Package ‘ds4psy’
June 15, 2020

Type Package
Title Data Science for Psychologists
Version 0.3.0
Date 2020-06-15
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 (>= 3.5.0)
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
URL https://bookdown.org/hneth/ds4psy/,
https://github.com/hneth/ds4psy/
BugReports https://github.com/hneth/ds4psy/issues
VignetteBuilder knitr
RoxygenNote 7.1.0
Language en-US
NeedsCompilation no
Author Hansjoerg Neth [aut, cre] (<https://orcid.org/0000-0001-5427-3141>)
Repository CRAN
Date/Publication 2020-06-15 05:00:02 UTC
R topics documented:

<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bushisms</td>
<td>3</td>
</tr>
<tr>
<td>capitalize</td>
<td>4</td>
</tr>
<tr>
<td>caseflip</td>
<td>5</td>
</tr>
<tr>
<td>cclass</td>
<td>5</td>
</tr>
<tr>
<td>coin</td>
<td>6</td>
</tr>
<tr>
<td>countries</td>
<td>7</td>
</tr>
<tr>
<td>count_chars</td>
<td>7</td>
</tr>
<tr>
<td>count_words</td>
<td>8</td>
</tr>
<tr>
<td>cur_date</td>
<td>9</td>
</tr>
<tr>
<td>cur_time</td>
<td>10</td>
</tr>
<tr>
<td>data_1</td>
<td>11</td>
</tr>
<tr>
<td>data_2</td>
<td>11</td>
</tr>
<tr>
<td>data_t1</td>
<td>12</td>
</tr>
<tr>
<td>data_t1_de</td>
<td>12</td>
</tr>
<tr>
<td>data_t1_tab</td>
<td>13</td>
</tr>
<tr>
<td>data_t2</td>
<td>14</td>
</tr>
<tr>
<td>data_t3</td>
<td>14</td>
</tr>
<tr>
<td>data_t4</td>
<td>15</td>
</tr>
<tr>
<td>dice</td>
<td>15</td>
</tr>
<tr>
<td>dice_2</td>
<td>16</td>
</tr>
<tr>
<td>ds4psy.guide</td>
<td>18</td>
</tr>
<tr>
<td>exp_wide</td>
<td>18</td>
</tr>
<tr>
<td>falsePosPsy_all</td>
<td>19</td>
</tr>
<tr>
<td>fame</td>
<td>20</td>
</tr>
<tr>
<td>flowery</td>
<td>21</td>
</tr>
<tr>
<td>fruits</td>
<td>22</td>
</tr>
<tr>
<td>is.wholenumber</td>
<td>23</td>
</tr>
<tr>
<td>l33t_rul35</td>
<td>24</td>
</tr>
<tr>
<td>make_grid</td>
<td>24</td>
</tr>
<tr>
<td>metachar</td>
<td>25</td>
</tr>
<tr>
<td>num_as_char</td>
<td>26</td>
</tr>
<tr>
<td>num_as_ordinal</td>
<td>27</td>
</tr>
<tr>
<td>outliers</td>
<td>28</td>
</tr>
<tr>
<td>pal_ds4psy</td>
<td>29</td>
</tr>
<tr>
<td>pal_n_sq</td>
<td>29</td>
</tr>
<tr>
<td>pi_100k</td>
<td>30</td>
</tr>
<tr>
<td>plot_fn</td>
<td>31</td>
</tr>
<tr>
<td>plot_fun</td>
<td>32</td>
</tr>
<tr>
<td>plot_n</td>
<td>33</td>
</tr>
<tr>
<td>plot_text</td>
<td>35</td>
</tr>
<tr>
<td>plot_tiles</td>
<td>37</td>
</tr>
<tr>
<td>posPsy_AHI_CESD</td>
<td>39</td>
</tr>
<tr>
<td>posPsy_long</td>
<td>41</td>
</tr>
<tr>
<td>posPsy_p_info</td>
<td>42</td>
</tr>
<tr>
<td>posPsy_wide</td>
<td>43</td>
</tr>
<tr>
<td>read_ascii</td>
<td>44</td>
</tr>
</tbody>
</table>
**Description**

Bushisms contains phrases spoken by or attributed to U.S. president George W. Bush (the 43rd president of the United States, in office from January 2001 to January 2009).

**Usage**

Bushisms

**Format**

A vector of type character with length(Bushisms) = 22.

**Source**

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 objects and functions: Umlaut, caseflip(), cclass, count_chars(), count_words(), l33t_rul35, metachar, read_ascii(), text_to_sentences(), text_to_words(), transl33t()

**Examples**

```r
x <- c("Hello world! This is a 1st TEST sentence. The end."

capitalize(x)
capitalize(x, n = 3)
capitalize(x, n = 2, upper = FALSE)
capitalize(x, as_text = FALSE)

# Note: A vector of character strings returns the same results:
x <- c("Hello world!", "This is a 1st TEST sentence.", "The end."

capitalize(x)
capitalize(x, n = 3)
capitalize(x, n = 2, upper = FALSE)
capitalize(x, as_text = FALSE)
```
caseflip

---

**caseflip**  
`caseflip` flips the case of characters in a string of text `x`.

### Description

`caseflip` flips the case of characters in a string of text `x`.

