

Usage

cur_date(rev = FALSE, sep = "-")
cur_time

Arguments

- **rev** (Boolean): Reverse from "yyyy-mm-dd" to "dd-mm-yyyy" format? Default: `rev = FALSE`.
- **sep** (Character): Separator to use. Default: `sep = "-"`.

Details

cur_date returns `Sys.time()` (in " using current system settings.
By default, this corresponds to the " format used as the ISO 8601 standard.
For more options, see the date() and Sys.Date() functions of **base R** and the plethora of formatting options for Sys.time().

See Also

date() and today() functions of the **lubridate** package; date(), Sys.Date(), and Sys.time() functions of **base R**.
Other date and time functions: cur_time(), what_date(), what_day(), what_month(), what_time(), what_week(), what_year()

Examples

```r
cur_date()
cur_date(sep = "/")
cur_date(rev = TRUE)
cur_date(rev = TRUE, sep = ".")
```

---

cur_time

Current time (in hh:mm or hh:mm:ss format).

Description

cur_time provides a satisficing version of `Sys.time()` that is sufficient for most purposes.

Usage

cur_time(seconds = FALSE, sep = ":")

Arguments

- **seconds** (Boolean): Show time with seconds? Default: `seconds = FALSE`.
- **sep** (Character): Separator to use. Default: `sep = ":"`.

Details

cur_time returns `Sys.time()` (in " using current system settings.
For a time zone argument, see the now() function of the **lubridate** package.
See Also

now() function of the lubridate package; Sys.time() function of base R.

Other date and time functions: cur_date(), what_date(), what_day(), what_month(), what_time(), what_week(), what_year()

Examples

cur_time()
cur_time(seconds = TRUE)
cur_time(sep = ",")

data_1

Data import data_1.

Description

data_1 is a fictitious dataset to practice data import (from a DELIMITED file).

Usage

data_1

Format

A table with 100 cases (rows) and 4 variables (columns).

Source


See Also

Other datasets: data_2, data_t1_de, data_t1_tab, data_t1, data_t2, data_t3, data_t4, exp_wide, falsePosPsy_all, fame, outliers, pi_100k, posPsy_AHI_CESD, posPsy_long, posPsy_p_info, posPsy_wide, t3, t4, t_1, t_2, t_3, t_4, table6, table7, table8, tb
**data_2**  
*Data import data_2.*

---

**Description**

Data_2 is a fictitious dataset to practice data import (from a FWF file).

**Usage**

data_2

**Format**

A table with 100 cases (rows) and 4 variables (columns).

**Source**


**See Also**

Other datasets: `data_1, data_t1_de, data_t1_tab, data_t1, data_t2, data_t3, data_t4, exp_wide, falsePosPsy_all, fame, outliers, pi_100k, posPsy_AHI_CESD, posPsy_long, posPsy_p_info, posPsy_wide, t3, t4, t_1, t_2, t_3, t_4, table6, table7, table8, tb`

---

**data_t1**  
*Data table data_t1.*

---

**Description**

Data_t1 is a fictitious dataset to practice importing and joining data (from a CSV file).

**Usage**

data_t1

**Format**

A table with 20 cases (rows) and 4 variables (columns).

**Source**

data_t1_de

Data import data_t1_de.

Description

data_t1_de is a fictitious dataset to practice data import (from a CSV file, de/European style).

Usage

data_t1_de

Format

A table with 20 cases (rows) and 4 variables (columns).

Source

See CSV data at http://rpository.com/ds4psy/data/data_t1_de.csv.

See Also

Other datasets: data_1, data_2, data_t1_de, data_t1_tab, data_t2, data_t3, data_t4, exp_wide, falsePosPsy_all, fame, outliers, pi_100k, posPsy_AHI_CESD, posPsy_long, posPsy_p_info, posPsy_wide, t3, t4, t_1, t_2, t_3, t_4, table6, table7, table8, tb

data_t1_tab

Data import data_t1_tab.

Description

data_t1_tab is a fictitious dataset to practice data import (from a TAB file).

Usage

data_t1_tab

Format

A table with 20 cases (rows) and 4 variables (columns).
**data_t2**

**Source**

See TAB-delimited data at [http://rpository.com/ds4psy/data/data_t1_tab.csv](http://rpository.com/ds4psy/data/data_t1_tab.csv).

**See Also**

Other datasets: `data_1, data_2, data_t1_de, data_t1, data_t2, data_t3, data_t4, exp_wide, falsePosPsy_all, fame, outliers, pi_100k, posPsy_AHI_CESD, posPsy_long, posPsy_p_info, posPsy_wide, t3, t4, t_1, t_2, t_3, t_4, table6, table7, table8, tb`

---

**data_t2**  
*Data table data_t2.*

**Description**

`data_t2` is a fictitious dataset to practice importing and joining data (from a CSV file).

**Usage**

`data_t2`

**Format**

A table with 20 cases (rows) and 4 variables (columns).

**Source**


**See Also**

Other datasets: `data_1, data_2, data_t1_de, data_t1, data_t2, data_t3, data_t4, exp_wide, falsePosPsy_all, fame, outliers, pi_100k, posPsy_AHI_CESD, posPsy_long, posPsy_p_info, posPsy_wide, t3, t4, t_1, t_2, t_3, t_4, table6, table7, table8, tb`

---

**data_t3**  
*Data table data_t3.*

**Description**

`data_t3` is a fictitious dataset to practice importing and joining data (from a CSV file).

**Usage**

`data_t3`
**Data Table**

**data_t4**

**Format**
A table with 20 cases (rows) and 4 variables (columns).

**Source**

**See Also**
Other datasets: data_1, data_2, data_t1_de, data_t1_tab, data_t1, data_t2, data_t4, exp-wide, falsePosPsy_all, fame, outliers, pi_100k, posPsy_AHI_CESD, posPsy_long, posPsy_p_info, posPsy_wide, t3, t4, t_1, t_2, t_3, t_4, table6, table7, table8, tb
dice

**Description**

dice generates a sequence of events that represent the results of throwing a fair dice (with a given number of events or number of sides) \( n \) times.

**Usage**

dice(\( n = 1 \), events = 1:6)

**Arguments**

- **n**: Number of dice throws. Default: \( n = 1 \).
- **events**: Events to draw from (or number of sides). Default: \( \text{events} = 1:6 \).

**Details**

By default, the 6 possible events for each throw of the dice are the numbers from 1 to 6.

**See Also**

Other random functions: `coin()`, `dice_2()`, `sample_dates()`, `sample_times()`

**Examples**

# Basics:
dice()
table(dice(10^4))

# 5-sided dice:
dice(events = 1:5)
table(dice(100, events = 5))

# Strange dice:
dice(5, events = 8:9)
table(dice(100, LETTERS[1:3]))

# Note:
dice(10, 1)
table(dice(100, 2))

# Note an oddity:
dice(10, events = 8:9)  # works as expected, but
dice(10, events = 9:9)  # odd: see sample() for an explanation.

# Limits:
dice(NA)

dice_2

Description

dice_2 is a variant of dice that generates a sequence of events that represent the results of throwing a dice (with a given number of sides) n times.

Usage

dice_2(n = 1, sides = 6)

Arguments

n Number of dice throws. Default: n = 1.
sides Number of sides. Default: sides = 6.

Details

Something is wrong with this dice. Can you examine it and measure its problems in a quantitative fashion?

See Also

Other random functions: coin(), dice(), sample_dates(), sample_times()

Examples

# Basics:
dice_2()
table(dice_2(100))

# 10-sided dice:
dice_2(sides = 10)
table(dice_2(100, sides = 10))

# Note:
dice_2(10, 1)
table(dice_2(5000, sides = 5))

# Note an oddity:
dice_2(n = 10, sides = 8:9)  # works, but
dice_2(n = 10, sides = 9:9)  # odd: see sample() for an explanation.

---

**Description**

Opens user guide of the ds4psy package.

**Usage**

ds4psy.guide()

---

**exp_wide**

Data exp_wide.

---

**Description**

exp_wide is a fictitious dataset to practice tidying data (here: converting from wide to long format).

**Usage**

exp_wide

**Format**

A table with 10 cases (rows) and 7 variables (columns).

