Package ‘ds4psy’
April 19, 2020

Type Package
Title Data Science for Psychologists
Version 0.2.0
Date 2020-04-20
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 dplyr, ggplot2, cowplot, magrittr, here, tibble, 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 7.0.2
Language en-US
NeedsCompilation no
Repository CRAN
Date/Publication 2020-04-19 07:40:02 UTC
### R topics documented:

<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>capitalize</td>
<td>3</td>
</tr>
<tr>
<td>caseflip</td>
<td>4</td>
</tr>
<tr>
<td>coin</td>
<td>4</td>
</tr>
<tr>
<td>count_char</td>
<td>5</td>
</tr>
<tr>
<td>cur_date</td>
<td>6</td>
</tr>
<tr>
<td>cur_time</td>
<td>7</td>
</tr>
<tr>
<td>data_1</td>
<td>8</td>
</tr>
<tr>
<td>data_2</td>
<td>8</td>
</tr>
<tr>
<td>data_t1</td>
<td>9</td>
</tr>
<tr>
<td>data_t1_de</td>
<td>9</td>
</tr>
<tr>
<td>data_t1_tab</td>
<td>10</td>
</tr>
<tr>
<td>data_t2</td>
<td>10</td>
</tr>
<tr>
<td>data_t3</td>
<td>11</td>
</tr>
<tr>
<td>data_t4</td>
<td>12</td>
</tr>
<tr>
<td>dice</td>
<td>12</td>
</tr>
<tr>
<td>dice_2</td>
<td>13</td>
</tr>
<tr>
<td>ds4psy.guide</td>
<td>14</td>
</tr>
<tr>
<td>exp_wide</td>
<td>15</td>
</tr>
<tr>
<td>falsePosPsy_all</td>
<td>15</td>
</tr>
<tr>
<td>fame</td>
<td>17</td>
</tr>
<tr>
<td>is.wholenumber</td>
<td>18</td>
</tr>
<tr>
<td>l33t_rul35</td>
<td>19</td>
</tr>
<tr>
<td>make_grid</td>
<td>19</td>
</tr>
<tr>
<td>num_as_char</td>
<td>20</td>
</tr>
<tr>
<td>num_as_ordinal</td>
<td>21</td>
</tr>
<tr>
<td>outliers</td>
<td>22</td>
</tr>
<tr>
<td>pal_ds4psy</td>
<td>23</td>
</tr>
<tr>
<td>pal_n_sq</td>
<td>24</td>
</tr>
<tr>
<td>pi_100k</td>
<td>24</td>
</tr>
<tr>
<td>plot_fn</td>
<td>25</td>
</tr>
<tr>
<td>plot_fun</td>
<td>26</td>
</tr>
<tr>
<td>plot_n</td>
<td>28</td>
</tr>
<tr>
<td>plot_text</td>
<td>30</td>
</tr>
<tr>
<td>plot_tiles</td>
<td>32</td>
</tr>
<tr>
<td>posPsy_AHI_CESD</td>
<td>34</td>
</tr>
<tr>
<td>posPsy_long</td>
<td>35</td>
</tr>
<tr>
<td>posPsy_p_info</td>
<td>36</td>
</tr>
<tr>
<td>posPsy_wide</td>
<td>37</td>
</tr>
<tr>
<td>read_ascii</td>
<td>38</td>
</tr>
<tr>
<td>sample_dates</td>
<td>39</td>
</tr>
<tr>
<td>sample_times</td>
<td>40</td>
</tr>
<tr>
<td>t3</td>
<td>41</td>
</tr>
<tr>
<td>t4</td>
<td>41</td>
</tr>
<tr>
<td>table6</td>
<td>42</td>
</tr>
<tr>
<td>table7</td>
<td>42</td>
</tr>
<tr>
<td>table8</td>
<td>43</td>
</tr>
</tbody>
</table>
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()

Examples
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)
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 functions: capitalize(), count_char(), l33t_rul35, read_ascii(), transl33t()

Examples

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

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").
count_char

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

# 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

```r
# 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

```r
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

    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 tibble 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`, `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 tibble with 100 cases (rows) and 4 variables (columns).

**Source**

**data_t1**

*Data table* \textit{data\_t1}.

**Description**

\textit{data\_t1} is a fictitious dataset to practice importing and joining data (from a CSV file).

**Usage**

\texttt{data\_t1}

**Format**

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

**Source**

See CSV data at \url{http://rpository.com/ds4psy/data/data\_t1.csv}.