### Usage

`caseflip(x)`

### Arguments

- **x**  
  A string of text (required).

### See Also

- `capitalize` for converting the case of initial letters.
- Other text objects and functions: `Umlaut`, `capitalize()`, `cclass`, `count_chars()`, `count_words()`, `l33t_rul35`, `metachar`, `read_ascii()`, `text_to_sentences()`, `text_to_words()`, `transl33t()`

### Examples

```r
x <- c("Hello world!", "This is a 1st sentence.", "This is the 2nd sentence.", "The end.")

caseflip(x)
```

class

---

**cclass**  
`cclass` provides character classes (as a named vector).

### Description

`cclass` provides different character classes (as a named character vector).

### Usage

`cclass`

### Format

An object of class `character` of length 6.

### Details

`cclass` allows illustrating matching character classes via regular expressions.  
See `?base::regex` for details.
See Also

metachar for a vector of metacharacters.

Other text objects and functions: Umlaut, capitalize(), caseflip(), count_chars(), count_words(), l33t_rul35, metachar, read_ascii(), text_to_sentences(), text_to_words(), transl33t()

Examples

class['hex'] # select by name
writeLines(cclass['pun'])
grep('[[:alpha:]]', cclass, value = TRUE)

---

**coin**

*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

```
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 sampling functions: dice_2(), dice(), sample_char(), sample_date(), sample_time()

Examples

```
# Basics:
coin() 
writeLines(coin(n = 100))
writeLines(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:
```
countries

<table>
<thead>
<tr>
<th>countries</th>
</tr>
</thead>
</table>

Data: Names of countries.

Description
countries is a dataset containing the names of 197 countries (as a vector of text strings).

Usage
countries

Format
A vector of type character with length(countries) = 197.

Source
Data from https://www.gapminder.org: Original data at https://www.gapminder.org/data/documentation/gd004/.

See Also
Other datasets: Bushisms, Trumpisms, data_1, data_2, data_t1_de, data_t1_tab, data_t1_data_t2, data_t3, data_t4, exp_wide, falsePosPsy_all, fame, flowery, fruits, 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

count_chars

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

Description
count_chars counts the frequency of characters in a string of text x.

Usage
count_chars(x, case_sense = TRUE, rm_specials = TRUE, sort_freq = TRUE)
count_words

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

count_words for counting the frequency of words; plot_text for a corresponding plot function.

Other text objects and functions: Umlaut, capitalize(), caseflip(), cclass, count_words(), l33t_rul35, metachar, read_ascii(), text_to_sentences(), text_to_words(), transl33t()

Examples

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

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

count_words counts the frequency of words in a string of text x.

Description

count_words counts the frequency of words in a string of text x.

Usage

count_words(x, case_sense = TRUE, sort_freq = TRUE)

Arguments

x | A string of text (required).
case_sense | Boolean: Distinguish lower- vs. uppercase characters? Default: case_sense = TRUE.
sort_freq | Boolean: Sort output by word frequency? Default: sort_freq = TRUE.
See Also

count_chars for counting the frequency of characters; plot_text for a corresponding plot function.

Other text objects and functions: Umlaute, capitalize(), caseflip(), cclass, count_chars(), l33t_rul35, metachar, read_ascii(), text_to_sentences(), text_to_words(), transl33t()

Examples

# Default:
s3 <- c("A first sentence.", "The second sentence.",
    "A third --- and also the final --- sentence.")
count_words(s3) # case-sensitive, sorts by frequency

# Options:
count_words(s3, case_sense = FALSE) # case insensitive
count_words(s3, sort_freq = FALSE) # sorts alphabetically

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 = "-")

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

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

Arguments

- `seconds` Boolean: Show time with seconds? Default: `seconds = FALSE`.

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

```r
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: Bushisms, Trumpisms, countries, data_2, data_t1_de, data_t1_tab, data_t1, data_t2, data_t3, data_t4, exp_wide, falsePosPsy_all, fame, flowery, fruits, 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: Bushisms, Trumpisms, countries, data_1, data_t1_de, data_t1_tab, data_t1, data_t2, data_t3, data_t4, exp_wide, falsePosPsy_all, fame, flowery, fruits, 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).
**data_t1_tab**

**Source**

See CSV data at [http://rpository.com/ds4psy/data/data_t1_de.csv](http://rpository.com/ds4psy/data/data_t1_de.csv).