**Source**


**See Also**

Other datasets: data_1, data_2, data_t1_de, data_t1_tab, data_t1, data_t2, data_t3, data_t4,
falsePosPsy_all, fame, outliers, pi_100k, posPsy_AHI_CESD, posPsy_long, posPsy_p_info,
posPsy_wide, t3, t4, t_1, t_2, t_3, t_4, table6, table7, table8, tb
falsePosPsy_all  False Positive Psychology data.

Description

falsePosPsy_all is a dataset containing the data from 2 studies designed to highlight problematic research practices within psychology.

Usage

falsePosPsy_all

Format

A table with 78 cases (rows) and 19 variables (columns):

Details

Simmons, Nelson and Simonsohn (2011) published a controversial article with a necessarily false finding. By conducting simulations and 2 simple behavioral experiments, the authors show that flexibility in data collection, analysis, and reporting dramatically increases the rate of false-positive findings.

study  Study ID.

id  Participant ID.

aged  Days since participant was born (based on their self-reported birthday).

aged365  Age in years.

female  Is participant a woman? 1: yes, 2: no.

dad  Father’s age (in years).

mom  Mother’s age (in years).

potato  Did the participant hear the song 'Hot Potato' by The Wiggles? 1: yes, 2: no.

when64  Did the participant hear the song 'When I am 64' by The Beatles? 1: yes, 2: no.

kalimba  Did the participant hear the song 'Kalimba' by Mr. Scrub? 1: yes, 2: no.

cond  In which condition was the participant? control: Subject heard the song 'Kalimba' by Mr. Scrub; potato: Subject heard the song 'Hot Potato' by The Wiggles; 64: Subject heard the song 'When I am 64' by The Beatles.

root  Could participant report the square root of 100? 1: yes, 2: no.

bird  Imagine a restaurant you really like offered a 30 percent discount for dining between 4pm and 6pm. How likely would you be to take advantage of that offer? Scale from 1: very unlikely, 7: very likely.

political  In the political spectrum, where would you place yourself? Scale: 1: very liberal, 2: liberal, 3: centrist, 4: conservative, 5: very conservative.
quarterback If you had to guess who was chosen the quarterback of the year in Canada last year, which of the following four options would you choose? 1: Dalton Bell, 2: Daryll Clark, 3: Jarius Jackson, 4: Frank Wilczynski.

olddays How often have you referred to some past part of your life as “the good old days”? Scale: 11: never, 12: almost never, 13: sometimes, 14: often, 15: very often.


computer Computers are complicated machines. Scale from 1: strongly disagree, to 5: strongly agree.

diner Imagine you were going to a diner for dinner tonight, how much do you think you would like the food? Scale from 1: dislike extremely, to 9: like extremely.

See https://bookdown.org/hneth/ds4psy/B-2-datasets-false.html for codebook and more information.

Source

Articles


See Also

Other datasets: data_1, data_2, data_t1_de, data_t1_tab, data_t1, data_t2, data_t3, data_t4, exp_wide,fame, outliers, pi_100k, posPsy_AHI_CESD, posPsy_long, posPsy_p_info, posPsy_wide, t3, t4, t_1, t_2, t_3, t_4, table6, table7, table8, tb

---

fame         Data table fame.

Description

fame is a dataset to practice working with dates.

fame contains the names, areas, dates of birth (DOB), and — if applicable — the dates of death (DOD) of famous people.
Usage

fame

Format

A table with 38 cases (rows) and 4 variables (columns).

Source


See Also

Other datasets: data_1, data_2, data_t1_de, data_t1_tab, data_t1, data_t2, data_t3, data_t4, exp_wide, falsePosPsy_all, outliers, pi_100k, posPsy_AHI_CESD, posPsy_long, posPsy_p_info, posPsy_wide, t3, t4, t_1, t_2, t_3, t_4, table6, table7, table8, tb

is.wholenumber

Test for whole numbers (i.e., integers).

Description

is.wholenumber tests if x contains integer numbers.

Usage

is.wholenumber(x, tol = .Machine$double.eps^0.5)

Arguments

x       Number(s) to test (required, accepts numeric vectors).
tol     Numeric tolerance value. Default: tol = .Machine$double.eps^0.5 (see ?.Machine for details).

Details

is.wholenumber does what the base R function is.integer is not designed to do:

- is.wholenumber returns TRUE or FALSE depending on whether its numeric argument x is an integer value (i.e., a whole number).
- is.integer returns TRUE or FALSE depending on whether its argument is of integer type, unless it is a factor when it returns FALSE.

See the documentation of is.integer for definition and details.
See Also

`is.integer` function of the R `base` package.

Other utility functions: `num_as_char()`, `num_as_ordinal()`

Examples

```r
is.wholenumber(1)  # is TRUE
is.wholenumber(1/2) # is FALSE
x <- seq(1, 2, by = 0.5)
is.wholenumber(x)
```

---

133t_rul35

133t_rul35 provides rules for translating into leet/l33t slang.

Description

133t_rul35 specifies rules for translating characters into other characters (typically symbols) to mimic leet/l33t slang (as a named character vector).

Usage

133t_rul35

Format

An object of class character of length 13.

Details

Old (i.e., to be replaced) characters are `paste(names(133t_rul35), collapse = "")`.

New (i.e., replaced) characters are `paste(133t_rul35, collapse = "")`.


See Also

Other text functions: `capitalize()`, `caseflip()`, `count_char()`, `read_ascii()`, `transl33t()`
**make_grid**  
*Generate a grid of x-y coordinates.*

**Description**

`make_grid` generates a grid of x/y coordinates and returns it (as a data frame).

**Usage**

```r
make_grid(x_min = 0, x_max = 2, y_min = 0, y_max = 1)
```

**Arguments**

- **x_min**: Minimum x coordinate. Default: `x_min = 0`.
- **x_max**: Maximum x coordinate. Default: `x_max = 2`.
- **y_min**: Minimum y coordinate. Default: `y_min = 0`.
- **y_max**: Maximum y coordinate. Default: `y_max = 1`.

**Examples**

```r
make_grid()
make_grid(x_min = -3, x_max = 3, y_min = -2, y_max = 2)
```

---

**num_as_char**  
*Convert a number into a character sequence.*

**Description**

`num_as_char` converts a number into a character sequence (of a specific length).

**Usage**

```r
num_as_char(x, n_pre_dec = 2, n_dec = 2, sym = "0", sep = ".")
```

**Arguments**

- **x**: Number(s) to convert (required, accepts numeric vectors).
- **n_pre_dec**: Number of digits before the decimal separator. Default: `n_pre_dec = 2`. This value is used to add zeros to the front of numbers. If the number of meaningful digits prior to decimal separator is greater than `n_pre_dec`, this value is ignored.
- **n_dec**: Number of digits after the decimal separator. Default: `n_dec = 2`.
- **sym**: Symbol to add to front or back. Default: `sym = 0`. Using `sym = " "` or `sym = "_"` can make sense, digits other than "0" do not.
- **sep**: Decimal separator to use. Default: `sep = "."`.
**num_as_char**

**Details**

The arguments `n_pre_dec` and `n_dec` set a number of desired digits before and after the decimal separator `sep`. `num_as_char` tries to meet these digit numbers by adding zeros to the front and end of `x`.

**Caveat:** Note that this function illustrates how numbers, characters, for loops, and `paste()` can be combined when writing functions. It is not written efficiently or well.

**See Also**

Other utility functions: `is.wholenumber()`, `num_as_ordinal()`

**Examples**

```r
num_as_char(1)
um_as_char(10/3)
um_as_char(1000/6)
```

# rounding down:
```r
num_as_char(1.3333), n_pre_dec = 0, n_dec = 0
num_as_char(1.3333), n_pre_dec = 2, n_dec = 0
num_as_char(1.3333), n_pre_dec = 2, n_dec = 1
```

# rounding up:
```r
num_as_char(1.6666, n_pre_dec = 1, n_dec = 0)
num_as_char(1.6666, n_pre_dec = 1, n_dec = 1)
num_as_char(1.6666, n_pre_dec = 2, n_dec = 2)
num_as_char(1.6666, n_pre_dec = 2, n_dec = 3)
```

# Note: If `n_pre_dec` is too small, actual number is used:
```r
num_as_char(11.33, n_pre_dec = 0, n_dec = 1)
num_as_char(11.66, n_pre_dec = 1, n_dec = 1)
```

# Details:
```r
num_as_char(1, sep = ",")
num_as_char(2, sym = " ")
num_as_char(3, sym = ",", n_dec = 0)
```

# Beware of bad inputs:
```r
num_as_char(4, sym = "8")
um_as_char(5, sym = "99")
```

# Works for vectors:
```r
num_as_char(1:10/1, n_pre_dec = 1, n_dec = 1)
num_as_char(1:10/3, n_pre_dec = 2, n_dec = 2)
```
num_as_ordinal  

Convert a number into an ordinal character sequence.