**See Also**

Other datasets: \texttt{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\_AH\_CESD, posPsy\_long, posPsy\_p\_info, posPsy\_wide, t3, t4, table6, table7, table8, tb}

**data_t1**

*Data import* \textit{data\_t1\_de}.

**Description**

\textit{data\_t1\_de} is a fictitious dataset to practice data import (from a CSV file, de/European style).

**Usage**

\texttt{data\_t1\_de}

**Format**

A tibble 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_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, table6, table7, table8, tb

data_t1_tab

\textit{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 tibble with 20 cases (rows) and 4 variables (columns).

Source

See TAB-delimited data at 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, table6, table7, table8, tb

data_t2

\textit{Data table data_t2.}

Description

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

Usage

data_t2
data_t3

Format

A tibble 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_t3, data_t4, exp_wide, falsePosPsy_all, fame, outliers, pi_100k, posPsy_AHI_CESD, posPsy_long, posPsy_p_info, posPsy_wide, t3, t4, table6, table7, table8, tb
Description

data_t4 is a fictitious dataset to practice importing and joining data (from a CSV file).

Usage
data_t4

Format

A tibble 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_t3, exp_wide, falsePosPsy_all, fame, outliers, pi_100k, posPsy_AHI_CESD, posPsy_long, posPsy_p_info, posPsy_wide, t3, t4, 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.
dice_2

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(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.
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.

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 tibble with 10 cases (rows) and 7 variables (columns).

Source

See CSV data at [http://repository.com/ds4psy/data/exp_wide.csv](http://repository.com/ds4psy/data/exp_wide.csv).

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, 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 tibble 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: 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, table6, table7, table8, tb

---

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

**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 tibble 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, 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

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 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 [https://en.wikipedia.org/wiki/Leet](https://en.wikipedia.org/wiki/Leet) for details.

**See Also**

Other text functions: `capitalize()`, `caseflip()`, `count_char()`, `read_ascii()`, `transl33t()`

---

`make_grid` generates a grid of x-y coordinates and returns it as a tibble.

**Description**

`make_grid` generates a grid of x/y coordinates and returns it as a tibble.

**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)
```

---

**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 = "."`.

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)
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_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:
```
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:
```
num_as_char(1, sep = ",")
num_as_char(2, sym = ",")
num_as_char(3, sym = ",", n_dec = 0)
```

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

# Works for vectors:
```
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**

```r
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
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(""")
num_as_ordinal(Sys.Date())
num_as_ordinal(Sys.time())
num_as_ordinal(seq(1.99, 2.14, by = .01))
```

```
outliers

<table>
<thead>
<tr>
<th>id</th>
<th>sex</th>
<th>height</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

Description

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

Usage

`outliers`

Format

A tibble 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 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, table6, table7, table8, tb

---

**pal_ds4psy**  

*ds4psy default color palette.*

---

**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

\(\text{pal}_n\_sq\) returns \(n^2\) dedicated colors of a color palette \(\text{pal}\) (up to a maximum of \(n = \text{"all"}\) colors).

Usage

\[
\text{pal}_n\_sq(n = \text{"all"}, \text{pal} = \text{pal\_ds4psy})
\]

Arguments

- \(n\): The desired number colors of \(\text{pal}\) (as a number) or the character string \(\text{"all"}\) (to get all colors of \(\text{pal}\)). Default: \(n = \text{"all"}\).
- \(\text{pal}\): A color palette (as a data frame). Default: \(\text{pal} = \text{pal\_ds4psy}\).

Details

Use the more specialized function \texttt{unikn::usecol} for choosing \(n\) dedicated colors of a known color palette.

See Also

- \texttt{plot\_tiles} to plot tile plots.
- Other color objects and functions: \texttt{pal\_ds4psy}

Examples

\[
\begin{align*}
\text{pal}_n\_sq(1) & \quad \text{# 1 color: seeblau3} \\
\text{pal}_n\_sq(2) & \quad \text{# 4 colors} \\
\text{pal}_n\_sq(3) & \quad \text{# 9 colors (5: white)} \\
\text{pal}_n\_sq(4) & \quad \text{# 11 colors (6: white)}
\end{align*}
\]

pi_100k

Data: 100k digits of pi.

Description

\(\text{pi}_100k\) is a dataset containing the first 100k digits of pi.

Usage

\(\text{pi}_100k\)
plot_fn

Format
A character of $nchar(pi_{100k}) = 100001$.