**See Also**

Other datasets: Bushisms, Trumpisms, countries, data_1, data_2, data_t1_tab, data_t1, data_t2, data_t3, data_t4, exp_wide, falsePosPsy_all, fame, flowery, fruits, 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

---

**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).

**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: Bushisms, Trumpisms, countries, data_1, data_2, data_t1_de, data_t1, data_t2, data_t3, data_t4, exp_wide, falsePosPsy_all, fame, flowery, fruits, 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: Bushisms, Trumpisms, countries, data_1, data_2, data_t1_de, data_t1_tab, data_t1, data_t3, data_t4, exp Wide, falsePosPsy_all, fame, flowery, fruits, outliers, pi_l00k, 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

**Format**

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

**Source**

**data_t4**

**See Also**

Other datasets: Bushisms, Trumpisms, countries, data_1, data_2, data_t1_de, data_t1_tab, data_t1, data_t2, data_t4, exp_wide, falsePosPsy_all, fame, flowery, fruits, 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_t4**

*Data table data_t4.*

**Description**

Data_t4 is a fictitious dataset to practice importing and joining data (from a CSV file).

**Usage**

data_t4

**Format**

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

**Source**


**See Also**

Other datasets: Bushisms, Trumpisms, countries, data_1, data_2, data_t1_de, data_t1_tab, data_t1, data_t2, data_t4, exp_wide, falsePosPsy_all, fame, flowery, fruits, 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**

*Throw a fair dice (with a given number of sides) n times.*

**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: 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 sampling functions: coin(), dice_2(), sample_char(), sample_date(), sample_time()

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(0)
dice(1/2)
dice(2:3)
dice(5, events = NA)
dice(5, events = 1/2)
dice(NULL, NULL)

---

dice_2

Throw a questionable dice (with a given number of sides) n times.
**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: \( n = 1 \).
- \( \text{sides} \) Number of sides. Default: \( \text{sides} = 6 \).

**Details**

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

**See Also**

Other sampling functions: coin(), dice(), sample_char(), sample_date(), sample_time()

**Examples**

```r
# 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.
```
ds4psy.guide  

*Opens user guide of the ds4psy package.*

**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: Bushisms, Trumpisms, countries, data_1, data_2, data_t1_de, data_t1_tab, data_t1, data_t2, data_t3, data_t4, falsePosPsy_all, fame, flowery, fruits, 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: Jarious 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: Bushisms, Trumpisms, countries, data_1, data_2, data_t1_de, data_t1_tab, data_t1, data_t2, data_t3, data_t4, exp_wide, fame, flowery, fruits, 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.
Data: Flowery phrases.

Description

flowery contains versions and variations of Gertrude Stein’s popular phrase "A rose is a rose is a rose".

Usage

flowery

Format

A vector of type character with length(flowery) = 60.

Details

The phrase stems from Gertrude Stein’s poem "Sacred Emily" (written in 1913 and published in 1922, in "Geography and Plays"). The verbatim line in the poem actually reads "Rose is a rose is a rose is a rose".

See https://en.wikipedia.org/wiki/Rose_is_a_rose_is_a_rose_is_a_rose for additional variations and sources.

Source

Data based on https://en.wikipedia.org/wiki/Rose_is_a_rose_is_a_rose_is_a_rose.
fruits

Data: Names of fruits.

Description

fruits is a dataset containing the names of 122 fruits (as a vector of text strings).

Usage

fruits

Format

A vector of type character with length(fruits) = 122.

Details

Botanically, "fruits" are the seed-bearing structures of flowering plants (angiosperms) formed from the ovary after flowering.

In common usage, "fruits" refer to the fleshy seed-associated structures of a plant that taste sweet or sour, and are edible in their raw state.

Source


See Also

Other datasets: Bushisms, Trumpisms, countries, data_1, data_2, data_t1_de, data_t1_tab, data_t1, data_t2, data_t3, data_t4, exp_wide, falsePosPsy_all, fame, fruits, 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
Description

`is.wholenumber` tests if `x` contains integer numbers.

Usage

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

Arguments

- `x` Number(s) to test (required, accepts numeric vectors).

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)
```
l33t_rul35 provides rules for translating text into leet/l33t slang.

Description

l33t_rul35 specifies rules for translating characters into other characters (typically symbols) to mimic leet/l33t slang (as a named character vector).

Usage

l33t_rul35

Format

An object of class character of length 13.

Details

Old (i.e., to be replaced) characters are `paste(names(l33t_rul35), collapse = "")`.

New (i.e., replaced) characters are `paste(l33t_rul35, collapse = "")`.


See Also

- `transl33t` for a corresponding function.
- Other text objects and functions: `Umlaut`, `capitalize()`, `caseflip()`, `cclass`, `count_chars()`, `count_words()`, `metachar`, `read_ascii()`, `text_to_sentences()`, `text_to_words()`, `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

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)
```

---

**Description**

`metachar` provides the metacharacters of extended regular expressions (as a character vector).