Description

num_as_ordinal converts a given (cardinal) number into an ordinal character sequence.

Usage

num_as_ordinal(x, sep = "")

Arguments

x  Number(s) to convert (required, accepts numeric vectors).
sep  Decimal separator to use. Default: sep = "" (i.e., no separator).

Details

The function currently only works for the English language and does not accepts inputs that are characters, dates, or times.

Note that the toOrdinal() function of the toOrdinal package works for multiple languages and provides a toOrdinalDate() function.

Caveat: Note that this function illustrates how numbers, characters, for loops, and paste() can be combined when writing functions. It is not written efficiently or well.

See Also

toOrdinal() function of the toOrdinal package.

Other utility functions: is.wholenumber(), num_as_char()

Examples

num_as_ordinal(1:4)
num_as_ordinal(10:14)  # all with "th"
num_as_ordinal(110:114)  # all with "th"
num_as_ordinal(120:124)  # 4 different suffixes
num_as_ordinal(1:15, sep = "-")  # using sep

# Note special cases:
num_as_ordinal(NA)
num_as_ordinal("1")
num_as_ordinal(Sys.Date())
num_as_ordinal(Sys.time())
num_as_ordinal(seq(1.99, 2.14, by = .01))
outliers

Outlier data.

Description

outliers is a fictitious dataset containing the id, sex, and height of 1000 non-existing, but otherwise normal people.

Usage

outliers

Format

A table with 100 cases (rows) and 3 variables (columns).

Details

Codebook

id  Participant ID (as character code)
sex  Gender (female vs. male)
height  Height (in cm)

Source

See CSV data at http://repository.com/ds4psy/data/out.csv.

See Also

Other datasets: data_1, data_2, data_t1_de, data_t1_tab, data_t1, data_t2, data_t3, data_t4, exp_wide, falsePosPsy_all, fame, pi_100k, posPsy_AHI_CESD, posPsy_long, posPsy_p_info, posPsy_wide, t3, t4, t_1, t_2, t_3, t_4, table6, table7, table8, tb

pal_ds4psy

ds4psy default color palette.

Description

pal_ds4psy provides a dedicated color palette.

Usage

pal_ds4psy
pal_n_sq

Format
An object of class data.frame with 1 rows and 11 columns.

Details
By default, pal_ds4psy is based on pal_unikn of the unikn package.

See Also
Other color objects and functions: pal_n_sq()

pal_n_sq(n = "all", pal = pal_ds4psy)

Arguments
n
The desired number colors of pal (as a number) or the character string "all" (to get all colors of pal). Default: n = "all".

pal
A color palette (as a data frame). Default: pal = pal_ds4psy.

Details
Use the more specialized function unikn::usecol for choosing n dedicated colors of a known color palette.

See Also
plot_tiles to plot tile plots.

Other color objects and functions: pal_ds4psy

Examples
pal_n_sq(1)  # 1 color: seeblau3
pal_n_sq(2)  # 4 colors
pal_n_sq(3)  # 9 colors (5: white)
pal_n_sq(4)  # 11 colors (6: white)
**pi_100k**

Data: 100k digits of pi.

**Description**

pi_100k is a dataset containing the first 100k digits of pi.

**Usage**

pi_100k

**Format**

A character of nchar(pi_100k) = 100001.

**Source**

See TXT data at [http://rpository.com/ds4psy/data/pi_100k.txt](http://rpository.com/ds4psy/data/pi_100k.txt).

Original data at [http://www.geom.uiuc.edu/~huberty/math5337/groupe/digits.html](http://www.geom.uiuc.edu/~huberty/math5337/groupe/digits.html).

**See Also**

Other datasets: data_1, data_2, data_t1_de, data_t1_tab, data_t1, data_t2, data_t3, data_t4, exp_wide, falsePosPsy_all, fame, outliers, posPsy_AHI_CESD, posPsy_long, posPsy_p_info, posPsy_wide, t3, t4, t_1, t_2, t_3, t_4, table6, table7, table8, tb

---

**plot_fn**

A function to plot a plot.

**Description**

plot_fn is a function that uses parameters for plotting a plot.

**Usage**

plot_fn(
    x = NA,
    y = 1,
    A = TRUE,
    B = FALSE,
    C = TRUE,
    D = FALSE,
    E = FALSE,
    F = FALSE,
    f = c(rev(pal_seeblau), "white", pal_pinky),
    g = "white"
)
Arguments

- **x**  
  A (natural) number. Default: \(x = NA\).

- **y**  
  A (decimal) number. Default: \(y = 1\).

- **A**  
  Boolean. Default: \(A = \text{TRUE}\).

- **B**  
  Boolean. Default: \(B = \text{FALSE}\).

- **C**  
  Boolean. Default: \(C = \text{TRUE}\).

- **D**  
  Boolean. Default: \(D = \text{FALSE}\).

- **E**  
  Boolean. Default: \(E = \text{FALSE}\).

- **F**  
  Boolean. Default: \(F = \text{FALSE}\).

- **f**  
  A color palette (e.g., as a vector). Default: \(f = \text{c(rev(pal_seeblau)},"white","pal_pinky\)).
  Note: Using colors of the *unikn* package by default.

- **g**  
  A color (e.g., as a character). Default: \(g = "\text{white}"\).

Details

- *plot_fn* is deliberately kept cryptic and obscure to illustrate how function parameters can be explored.

- *plot_fn* also shows that brevity in argument names should not come at the expense of clarity. In fact, transparent argument names are absolutely essential for understanding and using a function.

- *plot_fn* currently requires *pal_seeblau* and *pal_pinky* (from the *unikn* package) for its default colors.

See Also

- *plot_fun* for a related function; *pal_ds4psy* for color palette.


Examples

```r
# Basics:
plot_fn()
```

```r
# Exploring options:
plot_fn(x = 2, A = \text{TRUE})
plot_fn(x = 3, A = \text{FALSE}, E = \text{TRUE})
plot_fn(x = 4, A = \text{TRUE}, B = \text{TRUE}, D = \text{TRUE})
plot_fn(x = 5, A = \text{FALSE}, B = \text{TRUE}, E = \text{TRUE}, f = \text{c("black", "white", "gold")})
plot_fn(x = 7, A = \text{TRUE}, B = \text{TRUE}, F = \text{TRUE}, f = \text{c("steelblue", "white", "forestgreen")})
```
plot_fun

Another function to plot some plot.

Description

plot_fun is a function that provides options for plotting a plot.

Usage

```r
plot_fun(
  a = NA,
  b = TRUE,
  c = TRUE,
  d = 1,
  e = FALSE,
  f = FALSE,
  g = FALSE,
  c1 = c(rev(pal_seeblau), "white", pal_grau, "black", Bordeaux),
  c2 = "black"
)
```

Arguments

- `a` A (natural) number. Default: `a = NA`.
- `b` Boolean. Default: `b = TRUE`.
- `d` A (decimal) number. Default: `d = 1.0`.
- `e` Boolean. Default: `e = FALSE`.
- `g` Boolean. Default: `g = FALSE`.
- `c1` A color palette (e.g., as a vector). Default: `c1 = c(rev(pal_seeblau), "white", pal_grau, "black", Bordeaux)`.
- `c2` A color (e.g., as a character). Default: `c2 = "black"`.

Details

plot_fun is deliberately kept cryptic and obscure to illustrate how function parameters can be explored.

plot_fun also shows that brevity in argument names should not come at the expense of clarity. In fact, transparent argument names are absolutely essential for understanding and using a function.

plot_fun currently requires `pal_seeblau`, `pal_grau`, and `Bordeaux` (from the `unikn` package) for its default colors.
See Also

`plot_fn` for a related function; `pal_ds4psy` for color palette.