Source
See TXT data at http://rpository.com/ds4psy/data/pi_{100k}.txt.
Original data at 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, 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

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td>A (natural) number. Default: x = NA.</td>
</tr>
<tr>
<td>y</td>
<td>A (decimal) number. Default: y = 1.</td>
</tr>
<tr>
<td>A</td>
<td>Boolean. Default: A = TRUE.</td>
</tr>
<tr>
<td>B</td>
<td>Boolean. Default: B = FALSE.</td>
</tr>
<tr>
<td>C</td>
<td>Boolean. Default: C = TRUE.</td>
</tr>
<tr>
<td>D</td>
<td>Boolean. Default: D = FALSE.</td>
</tr>
<tr>
<td>E</td>
<td>Boolean. Default: E = FALSE.</td>
</tr>
</tbody>
</table>
plot_fun

- **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.

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"))
```

---

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.
c Boolean. Default: c = TRUE.
d A (decimal) number. Default: d = 1.0.
e Boolean. Default: e = FALSE.
f Boolean. Default: f = 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). Note: Using colors of the unikn package by default.
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

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).
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 = 6, sort = FALSE) # random order
plot_n(n = 8, borders = FALSE) # no borders
plot_n(n = 10, lbl_tiles = TRUE) # with tile labels
plot_n(n = 10, lbl_title = TRUE) # with title label

# Set colors:
plot_n(n = 3, pal = c("forestgreen", "white", "black"),
       lbl_tiles = TRUE, sort = TRUE)
plot_n(n = 5, row = FALSE, 
       pal = c("orange", "white", "firebrick"),
       lbl_tiles = TRUE, lbl_title = TRUE, sort = TRUE)
plot_n(n = 10, 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 = 7, polar = TRUE, lbl_tiles = TRUE) # PIE with tile labels
plot_n(n = 7, 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 tibble 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 = "".

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::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`.


`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 tibble; `pal_ds4psy` for default color palette.

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

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) 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

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 x 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).
**plot_tiles**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>lbl_title</strong></td>
<td>Boolean: Add numeric label (of n) to plot? Default: <code>lbl_title = FALSE</code> (i.e., no title).</td>
</tr>
<tr>
<td><strong>polar</strong></td>
<td>Boolean: Plot on polar coordinates? Default: <code>polar = FALSE</code> (i.e., using fixed coordinates).</td>
</tr>
<tr>
<td><strong>rseed</strong></td>
<td>Random seed (number). Default: <code>rseed = NA</code> (using random seed).</td>
</tr>
<tr>
<td><strong>save</strong></td>
<td>Boolean: Save plot as png file? Default: <code>save = FALSE</code>.</td>
</tr>
<tr>
<td><strong>save_path</strong></td>
<td>Path to save plot (if <code>save = TRUE</code>). Default: <code>save_path = &quot;images/tiles&quot;</code>.</td>
</tr>
<tr>
<td><strong>prefix</strong></td>
<td>Prefix to plot name (if <code>save = TRUE</code>). Default: <code>prefix = &quot;&quot;</code>.</td>
</tr>
<tr>
<td><strong>suffix</strong></td>
<td>Suffix to plot name (if <code>save = TRUE</code>). Default: <code>suffix = &quot;&quot;</code>.</td>
</tr>
</tbody>
</table>

**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 = 6, sort = FALSE)  # random order
plot_tiles(n = 8, borders = FALSE)  # no borders
plot_tiles(n = 10, lbl_tiles = TRUE)  # with tile labels
plot_tiles(n = 10, lbl_title = TRUE)  # with title label

# Set colors:
plot_tiles(n = 3, pal = c("steelblue", "white", "black"),
           lbl_tiles = TRUE, sort = TRUE)
plot_tiles(n = 5, pal = c("orange", "white", "firebrick"),
           lbl_tiles = TRUE, lbl_title = TRUE,
           sort = TRUE)
plot_tiles(n = 10, 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 = 6, polar = TRUE, sort = FALSE)  # random order
plot_tiles(n = 8, polar = TRUE, borders = FALSE)  # no borders
plot_tiles(n = 10, polar = TRUE, lbl_tiles = TRUE)  # with tile labels
plot_tiles(n = 10, polar = TRUE, lbl_title = TRUE)  # with title label

plot_tiles(n = 4, sort = FALSE, borders = TRUE,
           border_col = "white", border_size = 2,
           polar = TRUE, rseed = 132)
```
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 tibble 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.

Source

Articles

posPsy_long


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**

- See *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, 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 tibble 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).
### Positive Psychology: Participant data.