**Usage**

```r
metachar
```

**Format**

An object of class `character` of length 12.

**Details**

`metachar` allows illustrating the notion of meta-characters in regular expressions (and provides corresponding exemplars).

See `?base::regex` for details.

**See Also**

`cclass` for a vector of character classes.

Other text objects and functions: `Umlaut, capitalize(), caseflip(), cclass, count_chars(), count_words(), l33t_rul35, read_ascii(), text_to_sentences(), text_to_words(), transl33t()`

**Examples**

```r
metachar
length(metachar)  # 12
nchar(paste0(metachar, collapse = ""))  # 12
```
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

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 = ".".

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

num_as_char(1)
num_as_char(10/3)
num_as_char(1000/6)

# rounding down:
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:
num_as_char(1.6666, n_pre_dec = 1, n_dec = 0)
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

```r
um_as_ordinal(1:4)
um_as_ordinal(10:14)  # all with "th"
um_as_ordinal(110:114) # all with "th"
um_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

```r
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

**pal_ds4psy**

See Also

Other datasets: Bushisms, Trumpisms, countries, data_1, data_2, data_t1_de, data_t1_tab, data_t1, data_t2, data_t3, data_t4, exp_wide, falsePosPsy_all, fame, flowery, fruits, 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

---

**Description**

pal_ds4psy provides a dedicated color palette.

**Usage**

pal_ds4psy

**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**

Get n-by-n dedicated colors of a color palette.

---

**Description**

pal_n_sq returns n^2 dedicated colors of a color palette pal (up to a maximum of n = "all" colors).

**Usage**

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

```r
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

```r
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: Bushisms, Trumpisms, countries, data_1, data_2, data_t1_de, data_t1_tab, data_t1, data_t2, data_t3, data_t4, exp_wide, falsePosPsy_all, fame, flowery, fruits, 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 = TRUE.
B Boolean. Default: B = FALSE.
C Boolean. Default: C = TRUE.
D Boolean. Default: D = FALSE.
E Boolean. Default: E = FALSE.
F Boolean. Default: F = FALSE.
f A color palette (e.g., as a vector). Default: f = 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 = "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.

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

Examples

```r
# Basics:
plot_fn()

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

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`.
- **c**: Boolean. Default: `c = TRUE`.
- **d**: A (decimal) number. Default: `d = 1.0`.
- **e**: Boolean. Default: `e = FALSE`.
- **f**: Boolean. Default: `f = FALSE`. 
plot_n

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,
)```
plot_n

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).
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, # 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_n(n = 4, polar = TRUE, sort = FALSE) # PIE in random order
plot_n(n = 5, polar = TRUE, row = FALSE, borders = FALSE) # TARGET no borders
plot_n(n = 5, polar = TRUE, lbl_tiles = TRUE) # PIE with tile labels
plot_n(n = 5, polar = TRUE, row = FALSE, lbl_title = TRUE) # TARGET with title label
plot_n(n = 4, row = TRUE, sort = FALSE, borders = TRUE,
       border_col = "white", border_size = 2,
       polar = TRUE, rseed = 132)
plot_n(n = 4, row = FALSE, sort = FALSE, borders = TRUE,
       border_col = "white", border_size = 2,
       polar = TRUE, rseed = 134)

---

**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**

```r
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 = ""

c_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".

pal Color palette for filling tiles of text (used in order of character frequency). Default: pal = pal_ds4psy[1:5] (i.e., shades of unikn: :Seeb1au).

c_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.

c_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_text("test.txt", col_lbl = "darkblue", cex = 4, family = "sans", fontface = 3,
#  pal = "gold!", pal_extend = TRUE, border_col = NA)
# plot_text("test.txt", family = "serif", cex = 6, lbl_rotate = TRUE,
#  pal = NA, borders = FALSE)
# plot_text("test.txt", col_lbl = "white", pal = c("green3", "black"),
#  border_col = "black", border_size = .2)

## Color ranges:
# plot_text("test.txt", pal = c("red2", "orange", "gold"))
# plot_text("test.txt", pal = c("olivedrab4", "gold"))

# unlink("test.txt") # clean up (by deleting file).

## (b) Plot text (from file in subdir):
# plot_text("data-raw/txt/hello.txt") # requires txt file
# plot_text(file = "data-raw/txt/ascii.txt", cex = 5,
#  col_bg = "grey", char_bg = ")

## (c) Plot text input (from console):
# plot_text()

---

plot_tiles

Plot n-by-n tiles.