Other plot functions: `plot_fn()`, `plot_n()`, `plot_text()`, `plot_tiles()`, `theme_ds4psy()`

Examples

# Basics:
plot_fun()

# Exploring options:
plot_fun(a = 3, b = FALSE, e = TRUE)
plot_fun(a = 4, f = TRUE, g = TRUE, c1 = c("steelblue", "white", "firebrick"))

---

**plot_n**  
*Plot n tiles.*

Description

`plot_n` plots a row or column of n tiles on fixed or polar coordinates.

Usage

``` r
plot_n(
  n = NA,
  row = TRUE,
  polar = FALSE,
  pal = pal_ds4psy,
  sort = TRUE,
  borders = TRUE,
  border_col = "black",
  border_size = 0,
  lbl_tiles = FALSE,
  lbl_title = FALSE,
  rseed = NA,
  save = FALSE,
  save_path = "images/tiles",
  prefix = "",
  suffix = ""
)
```

Arguments

- **n**  
  Basic number of tiles (on either side).

- **row**  
  Plot as a row? Default: `row = TRUE` (else plotted as a column).

- **polar**  
  Plot on polar coordinates? Default: `polar = FALSE` (i.e., using fixed coordinates).
### plot_n

- **pal**: A color palette (automatically extended to \( n \) colors). Default: `pal = pal_ds4psy`.
- **sort**: Sort tiles? Default: `sort = TRUE` (i.e., sorted tiles).
- **borders**: Add borders to tiles? Default: `borders = TRUE` (i.e., use borders).
- **border_col**: Color of borders (if `borders = TRUE`). Default: `border_col = “black”`.
- **border_size**: Size of borders (if `borders = TRUE`). Default: `border_size = 0` (i.e., invisible).
- **lbl_tiles**: Add numeric labels to tiles? Default: `lbl_tiles = FALSE` (i.e., no labels).
- **lbl_title**: Add numeric label (of \( n \)) to plot? Default: `lbl_title = FALSE` (i.e., no title).
- **rseed**: Random seed (number). Default: `rseed = NA` (using random seed).
- **save**: Save plot as png file? Default: `save = FALSE`.
- **save_path**: Path to save plot (if `save = TRUE`). Default: `save_path = “images/tiles”`.
- **prefix**: Prefix to plot name (if `save = TRUE`). Default: `prefix = “”`.
- **suffix**: Suffix to plot name (if `save = TRUE`). Default: `suffix = “”`.

### Details

Note that a polar row makes a tasty pie, whereas a polar column makes a target plot.

### See Also

- **pal_ds4psy** for default color palette.
- Other plot functions: `plot_fn()`, `plot_fun()`, `plot_text()`, `plot_tiles()`, `theme_ds4psy()`

### Examples

#### (1) Basics (as ROW or COL):

- `plot_n()` # default plot (random \( n \), row = TRUE, with borders, no labels)
- `plot_n(row = FALSE)` # default plot (random \( n \), with borders, no labels)
- `plot_n(n = 4, sort = FALSE)` # random order
- `plot_n(n = 6, borders = FALSE)` # no borders
- `plot_n(n = 8, lbl_tiles = TRUE, lbl_title = TRUE)` # with tile +
  - `lbl_title = TRUE)` # title labels

#### Set colors:

- `plot_n(n = 5, row = FALSE, pal = c("orange", "white", "firebrick"),
  lbl_tiles = TRUE, lbl_title = TRUE, sort = TRUE)`
- `plot_n(n = 6, sort = FALSE, border_col = "white", border_size = 2)`

#### Fixed rseed:

- `plot_n(n = 4, sort = FALSE, borders = FALSE,
  lbl_tiles = TRUE, lbl_title = TRUE, rseed = 101)`

#### (2) polar plot (as PIE or TARGET):

- `plot_n(polar = TRUE)` # PIE plot (with borders, no labels)
- `plot_n(polar = TRUE, row = FALSE)` # TARGET plot (with borders, no labels)
plot_text

Plot text characters (from file or user input).

Description

plot_text parses text (from a file or from user input in Console) into a table and then plots all its characters as a tile plot (using ggplot2).

Usage

plot_text(
  file = "", 
  char_bg = " ", 
  lbl_tiles = TRUE, 
  lbl_rotate = FALSE, 
  cex = 3, 
  fontface = 1, 
  family = "sans", 
  col_lbl = "black", 
  col_bg = "white", 
  pal = pal_ds4psy[1:5], 
  pal_extend = TRUE, 
  case_sense = FALSE, 
  borders = TRUE, 
  border_col = "white", 
  border_size = 0.5 
)

Arguments

file The text file to read (or its path). If file = "" (the default), scan is used to read user input from the Console. If a text file is stored in a sub-directory, enter its path and name here (without any leading or trailing "." or "/"). Default: file = "".
plot_text

char_bg  Character used as background. Default: char_bg = " ". If char_bg = NA, the most frequent character is used.

lbl_tiles  Add character labels to tiles? Default: lbl_tiles = TRUE (i.e., show labels).

lbl_rotate  Rotate character labels? Default: lbl_rotate = FALSE (i.e., no rotation).

cex  Character size (numeric). Default: cex = 3.

fontface  Font face of text labels (numeric). Default: fontface = 1, (from 1 to 4).

family  Font family of text labels (name). Default: family = "sans". Alternative options: "sans", "serif", or "mono".

col_lbl  Color of text labels. Default: col_lbl = "black" (if lbl_tiles = TRUE).

col_bg  Color of char_bg (if defined), or the most frequent character in text (typically " "). Default: col_bg = "white".

col  Color palette for filling tiles of text (used in order of character frequency). Default: pal = pal_ds4psy[1:5] (i.e., shades of unikn::Seeblau).

pal_extend  Boolean: Should pal be extended to match the number of different characters in text? Default: pal_extend = TRUE. If pal_extend = FALSE, only the tiles of the length(pal) most frequent characters will be filled by the colors of pal.

case_sense  Boolean: Should lower- and uppercase characters be distinguished? Default: case_sense = FALSE.

borders  Boolean: Add borders to tiles? Default: borders = TRUE (i.e., use borders).

border_col  Color of borders (if borders = TRUE). Default: border_col = "white".

border_size  Size of borders (if borders = TRUE). Default: border_size = 0.5.

See Also

read_ascii for reading text into a table; pal_ds4psy for default color palette.

Other plot functions: plot_fn(), plot_fun(), plot_n(), plot_tiles(), theme_ds4psy()

Examples

## Create a temporary file "test.txt":
# cat("Hello world!", "This is a test.",
# "Can you see this text?",
# "Good! Please carry on...",
# file = "test.txt", sep = "\n")

## (a) Plot text (from file):
# plot_text("test.txt")

## Set colors, pal_extend, and case_sense:
# cols <- c("steelblue", "skyblue", "lightgrey")
# cols <- c("firebrick", "olivedrab", "steelblue", "orange", "gold")
# plot_text("test.txt", pal = cols, pal_extend = TRUE)
# plot_text("test.txt", pal = cols, pal_extend = FALSE)
# plot_text("test.txt", pal = cols, pal_extend = FALSE, case_sense = TRUE)

## Customize text and grid options:
plot_tiles

Plot n-by-n tiles.

Description

plot_tiles plots an area of n-by-n tiles on fixed or polar coordinates.

Usage

plot_tiles(
  n = NA,
  pal = pal_ds4psy,
  sort = TRUE,
  borders = TRUE,
  border_col = "black",
  border_size = 0.2,
  lbl_tiles = FALSE,
  lbl_title = FALSE,
  polar = FALSE,
  rseed = NA,
  save = FALSE,
  save_path = "images/tiles",
  prefix = "",
  suffix = ""
)
plot_tiles

Arguments

- **n**: Basic number of tiles (on either side).
- **pal**: Color palette (automatically extended to \(n \times n\) colors). Default: \(\text{pal} = \text{pal_ds4psy}\).
- **sort**: Boolean: Sort tiles? Default: \(\text{sort} = \text{TRUE}\) (i.e., sorted tiles).
- **borders**: Boolean: Add borders to tiles? Default: \(\text{borders} = \text{TRUE}\) (i.e., use borders).
- **border_col**: Color of borders (if \(\text{borders} = \text{TRUE}\)). Default: \(\text{border_col} = \text{"black"}\).
- **border_size**: Size of borders (if \(\text{borders} = \text{TRUE}\)). Default: \(\text{border_size} = 0.2\).
- **lbl_tiles**: Boolean: Add numeric labels to tiles? Default: \(\text{lbl_tiles} = \text{FALSE}\) (i.e., no labels).
- **lbl_title**: Boolean: Add numeric label (of \(n\)) to plot? Default: \(\text{lbl_title} = \text{FALSE}\) (i.e., no title).
- **polar**: Boolean: Plot on polar coordinates? Default: \(\text{polar} = \text{FALSE}\) (i.e., using fixed coordinates).
- **rseed**: Random seed (number). Default: \(\text{rseed} = \text{NA}\) (using random seed).
- **save**: Boolean: Save plot as png file? Default: \(\text{save} = \text{FALSE}\).
- **save_path**: Path to save plot (if \(\text{save} = \text{TRUE}\)). Default: \(\text{save_path} = \text{"images/tiles"}\).
- **prefix**: Prefix to plot name (if \(\text{save} = \text{TRUE}\)). Default: \(\text{prefix} = \text{""}\).
- **suffix**: Suffix to plot name (if \(\text{save} = \text{TRUE}\)). Default: \(\text{suffix} = \text{""}\).