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

#### Usage

`posPsy_p_info`

#### Format

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

#### Details

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>Participant ID.</td>
</tr>
<tr>
<td>intervention</td>
<td>Type of intervention: 3 positive psychology interventions (PPIs), plus 1 control condition: 1: &quot;Using signature strengths&quot;, 2: &quot;Three good things&quot;, 3: &quot;Gratitude visit&quot;, 4: &quot;Recording early memories&quot; (control condition).</td>
</tr>
<tr>
<td>sex</td>
<td>Sex: 1 = female, 2 = male.</td>
</tr>
<tr>
<td>age</td>
<td>Age (in years).</td>
</tr>
<tr>
<td>educ</td>
<td>Education level: Scale from 1: less than 12 years, to 5: postgraduate degree.</td>
</tr>
<tr>
<td>income</td>
<td>Income: Scale from 1: below average, to 3: above average.</td>
</tr>
</tbody>
</table>

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).
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 [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 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_wide.t3.t4.table6.table7.table8.tb`
Description

read_ascii parses text (from a file or from user input in Console) into a tibble 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 tibble 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 here package is used to determine the (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

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

## (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())
```

### 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

```r
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

\texttt{sample_times} draws a sample of \( n \) random times from a given range.

Usage

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

Arguments

\begin{itemize}
  \item \texttt{n} \quad \text{Number dates to draw. Default: } n = 1.
  \item \texttt{from} \quad \text{Earliest date (as string). Default: } from = "1970-01-01 00:00:00".
  \item \texttt{to} \quad \text{Latest date (as string). Default: } to = Sys.time().
\end{itemize}

Details

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

See Also

Other random functions: \texttt{coin()}, \texttt{dice_2()}, \texttt{dice()}, \texttt{sample_dates()}

Examples

\begin{itemize}
  \item \texttt{# Basics:}
  \texttt{sample_times()}
  \texttt{sample_times(n = 10)}
  \item \texttt{# Specific ranges:}
  \texttt{sort(sample_times(n = 10, from = (Sys.time() - 60)))} \quad \text{# within the last minute}
  \texttt{sort(sample_times(n = 10, from = (Sys.time() - 1 * 60 * 60)))} \quad \text{# within the last hour}
  \texttt{sort(sample_times(n = 10, from = Sys.time(), to = (Sys.time() + 1 * 60 * 60)))} \quad \text{# within the next hour}
  \texttt{sort(sample_times(n = 10, from = \texttt{"2020-01-01 00:00:00 CET"}, to = \texttt{"2020-01-01 00:00:01 CET"})} \quad \text{# within 1 sec range}
  \item \texttt{# Note: Oddity with \texttt{sample()}:}
  \texttt{sort(sample_times(n = 10, from = \texttt{"2020-01-01 00:00:00 CET"}, to = \texttt{"2020-01-01 00:00:00 CET")} \quad \text{# range of 0!}
  \item \texttt{# see sample(9:9, size = 10, replace = TRUE)}
\end{itemize}
Data table t3.

Description

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

Usage

t3

Format

A tibble 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, 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 tibble 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, table6, table7, table8, tb
### Data table6.

**Description**

table6 is a fictitious dataset to practice tidying data.

**Usage**

```r
table6
```

**Format**

A tibble 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`, `table7`, `table8`, `tb` 

---

### Data table7.

**Description**

table7 is a fictitious dataset to practice tidying data.

**Usage**

```r
table7
```

**Format**

A tibble 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.
Data table8.

Description

table8 is a fictitious dataset to practice tidying data.

Usage

table8

Format

A tibble 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, table6, table8, tb
Description

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

Usage

tb

Format

A tibble 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 Tibbles).

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

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, table6, table7, table8
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) +
transl33t translates text into leet (or l33t) slang given a set of rules.

Usage

```r
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

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

# Specify rules:
```r
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 = "hEllo world", in_case = "lo", out_case = "up",  
  rules = c("e" = "3", "l" = "1", "o" = "0"))  # e transl33ted

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 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
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")

Description

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

Usage

what_day(when = Sys.time(), unit = "week", 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.
- **unit**: Character: Unit of day? Possible values are "week","month","year". Default: unit = "week" (for day within week).
- **abbr**: Boolean: Return abbreviated? Default: abbr = FALSE.
- **as_integer**: Boolean: Return as integer? Default: as_integer = FALSE.

Details

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

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)
what_day(as_integer = 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, unit = "month", as_integer = TRUE)
```
```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.