Description

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

```r
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 = ""
)
```

Arguments

- **n**: Basic number of tiles (on either side).
- **pal**: Color palette (automatically extended to \( n \times n \) colors). Default: `pal = pal_ds4psy`.
- **sort**: Boolean: Sort tiles? Default: `sort = TRUE` (i.e., sorted tiles).
- **borders**: Boolean: 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 = 0.2`.
- **lbl_tiles**: Boolean: Add numeric labels to tiles? Default: `lbl_tiles = FALSE` (i.e., no labels).
- **lbl_title**: Boolean: Add numeric label (of \( n \)) to plot? Default: `lbl_title = FALSE` (i.e., no title).
- **polar**: Boolean: Plot on polar coordinates? Default: `polar = FALSE` (i.e., using fixed coordinates).
- **rseed**: Random seed (number). Default: `rseed = NA` (using random seed).
- **save**: Boolean: 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 = ""`.

See Also

- `pal_ds4psy` for default color palette.

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

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

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

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

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

# (2) polar plot:
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](https://bookdown.org/hneth/ds4psy/B-1-datasets-pos.html).

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: Bushisms, Trumpisms, countries, data_1, data_2, data_t1_de, data_t1_tab, data_t1, data_t2, data_t3, data_t4, exp_wide, falsePosPsy_all, fame, flowery, fruits, outliers, pi_100k, posPsy_long, posPsy_p_info, posPsy_wide, t3, t4, t_1, t_2, t_3, t_4, table6, table7, table8, tb
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 [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_AHI_CESD for source of this file and codebook information; posPsy_wide for a version of this file (in wide format).

Other datasets: Bushisms, Trumpisms, countries, data_1, data_2, data_t1_de, data_t1_tab, data_t1, data_t2, data_t3, data_t4, exp_wide, falsePosPsy_all, fame, flowery, fruits, 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
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


posPsy_wide

See Also

Other datasets: Bushisms, Trumpisms, countries, data_1, data_2, data_t1_de, data_t1_tab, data_t1, data_t2, data_t3, data_t4, exp_wide, falsePosPsy_all, fame, flowery, fruits, 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


read_ascii

read_ascii parses text (from a file) into a table.

Description

read_ascii parses text (from a file or from user input in Console) into a table that contains a row for each character.

Usage

read_ascii(file = "", flip_y = FALSE)

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 = "".

flip_y

Boolean: Should y-coordinates be flipped, so that the lowest line in the text file becomes y = 1, and the top line in the text file becomes y = n_lines? Default: flip_y = FALSE.

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 objects and functions: Umlaut, capitalize(), caseflip(), cclass, count_chars(), count_words(), l33t_rul35, metachar, text_to_sentences(), text_to_words(), 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

# unlink("test.txt") # clean up (by deleting file).

## (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_char**

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

**Description**

`sample_char` draws a sample of `n` random characters from a given range of characters.

**Usage**

```r
sample_char(x_char = c(letters, LETTERS), n = 1, replace = FALSE, ...)
```

**Arguments**

- `x_char`: Population of characters to sample from. Default: `x_char = c(letters, LETTERS)`.
- `n`: Number of characters to draw. Default: `n = 1`.
- `...`: Other arguments. (Use for specifying `prob`, as passed to `sample()`.)

**Details**

By default, `sample_char` draws `n = 1` a random alphabetic character from `x_char = c(letters, LETTERS).` As with `sample()`, the sample size `n` must not exceed the number of available characters `nchar(x_char)`, unless `replace = TRUE` (i.e., sampling with replacement).
sample_date

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

Description

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

Usage

sample_date(n = 1, from = "1970-01-01", to = Sys.Date())

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

See Also

Other sampling functions: coin(), dice_2(), dice(), sample_char(), sample_time()
**sample_time**

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

**Description**

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

**Usage**

```r
sample_time(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_time draws `n = 1` random time in the range `from = "1970-01-01 00:00:00"` to `to = Sys.time()` (current time).

**See Also**

Other sampling functions: `coin()`, `dice_2()`, `dice()`, `sample_char()`, `sample_date()`

**Examples**

# Basics:
sample_time()
sample_time(n = 10)

# Specific ranges:
sort(sample_time(n = 10, from = (Sys.time() - 60))) # within the last minute
sort(sample_time(n = 10, from = (Sys.time() - 1 * 60 * 60))) # within the last hour
```r
sort(sample_time(n = 10, from = Sys.time(),
             to = (Sys.time() + 1 * 60 * 60))) # within the next hour
```r
sort(sample_time(n = 10, from = "2020-01-01 00:00:00 CET",
```
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: Bushisms, Trumpisms, countries, data_1, data_2, data_t1_de, data_t1_tab, data_t1, data_t2, data_t3, data_t4, exp_wide, falsePosPsy_all, fame, flowery, fruits, 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

---

t4

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: Bushisms, Trumpisms, countries, data_1, data_2, data_t1_de, data_t1_tab, data_t1, data_t2, data_t3, data_t4, exp_wide, falsePosPsy_all, fame, flowery, fruits, 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

---

**table6**  
*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: Bushisms, Trumpisms, countries, data_1, data_2, data_t1_de, data_t1_tab, data_t1, data_t2, data_t3, data_t4, exp_wide, falsePosPsy_all, fame, flowery, fruits, outliers, pi_100k, posPsy_AHI_CESD, posPsy_long, posPsy_p_info, posPsy_wide, t3, t_4, t_1, t_2, t_3, t_4, table7, table8, tb
table7

Data 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 tidy package.