See Also

- **pal_ds4psy** for default color palette.
- Other plot functions: \(\text{plot_fn()}, \text{plot_fun()}, \text{plot_n()}, \text{plot_text()}, \text{theme_ds4psy()}\)

Examples

# (1) Tile plot:
plot_tiles() # default plot (random n, with borders, no labels)

plot_tiles(n = 4, sort = \text{FALSE}) # random order
plot_tiles(n = 6, borders = \text{FALSE}) # no borders
plot_tiles(n = 8, lbl_tiles = \text{TRUE}, # with tile +
lbl_title = \text{TRUE}) # title labels

# Set colors:
plot_tiles(n = 4, pal = \text{c("orange", "white", "firebrick")},
lbl_tiles = \text{TRUE}, lbl_title = \text{TRUE},
sort = \text{TRUE})
plot_tiles(n = 6, sort = \text{FALSE}, border_col = \text{"white"}, border_size = 2)

# Fixed rseed:
plot_tiles(n = 4, sort = \text{FALSE}, borders = \text{FALSE},
lbl_tiles = \text{TRUE}, lbl_title = \text{TRUE},
rseed = 101)

# (2) polar plot:
posPsy_AHI_CESD

plot_tiles(polar = TRUE)  # default polar plot (with borders, no labels)

plot_tiles(n = 4, polar = TRUE, sort = FALSE)  # random order
plot_tiles(n = 6, polar = TRUE, sort = TRUE,  # sorted and with
  lbl_tiles = TRUE, lbl_title = TRUE)  # tile + title labels
plot_tiles(n = 4, sort = FALSE, borders = TRUE,
  border_col = "white", border_size = 2,
  polar = TRUE, rseed = 132)  # fixed rseed

---

posPsy_AHI_CESD  Positive Psychology: AHI CESD data.

Description

posPsy_AHI_CESD is a dataset containing answers to the 24 items of the Authentic Happiness Inventory (AHI) and answers to the 20 items of the Center for Epidemiological Studies Depression (CES-D) scale (Radloff, 1977) for multiple (1 to 6) measurement occasions.

Usage

posPsy_AHI_CESD

Format

A table with 992 cases (rows) and 50 variables (columns).

Details

Codebook

- 1. id: Participant ID.
- 2. occasion: Measurement occasion: 0: Pretest (i.e., at enrolment), 1: Posttest (i.e., 7 days after pretest), 2: 1-week follow-up, (i.e., 14 days after pretest), 7 days after posttest), 3: 1-month follow-up, (i.e., 38 days after pretest, 31 days after posttest), 4: 3-month follow-up, (i.e., 98 days after pretest, 91 days after posttest), 5: 6-month follow-up, (i.e., 189 days after pretest, 182 days after posttest).
- 3. elapsed.days: Time since enrolment measured in fractional days.
- 4. intervention: Type of intervention: 3 positive psychology interventions (PPIs), plus 1 control condition: 1: "Using signature strengths", 2: "Three good things", 3: "Gratitude visit", 4: "Recording early memories" (control condition).
- 5.-28. (from ahi01 to ahi24): Responses on 24 AHI items.
- 29.-48. (from cesd01 to cesd20): Responses on 20 CES-D items.
- 49. ahiTotal: Total AHI score.
- 50. cesdTotal: Total CES-D score.

See codebook and references at https://bookdown.org/hneth/ds4psy/B-1-datasets-pos.html.
posPsy_long

Source

Articles


See [https://openpsychologydata.metajnl.com/articles/10.5334/jopd.35/](https://openpsychologydata.metajnl.com/articles/10.5334/jopd.35/) for details and [https://doi.org/10.6084/m9.figshare.1577563.v1](https://doi.org/10.6084/m9.figshare.1577563.v1) for original dataset.

Additional references at [https://bookdown.org/hneth/ds4psy/B-1-datasets-pos.html](https://bookdown.org/hneth/ds4psy/B-1-datasets-pos.html).

See Also

posPsy_long for a corrected version of this file (in long format).

Other datasets: data_1, data_2, data_t1_de, data_t1_tab, data_t1, data_t2, data_t3, data_t4, exp_wide, falsePosPsy_all, fame, outliers, pi_100k, posPsy_long, posPsy_p_info, posPsy_wide, t3, t4, t_1, t_2, t_3, t_4, table6, table7, table8, tb

---

posPsy_long  Positive Psychology: AHI CESD corrected data (in long format).

Description

posPsy_long is a dataset containing answers to the 24 items of the Authentic Happiness Inventory (AHI) and answers to the 20 items of the Center for Epidemiological Studies Depression (CES-D) scale (see Radloff, 1977) for multiple (1 to 6) measurement occasions.

Usage

posPsy_long

Format

A table with 990 cases (rows) and 50 variables (columns).

Details

This dataset is a corrected version of posPsy_AHI_CESD and in long-format.
Source

Articles


See Also

*posPsy_AHI_CESD* for source of this file and codebook information; *posPsy_wide* for a version of this file (in wide format).

Other datasets: *data_1, data_2, data_t1_de, data_t1_tab, data_t1, data_t2, data_t3, data_t4, exp_wide, falsePosPsy_all, fame, outliers, pi_100k, posPsy_AHI_CESD, posPsy_p_info, posPsy_wide, t3, t4, t_1, t_2, t_3, t_4, table6, table7, table8, tb*

---

**posPsy_p_info** Positive Psychology: Participant data.

Description

*posPsy_p_info* is a dataset containing details of 295 participants.

Usage

**posPsy_p_info**

Format

A table with 295 cases (rows) and 6 variables (columns).

Details

**id** Participant ID.

**intervention** Type of intervention: 3 positive psychology interventions (PPIs), plus 1 control condition: 1: "Using signature strengths", 2: "Three good things", 3: "Gratitude visit", 4: "Recording early memories" (control condition).

**sex** Sex: 1 = female, 2 = male.

**age** Age (in years).

**educ** Education level: Scale from 1: less than 12 years, to 5: postgraduate degree.

**income** Income: Scale from 1: below average, to 3: above average.

See codebook and references at https://bookdown.org/hneth/ds4psy/B-1-datasets-pos.html.
Source

Articles


See Also

Other datasets: data_1, data_2, data_t1_de, data_t1_tab, data_t1, data_t2, data_t3, data_t4, exp_wide, falsePosPsy_all, fame, outliers, pi_100k, posPsy_AHI_CESD, posPsy_long, posPsy_wide, t3, t4, t_1, t_2, t_3, t_4, table6, table7, table8, tb

### posPsy_wide

**Positive Psychology: All corrected data (in wide format).**

Description

`posPsy_wide` is a dataset containing answers to the 24 items of the Authentic Happiness Inventory (AHI) and answers to the 20 items of the Center for Epidemiological Studies Depression (CES-D) scale (see Radloff, 1977) for multiple (1 to 6) measurement occasions.

Usage

`posPsy_wide`

Format

An object of class `spec_tbl_df` (inherits from `tbl_df`, `tbl`, `data.frame`) with 295 rows and 294 columns.

Details

This dataset is based on `posPsy_AHI_CESD` and `posPsy_long`, but is in wide format.
**Source**

**Articles**


**See Also**

- `posPsy_AHI_CESD` for the source of this file, `posPsy_long` for a version of this file (in long format).