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

```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)
```
```
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

```r
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 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 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):
ds <- 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)
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()

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, 8
  data_t1, 9
  data_t1_de, 9
  data_t1_tab, 10
  data_t2, 10
  data_t3, 11
  data_t4, 12
  exp_wide, 15
  falsePosPsy_all, 15
  fame, 17
  l33t_rul35, 19
  outliers, 22
  pal_ds4psy, 23
  pi_100k, 24
  posPsy_AHI_CESD, 34
  posPsy_long, 35
  posPsy_p_info, 36
  posPsy_wide, 37
  t3, 41
  t4, 41
  table6, 42
  table7, 42
  table8, 43
  tb, 44

capitalize, 3, 4, 6, 19, 38, 46
  caseflip, 3, 4, 6, 19, 38, 46
  coin, 4, 13, 14, 39, 40
  count_char, 3, 4, 5, 19, 38, 46
  cur_date, 6, 7, 47–52
  cur_time, 6, 7, 47–52

data_1, 8, 9–12, 15, 17, 23, 25, 35–37, 41–44
  data_2, 8, 9–12, 15, 17, 23, 25, 35–37, 41–44
  data_t1, 8, 9, 10–12, 15, 17, 23, 25, 35–37, 41–44
  data_t1_de, 8, 9, 10–12, 15, 17, 23, 25, 35–37, 41–44
  data_t1_tab, 8–10, 10, 12, 15, 17, 23, 25, 35–37, 41–44
  data_t2, 8–10, 10, 12, 15, 17, 23, 25, 35–37, 41–44
  data_t3, 8–11, 11, 12, 15, 17, 23, 25, 35–37, 41–44
  data_t4, 8–11, 12, 15, 17, 23, 25, 35–37, 41–44
dice, 5, 12, 13, 14, 39, 40
  dice_2, 5, 13, 13, 39, 40
  ds4psy.guide, 14

  exp_wide, 8–12, 15, 17, 23, 25, 35–37, 41–44
  falsePosPsy_all, 8–12, 15, 15, 17, 23, 25, 35–37, 41–44
  fame, 8–12, 15, 17, 17, 23, 25, 35–37, 41–44
  is.integer, 18
  is.wholenumber, 18, 20, 22

  l33t_rul35, 3, 4, 6, 19, 38, 46

  make_grid, 19

  num_as_char, 18, 20, 22
  num_as_ordinal, 18, 20, 21

  outliers, 8–12, 15, 17, 22, 25, 35–37, 41–44

  pal_ds4psy, 23, 24, 26–29, 31–33
  pal_n_sq, 23, 24
  pi_100k, 8–12, 15, 17, 23, 24, 35–37, 41–44
  plot_fn, 25, 27, 29, 31, 33, 45
  plot_fun, 26, 26, 29, 31, 33, 45
  plot_n, 26, 27, 28, 31, 33, 45
  plot_text, 6, 26, 27, 29, 30, 33, 38, 45
  plot_tiles, 24, 26, 27, 29, 31, 32, 45

53
posPsy_AHI_CESD, 8–12, 15, 17, 23, 25, 34, 35–37, 41–44
posPsy_long, 8–12, 15, 17, 23, 25, 35, 37, 41–44
posPsy_p_info, 8–12, 15, 17, 23, 25, 35, 36, 37, 41–44
posPsy_wide, 8–12, 15, 17, 23, 25, 35–37, 37, 41–44
read_ascii, 3, 4, 6, 19, 31, 38, 46
sample_dates, 5, 13, 14, 39, 40
sample_times, 5, 13, 14, 39, 40
t3, 8–12, 15, 17, 23, 25, 35–37, 41, 41, 42–44
t4, 8–12, 15, 17, 23, 25, 35–37, 41, 41, 42–44
table6, 8–12, 15, 17, 23, 25, 35–37, 41, 42, 43, 44
table7, 8–12, 15, 17, 23, 25, 35–37, 41, 42, 42, 43, 44
table8, 8–12, 15, 17, 23, 25, 35–37, 41–43, 43, 44
tb, 8–12, 15, 17, 23, 25, 35–37, 41–43, 44
theme_ds4psy, 26, 27, 29, 31, 33, 45
transl33t, 3, 4, 6, 19, 38, 46
what_date, 6, 7, 47, 48–52
what_day, 6, 7, 47, 48, 49–52
what_month, 6, 7, 47, 48, 49, 50–52
what_time, 6, 7, 47–49, 50, 51, 52
what_week, 6, 7, 47–50, 51, 52
what_year, 6, 7, 47–51, 52