Source


See Also

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

table8

Data 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: `Bushisms`, `Trumpisms`, `countries`, `data_1`, `data_2`, `data_t1_de`, `data_t1_tab`, `data_t2`, `data_t3`, `data_t4`, `exp_wide`, `falsePosPsy_all`, `fame`, `flowery`, `fruits`, `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`: Shoe size (EU standard).
- 5. `IQ`: IQ score (according Raven’s Regressive Tables).

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

Source

See Also

Other datasets: Bushisms, Trumpisms, countries, data_1, data_2, data_t1_de, data_t1_tab, data_t1, data_t2, data_t3, data_t4, exp_wide, falsePosPsy_all, fame, flowery, fruits, 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

text_to_sentences splits a string of text x (consisting of one or more character strings) into a vector of its constituting sentences.

text_to_sentences splits at given punctuation marks (as a regular expression, default: split_delim = "\.|\?|!")) and removes empty leading and trailing spaces before returning a vector of the remaining character sequences (as the sentences).

Usage

text_to_sentences(x, split_delim = "\.|\?|!", force_delim = FALSE)

Arguments

x A string of text (required), typically a character vector.

split_delim Sentence delimiters (as regex) used to split x into substrings. By default, split_delim = "\.|\?|!".

force_delim Boolean: Enforce splitting at split_delim? If force_delim = FALSE (as per default), the function assumes a standard sentence-splitting pattern: split_delim is followed by a single space and a capital letter. If force_delim = TRUE, splits at split_delim are enforced (regardless of spacing or capitalization).

Details

The Boolean force_delim distinguishes between two splitting modes:

1. If force_delim = FALSE (as per default), the function assumes a standard sentence-splitting pattern: A sentence delimiter in split_delim must be followed by a single space and a capital letter starting the next sentence. Sentence delimiters in split_delim are not removed from the output.

2. If force_delim = TRUE, the function enforces splits at each delimiter in split_delim. For instance, any dot (i.e., the metacharacter ".") is interpreted as a full stop, so that sentences containing dots mid-sentence (e.g., for abbreviations, etc.) are split into parts. Sentence delimiters in split_delim are removed from the output.

Internally, text_to_sentences uses \texttt{strsplit} to split strings.
text_to_words

See Also

text_to_words for splitting text into a vector of words; count_words for counting the frequency of words; strsplit for splitting strings.

Other text objects and functions: Umlaut, capitalize(), caseflip(), cclass, count_chars(), count_words(), l33t_rul35, metachar, read_ascii(), text_to_words(), transl33nt()

Examples

x <- c("A first sentence. Exclamation sentence!",
       "Any questions? But etc. can be tricky. A fourth --- and final --- sentence.")
text_to_sentences(x)
text_to_sentences(x, force_delim = TRUE)

# Changing split delimiters:
text_to_sentences(x, split_delim = "\\." ) # only split at "."

text_to_sentences("Buy apples, berries, and coconuts.")
text_to_sentences("Buy apples, berries; and coconuts.",
                split_delim = "|;\.|", force_delim = TRUE)
text_to_sentences(c("123. 456? 789! 007 etc."), force_delim = TRUE)
text_to_sentences("Dr. Who is problematic.")

---

## text_to_words

text_to_words splits a string of text x (consisting of one or more character strings) into a vector of its constituting words.

### Description

text_to_words removes all (standard) punctuation marks and empty spaces in the resulting parts, before returning a vector of the remaining character symbols (as the words).

### Usage

text_to_words(x)

### Arguments

x A string of text (required), typically a character vector.

### Details

Internally, text_to_words uses strsplit to split strings.
theme_ds4psy

See Also
text_to_sentences for splitting text into a vector of sentences; count_words for counting the frequency of words; strsplit for splitting strings.

Other text objects and functions: Umlaut, capitalize(), caseflip(), cclass, count_chars(), count_words(), l33t_rul35, metachar, read_ascii(), text_to_sentences(), transl33t()

Examples

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

theme_ds4psy  
ds4psy default plot theme (using ggplot2 and unikn).

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 translates 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_rule3` for default rules used.