Other datasets: `data_1`, `data_2`, `data_t1_de`, `data_t1_tab`, `data_t1`, `data_t2`, `data_t3`, `data_t4`, `exp_wide`, `falsePosPsy_all`, `fame`, `outliers`, `pi_100k`, `posPsy_AHI_CESD`, `posPsy_long`, `posPsy_p_info`, `t3`, `t4`, `t_1`, `t_2`, `t_3`, `t_4`, `table6`, `table7`, `table8`, `tb`
Details

read_ascii creates a data frame with 3 variables: Each character's x- and y-coordinates (from top to bottom) and a variable char for the character at this coordinate.

The getwd function is used to determine the current working directory. This replaces the here package, which was previously used to determine an (absolute) file path.

See Also

plot_text for a corresponding plot function.

Other text functions: capitalize(), caseflip(), count_char(), l33t_rul35, transl33t()

Examples

```r
## Create a temporary file "test.txt":
# cat("Hello world!", "This is a test.",
#     "Can you see this text?",
#     "Good! Please carry on...",
#     file = "test.txt", sep = "\n")

## (a) Read text (from file):
# read_ascii("test.txt")
# read_ascii("test.txt", flip_y = TRUE) # y flipped

## (b) Read text (from file in subdir):
# read_ascii("data-raw/txt/ascii.txt") # requires txt file

## (c) Scan user input (from console):
# read_ascii()
```

sample_dates

*Draw a sample of n random dates (from a given range).*

Description

sample_dates draws a sample of n random dates from a given range.

Usage

```r
sample_dates(n = 1, from = "1970-01-01", to = Sys.Date())
```
sample_times

Arguments

n Number dates to draw. Default: n = 1.
from Earliest date (as string). Default: from = "1970-01-01".
to Latest date (as string). Default: to = Sys.Date().

Details

By default, sample_dates draws n = 1 random date in the range from = "1970-01-01" to = Sys.Date() (current date).

See Also

Other random functions: coin(), dice_2(), dice(), sample_times()

Examples

sample_dates()
sort(sample_dates(n = 10))
sort(sample_dates(n = 10, from = "2020-02-28", to = "2020-03-01")) # 2020 is a leap year

# Note: Oddity with sample():
sort(sample_dates(n = 10, from = "2020-01-01", to = "2020-01-01")) # range of 0!
# see sample(9:9, size = 10, replace = TRUE)

sample_times

Draw a sample of n random times (from a given range).

Description

sample_times draws a sample of n random times from a given range.

Usage

sample_times(n = 1, from = "1970-01-01 00:00:00", to = Sys.time())

Arguments

n Number dates to draw. Default: n = 1.
from Earliest date (as string). Default: from = "1970-01-01 00:00:00".
to Latest date (as string). Default: to = Sys.time().

Details

By default, sample_times draws n = 1 random time in the range from = "1970-01-01 00:00:00" to = Sys.time() (current time).
See Also

Other random functions: `coin()`, `dice_2()`, `dice()`, `sample_dates()`

Examples

```r
# Basics:
sample_times()
sample_times(n = 10)

# Specific ranges:
sort(sample_times(n = 10, from = (Sys.time() - 60)))  # within the last minute
sort(sample_times(n = 10, from = (Sys.time() - 1 * 60 * 60)))  # within the last hour
sort(sample_times(n = 10, from = Sys.time(),
                    to = (Sys.time() + 1 * 60 * 60)))  # within the next hour
sort(sample_times(n = 10, from = "2020-01-01 00:00:00 CET",
                    to = "2020-01-01 00:00:01 CET"))  # within 1 sec range

# Note: Oddity with sample():
sort(sample_times(n = 10, from = "2020-01-01 00:00:00 CET",
                    to = "2020-01-01 00:00:00 CET"))  # range of 0!
# see sample(9:9, size = 10, replace = TRUE)
```

---

t3  
Data table t3.

Description

t3 is a fictitious dataset to practice importing and joining data (from a CSV file).

Usage

t3

Format

A table with 10 cases (rows) and 4 variables (columns).

Source


See Also

Other datasets: `data_1`, `data_2`, `data_t1_de`, `data_t1_tab`, `data_t1`, `data_t2`, `data_t3`, `data_t4`, `exp_wide`, `falsePosPsy_all`, `fame`, `outliers`, `pi_100k`, `posPsy_AHI_CESD`, `posPsy_long`, `posPsy_p_info`, `posPsy_wide`, `t4`, `t_1`, `t_2`, `t_3`, `t_4`, `table6`, `table7`, `table8`, `tb`
Data table t4.

Description

t4 is a fictitious dataset to practice importing and joining data (from a CSV file).

Usage

t4

Format

A table with 10 cases (rows) and 4 variables (columns).

Source


See Also

Other datasets: data_1, data_2, data_t1_de, data_t1_tab, data_t1, data_t2, data_t3, data_t4, exp_wide, falsePosPsy_all, fame, outliers, pi_100k, posPsy_AHI_CESD, posPsy_long, posPsy_p_info, posPsy_wide, t3, t_1, t_2, t_3, t_4, table6, table7, table8, tb

Data table6.

Description

table6 is a fictitious dataset to practice tidying data.

Usage

table6

Format

A table with 6 cases (rows) and 2 variables (columns).

Details

This dataset is a variant of the table1 to table5 datasets of the tidyr package.

Source

See Also

Other datasets: data_1, data_2, data_t1_de, data_t1_tab, data_t1, data_t2, data_t3, data_t4, exp_wide, falsePosPsy_all, fame, outliers, pi_100k, posPsy_AHI_CESD, posPsy_long, posPsy_p_info, posPsy_wide, t3, t4, t_1, t_2, t_3, t_4, table7, table8, tb

---

**table7**

**Description**

*table7* is a fictitious dataset to practice tidying data.

**Usage**

table7

**Format**

A table with 6 cases (rows) and 1 (horrendous) variable (column).

**Details**

This dataset is a variant of the *table1* to *table5* datasets of the *tidyr* package.

**Source**


---

**table8**

**Description**

*table8* is a fictitious dataset to practice tidying data.

**Usage**

table8
Format

A table with 3 cases (rows) and 5 variables (columns).

Details

This dataset is a variant of the table1 to table5 datasets of the tidyr package.

Source


See Also

Other datasets: data_1, data_2, data_t1_de, data_t1_tab, data_t1, data_t2, data_t3, data_t4, exp_wide, falsePosPsy_all, fame, outliers, pi_100k, posPsy_AHI_CESD, posPsy_long, posPsy_p_info, posPsy_wide, t3, t4, t_1, t_2, t_3, t_4, table6, table7, tb

tb

Data table tb.

Description

tb is a fictitious dataset describing 100 non-existing, but otherwise ordinary people.

Usage

tb

Format

A table with 100 cases (rows) and 5 variables (columns).

Details

Codebook

- 1. id: Participant ID.
- 2. age: Age (in years).
- 3. height: Height (in cm).
- 4. shoesize: Shoesize (EU standard).
- 5. IQ: IQ score (according Raven’s Regressive Tables).

tb was orginally created to practice loops and iterations (as a CSV file).

Source

theme_ds4psy

See Also

Other datasets: data_1, data_2, data_t1_de, data_t1_tab, data_t1, data_t2, data_t3, data_t4, exp_wide, falsePosPsy_all, fame, outliers, pi_100k, posPsy_AHI_CESD, posPsy_long, posPsy_p_info, posPsy_wide, t3, t4, t_1, t_2, t_3, t_4, table6, table7, table8

theme_ds4psy

Description

theme_ds4psy provides a basic ds4psy theme to use in ggplot2 commands.

Usage

theme_ds4psy(
  col_title = "black",
  base_size = 11,
  base_family = "",
  base_line_size = base_size/20,
  base_rect_size = base_size/20
)

Arguments

col_title      Color of title (text) elements (optional, numeric). Default: col_title = "black". Consider using col_title = unikn::pal_seeblau[[4]] in combination with black or grey data points.
base_size      Base font size (optional, numeric). Default: base_size = 11.
base_family    Base font family (optional, character). Default: base_family = "".
base_line_size Base line size (optional, numeric). Default: base_line_size = base_size/20.