Other text objects and functions: `Umlaut`, `capitalize()`, `caseflip()`, `cclass`, `count_chars()`, `count_words()`, `l33t_rule3`, `metachar`, `read_ascii()`, `text_to_sentences()`, `text_to_words()`

Examples

```r
# Use defaults:
transl33t(txt = "hello world")
transl33t(txt = c(letters))
transl33t(txt = c(LETTERS))

# Specify rules:
transl33t(txt = "hello world",
           rules = c("e" = "3", "l" = "1", "o" = "0"))

# Set input and output case:
transl33t(txt = "hello world", in_case = "up",
           rules = c("e" = "3", "l" = "1", "o" = "0"))  # e only capitalized
transl33t(txt = "hE1lo world", in_case = "lo", out_case = "up",
           rules = c("e" = "3", "l" = "1", "o" = "0"))  # e transl33ted
```

---

### Trumpisms

*Data: Trumpisms.*

### Description

Trumpisms contains words frequently used by U.S. president Donald J. Trump (the 45th and current president of the United States, as of May 2020).

### Usage

Trumpisms

### Format

A vector of type character with `length(Trumpisms) = 96`.

### Source

Data originally based on [https://www.yourdictionary.com/slideshow/donald-trump-20-most-frequently-used-words.html](https://www.yourdictionary.com/slideshow/donald-trump-20-most-frequently-used-words.html) and expanded by public speeches and Twitter tweets on [https://twitter.com/realDonaldTrump](https://twitter.com/realDonaldTrump).
Data \textit{t}_1.

**Description**
\textit{t}_1 is a fictitious dataset to practice tidying data.

**Usage**
\textit{t}_1

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

**Source**
See CSV data at \url{http://rpository.com/ds4psy/data/t_1.csv}.

See Also
Other datasets: Bushisms, Trumpisms, countries, data\_1, data\_2, data\_t1\_de, data\_t1\_tab, data\_t2, data\_t3, data\_t4, exp\_wide, falsePosPsy\_all, fame, flowery, fruits, 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 \textit{t}_2.

**Description**
\textit{t}_2 is a fictitious dataset to practice tidying data.

**Usage**
\textit{t}_2

**Format**
A table with 8 cases (rows) and 5 variables (columns).
t_3

Source

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

See Also

Other datasets: Bushisms, Trumpisms, countries, data_1, data_2, data_t1_de, data_t1_tab, data_t1, data_t2, data_t3, data_t4, exp_wide, falsePosPsy_all, fame, flowery, fruits, 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

<table>
<thead>
<tr>
<th>t_3</th>
<th>Data t_3.</th>
</tr>
</thead>
</table>

Description

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


See Also

Other datasets: Bushisms, Trumpisms, countries, data_1, data_2, data_t1_de, data_t1_tab, data_t1, data_t2, data_t3, data_t4, exp_wide, falsePosPsy_all, fame, flowery, fruits, 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
Data t_4

Description

Data 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


See Also

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

Umlaut

Umlaut provides German Umlaut letters (as Unicode characters).

Description

Umlaut provides the German Umlaut letters (aka. diaeresis/diacritic) as a named character vector.

Usage

`Umlaut`

Format

An object of class character of length 7.

Details

For Unicode details, see https://home.unicode.org/.

what_date

See Also

Other text objects and functions: capitalize(), caseflip(), cclass, count_chars(), count_words(), l33t_rul35, metachar, read_ascii(), text_to_sentences(), text_to_words(), transl33t()

Examples

Umlaut

names(Umlaut)

paste0("Hansj", Umlaut["o"], "rg i", Umlaut["s"], "t s", Umlaut["u"], "sse ", Umlaut["A"], "pfel.")
paste0("Das d", Umlaut["u"], "nne M", Umlaut["a"], "dchen l", Umlaut["a"], "chelt.")
paste0("Der b", Umlaut["o"], "se Mann macht ", Umlaut["u"], "blen ", Umlaut["A"], "rger.")
paste0("Das ", Umlaut["U"], "ber-Ich ist ", Umlaut["a"], "rgerlich.")

what_date

<table>
<thead>
<tr>
<th>What date is it?</th>
</tr>
</thead>
<tbody>
<tr>
<td>what_date</td>
</tr>
</tbody>
</table>

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)
```

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`.

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()`
what_month

Examples

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)

what_month

What month is it?

Description

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

Usage

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.

Details

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

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_time

what_time(when = ds)
what_time(when = ds, abbr = TRUE, as_integer = FALSE)
what_time(when = ds, abbr = TRUE, as_integer = TRUE)

what_time (What time is it?)

Description

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

Usage

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.
sep Character: Separator to use. Default: sep = ":".

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(),

Examples

what_week()
what_week(as_integer = TRUE)

# Other dates/times:
d1 <- as.Date("2019-08-23")
what_week(when = d1, unit = "year")
what_week(when = d1, unit = "month")

# Work with vectors (when as characters):
ds1 <- c("2020-01-01", "2020-02-29", "2020-12-24", "2020-12-31")
what_week(when = ds)
what_week(when = ds, unit = "month", as_integer = TRUE)
what_week(when = ds, unit = "year", as_integer = TRUE)
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()

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

Bushisms, 3
class, 5
countries, 7
data_1, 11
data_2, 11
data_t1, 12
data_t1_de, 12
data_t1_tab, 13
data_t2, 14
data_t3, 14
data_t4, 15
exp_wide, 18
falsePosPsy_all, 19
fame, 20
flowery, 21
fruits, 22
l33t_rul35, 24
metachar, 25
outliers, 28
pal_ds4psy, 29
pi_100k, 30
posPsy_AHI_CESD, 39
posPsy_long, 41
posPsy_p_info, 42
posPsy_wide, 43
t3, 48
t4, 48
t_1, 57
t_2, 57
t_3, 58
t_4, 59
table6, 49
table7, 50
table8, 50
tb, 51
Trumpisms, 56
Umlaut, 59

capitalize, 4, 5, 6, 8, 9, 24, 25, 44, 53, 54, 56, 60
caseflip, 4, 5, 6, 8, 9, 24, 25, 44, 53, 54, 56, 60
class, 4, 5, 5, 8, 9, 24, 25, 44, 53, 54, 56, 60
coin, 6, 16, 17, 46, 47
count_chars, 4–6, 7, 9, 24, 25, 44, 53, 54, 56, 60
count_words, 4–6, 8, 8, 24, 25, 44, 53, 54, 56, 60
cur_date, 9, 10, 60–65
cur_time, 9, 10, 60–65
data_1, 4, 7, 11, 12–15, 18, 20–22, 29, 30, 40, 41, 43, 44, 48–52, 57–59
data_t1, 4, 7, 11, 12, 12–15, 18, 20–22, 29, 30, 40, 41, 43, 44, 48–52, 57–59
data_t1_de, 4, 7, 11, 12, 12–15, 18, 20–22, 29, 30, 40, 41, 43, 44, 48–52, 57–59
data_t1_tab, 4, 7, 11–13, 13, 14, 15, 18, 20–22, 29, 30, 40, 41, 43, 44, 48–52, 57–59
data_t2, 4, 7, 11–13, 14, 15, 18, 20–22, 29, 30, 40, 41, 43, 44, 48–52, 57–59
data_t3, 4, 7, 11–14, 14, 15, 18, 20–22, 29, 30, 40, 41, 43, 44, 48–52, 57–59
dice, 6, 15, 17, 46, 47
dice_2, 6, 16, 16, 46, 47
ds4psy_guide, 18

INDEX

falsePosPsy_all, 4, 7, 11–15, 18, 19, 21, 22, 29, 30, 40, 41, 43, 44, 48–52, 57–59

is.integer, 23
is.wholenumber, 23, 26, 28

l33t_rul35, 4–6, 8, 9, 24, 25, 44, 53–56, 60
make_grid, 24
metachar, 4–6, 8, 9, 24, 25, 44, 53, 54, 56, 60
num_as_char, 23, 26, 28
num_as_ordinal, 23, 26, 27


pal_ds4psy, 29, 29, 30, 32–34, 36, 38
pal_n_sq, 29, 29
plot_fn, 31, 33, 34, 36, 38, 55
plot_fun, 32, 32, 34, 36, 38, 55
plot_n, 32, 33, 33, 36, 38, 55
plot_text, 8, 9, 32–34, 35, 38, 44, 55
plot_tiles, 30, 32–34, 36, 37, 55

read_ascii, 4–6, 8, 9, 24, 25, 36, 44, 53, 54, 56, 60

sample_char, 6, 16, 17, 45, 46, 47
sample_date, 6, 16, 17, 46, 46, 47
sample_time, 6, 16, 17, 46, 47
strsplit, 52–54

t_1, 4, 7, 11–15, 18, 20–22, 29, 30, 40, 41, 43, 44, 48–52, 57, 57, 58, 59
t_2, 4, 7, 11–15, 18, 20–22, 29, 30, 40, 41, 43, 44, 48–52, 57, 57, 58, 59
t_3, 4, 7, 11–15, 18, 20–22, 29, 30, 40, 41, 43, 44, 48–52, 57, 58, 59
t_4, 4, 7, 11–15, 18, 20–22, 29, 30, 40, 41, 43, 44, 48–52, 57, 58, 59
text_to_sentences, 4–6, 8, 9, 24, 25, 44, 52, 54, 56, 60
text_to_words, 4–6, 8, 9, 24, 25, 44, 53, 53, 56, 60
theme_ds4psy, 32–34, 36, 38, 54
transl33t, 4–6, 8, 9, 24, 25, 44, 53, 54, 55, 60

Umlaut, 4–6, 8, 9, 24, 25, 44, 53, 54, 56, 59

what_date, 9, 10, 60, 61–65
what_day, 9, 10, 60, 61, 62–65
what_month, 9, 10, 60, 61, 62, 63–65
what_time, 9, 10, 60–62, 63, 64, 65
what_week, 9, 10, 60–63, 64, 65
what_year, 9, 10, 60–64, 65