Details

The theme is lightweight and no-nonsense, but somewhat opinionated (e.g., in using mostly grey scales to allow emphasizing data points with color accents).

See Also

unikn::theme_unikn for the source of the current theme.
Other plot functions: plot_fn(), plot_fun(), plot_n(), plot_text(), plot_tiles()
Examples

# Plotting iris dataset (using ggplot2 and theme_ds4psy):
library("ggplot2") # theme_ds4psy requires loading ggplot2

ggplot(datasets::iris) +
  geom_jitter(aes(x = Petal.Length, y = Petal.Width, color = Species), size = 3, alpha = 2/3) +
  labs(title = "Iris species",
       caption = "Data from datasets::iris") +
  theme_ds4psy(col_title = "black", base_size = 11)

transl33t # transl33t text into leet slang.

Description

transl33t translates text into leet (or l33t) slang given a set of rules.

Usage

transl33t(txt, rules = l33t_rul35, in_case = "no", out_case = "no")

Arguments

txt
  The text (character string) to translate.

rules
  Rules which existing character in txt is to be replaced by which new character
  (as a named character vector). Default: rules = l33t_rul35.

in_case
  Change case of input string txt. Default: in_case = "no". Set to "lo" or "up"
  for lower or uppercase, respectively.

out_case
  Change case of output string. Default: out_case = "no". Set to "lo" or "up"
  for lower or uppercase, respectively.

Details

The current version of transl33t only uses base R commands, rather than the stringr package.

See Also

l33t_rul35 for default rules.

Other text functions: capitalize(), caseflip(), count_char(), l33t_rul35, read_ascii()
**Examples**

```r
# Use defaults:
trans33t(txt = "hello world")
trans33t(txt = c(letters))
trans33t(txt = c(LETTERS))

# Specify rules:
trans33t(txt = "hello world",
    rules = c("e" = "3", "l" = "1", "o" = "0"))

# Set input and output case:
trans33t(txt = "hello world", in_case = "up",
    rules = c("e" = "3", "l" = "1", "o" = "0"))  # e only capitalized
trans33t(txt = "hEllo world", in_case = "lo", out_case = "up",
    rules = c("e" = "3", "l" = "1", "o" = "0"))  # e transl33ted
```

---

### t_1

**Data t_1.**

---

**Description**

`t_1` is a fictitious dataset to practice tidying data.

**Usage**

`t_1`

**Format**

A table with 8 cases (rows) and 9 variables (columns).

**Source**


**See Also**

Other datasets: `data_1`, `data_2`, `data_t1_de`, `data_t1_tab`, `data_t1`, `data_t2`, `data_t3`, `data_t4`, `exp_wide`, `falsePosPsy_all`, `fame`, `outliers`, `pi_100k`, `posPsy_AHI_CESD`, `posPsy_long`, `posPsy_p_info`, `posPsy_wide`, `t3`, `t4`, `t_2`, `t_3`, `t_4`, `table6`, `table7`, `table8`, `tb`
Description

t_2 is a fictitious dataset to practice tidying data.

Usage

t_2

Format

A table with 8 cases (rows) and 5 variables (columns).

Source

See CSV data at http://rpository.com/ds4psy/data/t_2.csv.

See Also

Other datasets: data_1, data_2, data_t1_de, data_t1_tab, data_t1, data_t2, data_t3, data_t4, exp_wide, falsePosPsy_all, fame, outliers, pi_100k, posPsy_AHI_CESD, posPsy_long, posPsy_p_info, posPsy_wide, t3, t4, t_1, t_3, t_4, table6, table7, table8, tb

Description

Data t_3.

t_3 is a fictitious dataset to practice tidying data.

Usage

t_3

Format

A table with 16 cases (rows) and 6 variables (columns).

Source

**t_4**

**See Also**

Other datasets: `data_1, data_2, data_t1_de, data_t1_tab, data_t1, data_t2, data_t3, data_t4, exp_wide, falsePosPsy_all, fame, outliers, pi_100k, posPsy_AHI_CESD, posPsy_long, posPsy_p_info, posPsy_wide, t3, t4, t_1, t_2, t_4, table6, table7, table8, tb`

---

**t_4**  
*Data t_4.*

---

**Description**

`t_4` is a fictitious dataset to practice tidying data.

**Usage**

`t_4`

**Format**

A table with 16 cases (rows) and 8 variables (columns).

**Source**


---

**what_date**  
*What date is it?*

---

**Description**

`what_date` provides a satisficing version of `Sys.Date()` that is sufficient for most purposes.

**Usage**

`what_date(when = NA, rev = FALSE, sep = "-", month_form = "m")`
Arguments

- **when**: Date (as a scalar or vector). Default: when = NA. Using as.Date(when) to convert strings into dates, and Sys.Date(), if when = NA.
- **rev**: Boolean: Reverse date (to Default: rev = FALSE).
- **sep**: Character: Separator to use. Default: sep = "-".
- **month_form**: Character: Month format. Default: month_form = "m" for numeric month (01-12). Use month_form = "b" for short month name and month_form = "B" for full month name (in current locale).

Details

what_date returns either a simple version of when or Sys.Date() (in using current system settings).

See Also

what_day() function to obtain (week)days; what_time() function to obtain times; cur_time() function to print the current time; cur_date() function to print the current date; now() function of the lubridate package; Sys.time() function of base R.

Other date and time functions: cur_date(), cur_time(), what_day(), what_month(), what_time(), what_week(), what_year()

Examples

```r
what_date()
what_date(sep = "/")
what_date(rev = TRUE)
what_date(rev = TRUE, sep = ".")
what_date(rev = TRUE, sep = " ", month_form = "B")

# with vector (of dates):
ds <- c("2020-01-15 01:02:03 CET", "2020-12-31 14:15:16")
what_date(ds)
what_date(ds, rev = TRUE, sep = ".")
what_date(ds, rev = TRUE, month_form = "b")
```

---

**what_day**

What day (of the week) is it?

Description

what_day provides a satisficing version of to determine the day of the week corresponding to a given date.

Usage

```r
what_day(when = Sys.time(), abbr = FALSE)
```
**what_month**

Arguments

when Date (as a scalar or vector). Default: when = NA. Using as.Date(when) to convert strings into dates, and Sys.Date(), if when = NA.

abbr Boolean: Return abbreviated? Default: abbr = FALSE.

Details

what_day returns the weekday of when or Sys.Date() (as a name).

See Also

what_date() function to obtain dates; what_time() function to obtain times; cur_time() function to print the current time; cur_date() function to print the current date; now() function of the lubridate package; Sys.time() function of base R.

Other date and time functions: cur_date(), cur_time(), what_date(), what_month(), what_time(), what_week(), what_year()

Examples

```r
what_day()
what_day(abbr = TRUE)

# Work with vectors (when as characters):
ds <- c("2020-01-01", "2020-02-29", "2020-12-24", "2020-12-31")
what_day(when = ds)
what_day(when = ds, abbr = TRUE)
```

Description

what_month provides a satisficing version of to determine the month corresponding to a given date.

Usage

```r
what_month(when = Sys.time(), abbr = FALSE, as_integer = FALSE)
```

Arguments

when Date (as a scalar or vector). Default: when = NA. Using as.Date(when) to convert strings into dates, and Sys.Date(), if when = NA.

abbr Boolean: Return abbreviated? Default: abbr = FALSE.

as_integer Boolean: Return as integer? Default: as_integer = FALSE.
what_month returns the month of when or Sys.Date() (as a name or number).

See Also

what_week() function to obtain weeks; what_date() function to obtain dates; cur_time() function to print the current time; cur_date() function to print the current date; now() function of the lubridate package; Sys.time() function of base R.

Other date and time functions: cur_date(), cur_time(), what_date(), what_day(), what_time(), what_week(), what_year()

Examples

```r
what_month()
what_month(abbr = TRUE)
what_month(as_integer = TRUE)

# Work with vectors (when as characters):
ds <- c("2020-01-01", "2020-02-29", "2020-12-24", "2020-12-31")
what_month(when = ds)
what_month(when = ds, abbr = TRUE, as_integer = FALSE)
what_month(when = ds, abbr = TRUE, as_integer = TRUE)
```

Description

what_time provides a satisficing version of Sys.time() that is sufficient for most purposes.

Usage

```r
what_time(when = NA, seconds = FALSE, sep = ":")
```

Arguments

- `when` Time (as a scalar or vector). Default: `when = NA`. Returning Sys.time(), if `when = NA`.
- `seconds` Boolean: Show time with seconds? Default: `seconds = FALSE`.

Details

what_time returns either a simple version of `when` or `Sys.time()` in " using current system settings.
See Also

cur_time() function to print the current time; cur_date() function to print the current date; now() function of the lubridate package; Sys.time() function of base R.

Other date and time functions: cur_date(), cur_time(), what_date(), what_day(), what_month(), what_week(), what_year()

Examples

what_time()

# with vector (of times):
ts <- c("2020-01-14 01:02:03 CET", "2020-12-31 14:15:16")
what_time(ts)
what_time(ts, seconds = TRUE, sep = " ")

what_week    What week is it?

Description

what_week provides a satisficing version of to determine the week corresponding to a given date.

Usage

what_week(when = Sys.time(), unit = "year", as_integer = FALSE)

Arguments

when Date (as a scalar or vector). Default: when = NA. Using as.Date(when) to convert strings into dates, and Sys.Date(), if when = NA.

unit Character: Unit of week? Possible values are "month", "year". Default: unit = "year" (for week within year).

as_integer Boolean: Return as integer? Default: as_integer = FALSE.

Details

what_week returns the week of when or Sys.Date() (as a name or number).

See Also

what_day() function to obtain (week)days; what_date() function to obtain dates; cur_time() function to print the current time; cur_date() function to print the current date; now() function of the lubridate package; Sys.time() function of base R.

Other date and time functions: cur_date(), cur_time(), what_date(), what_day(), what_month(), what_time(), what_year()
what_year

What year is it?

Description

what_year provides a satisficing version of to determine the year corresponding to a given date.

Usage

what_year(when = Sys.time(), abbr = FALSE, as_integer = FALSE)

Arguments

when Date (as a scalar or vector). Default: when = NA. Using as.Date(when) to convert strings into dates, and Sys.Date(), if when = NA.
abbr Boolean: Return abbreviated? Default: abbr = FALSE.
as_integer Boolean: Return as integer? Default: as_integer = FALSE.

Details

what_year returns the year of when or Sys.Date() (as a name or number).

See Also

what_week() function to obtain weeks; what_month() function to obtain months; cur_time() function to print the current time; cur_date() function to print the current date; now() function of the lubridate package; Sys.time() function of base R.

Other date and time functions: cur_date(), cur_time(), what_date(), what_day(), what_month(), what_time(), what_week()
what_year

Examples

what_year()
what_year(abbr = TRUE)
what_year(as_integer = TRUE)

# Work with vectors (when as characters):
ds <- c("2020-01-01", "2020-02-29", "2020-12-24", "2020-12-31")
what_year(when = ds)
what_year(when = ds, abbr = TRUE, as_integer = FALSE)
what_year(when = ds, abbr = TRUE, as_integer = TRUE)
Index

*Topic datasets
  data_1, 8
  data_2, 9
  data_t1, 9
  data_t1_de, 10
  data_t1_tab, 10
  data_t2, 11
  data_t3, 11
  data_t4, 12
  exp_wide, 15
  falsePosPsy_all, 16
  fame, 17
  l33t_rul35, 19
  outliers, 23
  pal_ds4psy, 23
  pi_100k, 25
  posPsy_AHI_CESD, 34
  posPsy_long, 35
  posPsy_p_info, 36
  posPsy_wide, 37
  t3, 41
  t4, 42
  t_1, 47
  t_2, 48
  t_3, 48
  t_4, 49
  table6, 42
  table7, 43
  table8, 43
  tb, 44

  capitalize, 3, 4, 6, 19, 39, 46
  caseflip, 3, 4, 6, 19, 39, 46
  coin, 5, 13, 14, 40, 41
  count_char, 3, 4, 6, 19, 39, 46
  cur_date, 6, 8, 50–54
  cur_time, 7, 7, 50–54

  data_1, 8, 9–12, 15, 17, 18, 23, 25, 35–38, 41–45, 47–49
  data_2, 8, 9, 10–12, 15, 17, 18, 23, 25, 35–38, 41–45, 47–49
  data_t1, 8, 9, 10–12, 15, 17, 18, 23, 25, 35–38, 41–45, 47–49
  data_t1_de, 8–10, 10, 11, 12, 15, 17, 18, 23, 25, 35–38, 41–45, 47–49
  data_t1_tab, 8–10, 10, 11, 12, 15, 17, 18, 23, 25, 35–38, 41–45, 47–49
  data_t2, 8–11, 11, 12, 15, 17, 18, 23, 25, 35–38, 41–45, 47–49
  data_t3, 8–11, 11, 12, 15, 17, 18, 23, 25, 35–38, 41–45, 47–49
  data_t4, 8–12, 12, 15, 17, 18, 23, 25, 35–38, 41–45, 47–49
  dice, 5, 13, 14, 40, 41
  dice_2, 5, 13, 14, 40, 41
  ds4psy_guide, 15

  exp_wide, 8–12, 15, 17, 18, 23, 25, 35–38, 41–45, 47–49
  falsePosPsy_all, 8–12, 15, 16, 18, 23, 25, 35–38, 41–45, 47–49
  fame, 8–12, 15, 17, 17, 23, 25, 35–38, 41–45, 47–49
  is.integer, 18, 19
  is.wholenumber, 18, 21, 22
  l33t_rul35, 3, 4, 6, 19, 39, 46
  make_grid, 20

  num_as_char, 19, 20, 22
  num_as_ordinal, 19, 21, 22
  outliers, 8–12, 15, 17, 18, 23, 25, 35–38, 41–45, 47–49
  pal_ds4psy, 23, 24, 26, 28, 29, 31, 33
  pal_n_sq, 24, 24

56
INDEX

pi_100k, 8–12, 15, 17, 18, 23, 25, 35–38, 41–45, 47–49
plot_fn, 25, 28, 29, 31, 33, 45
plot_fun, 26, 27, 29, 31, 33, 45
plot_n, 26, 28, 29, 31, 33, 45
plot_text, 6, 26, 28, 29, 30, 33, 39, 45
plot_tiles, 24, 26, 28, 29, 31, 32, 45
posPsy_AHI_CESD, 8–12, 15, 17, 18, 23, 25, 34, 35–38, 41–45, 47–49
posPsy_long, 8–12, 15, 17, 18, 23, 25, 35, 35, 37, 38, 41–45, 47–49
posPsy_p_info, 8–12, 15, 17, 18, 23, 25, 35, 36, 36, 38, 41–45, 47–49
posPsy_wide, 8–12, 15, 17, 18, 23, 25, 35–37, 37, 41–45, 47–49
read_ascii, 3, 4, 6, 19, 38, 46
sample_dates, 5, 13, 14, 39, 41
sample_times, 5, 13, 14, 40, 40
t3, 8–12, 15, 17, 18, 23, 25, 35–38, 41, 42–45, 47–49
t4, 8–12, 15, 17, 18, 23, 25, 35–38, 41, 42, 43–45, 47–49
t_1, 8–12, 15, 17, 18, 23, 25, 35–38, 41–45, 47, 49

t_2, 8–12, 15, 17, 18, 23, 25, 35–38, 41–45, 47, 48, 49

t_3, 8–12, 15, 17, 18, 23, 25, 35–38, 41–45, 47, 48, 49

t_4, 8–12, 15, 17, 18, 23, 25, 35–38, 41–45, 47–49, 49
table6, 8–12, 15, 17, 18, 23, 25, 35–38, 41, 42, 42, 43–45, 47–49
table7, 8–12, 15, 17, 18, 23, 25, 35–38, 41–43, 43, 44, 45, 47–49
table8, 8–12, 15, 17, 18, 23, 25, 35–38, 41–43, 43, 45, 47–49
tb, 8–12, 15, 17, 18, 23, 25, 35–38, 41–44, 44, 47–49
theme_ds4psy, 26, 28, 29, 31, 33, 45
trans133t, 3, 4, 6, 19, 39, 46

what_date, 7, 8, 49, 51–54
what_day, 7, 8, 50, 50, 52–54
what_month, 7, 8, 50, 51, 51, 53, 54
what_time, 7, 8, 50–52, 52, 53, 54
what_week, 7, 8, 50–53, 53, 54
what_year, 7, 8, 50–53, 54