Package ‘faux’

September 22, 2020

Title  Simulation for Factorial Designs
Version  0.0.1.5
Date  2020-09-11
Depends  R (&gt;= 3.2.4)
Imports  lme4, dplyr, ggplot2 (&gt;= 3.3.0), jsonlite, truncnorm
License  MIT + file LICENSE
Suggests  testthat (&gt;= 2.1.0), tidyr, knitr, rmarkdown, roxygen2, covr, cowplot, ggExtra, purrr, broom, broom.mixed, psych
VignetteBuilder  knitr
RoxygenNote  7.1.1
Encoding  UTF-8
LazyData  true
URL  https://github.com/debruine/faux
BugReports  https://github.com/debruine/faux/issues
NeedsCompilation  no
Author  Lisa DeBruine [aut, cre] (<https://orcid.org/0000-0002-7523-5539>), Anna Krystalli [ctb] (<https://orcid.org/0000-0002-2378-4915>), Andrew Heiss [ctb] (<https://orcid.org/0000-0002-3948-3914>)
Maintainer  Lisa DeBruine <debruine@gmail.com>
Repository  CRAN
Date/Publication  2020-09-22 06:30:13 UTC

R topics documented:

  average_r2tau_0 .................................................. 3
  check_design ..................................................... 3
<table>
<thead>
<tr>
<th>R topics documented:</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>check_mixed_design</td>
<td>5</td>
</tr>
<tr>
<td>codebook</td>
<td>5</td>
</tr>
<tr>
<td>codebook_interactive</td>
<td>6</td>
</tr>
<tr>
<td>cormat</td>
<td>7</td>
</tr>
<tr>
<td>cormat_from_triangle</td>
<td>8</td>
</tr>
<tr>
<td>faceratings</td>
<td>8</td>
</tr>
<tr>
<td>faux</td>
<td>9</td>
</tr>
<tr>
<td>faux_options</td>
<td>9</td>
</tr>
<tr>
<td>fix_name_labels</td>
<td>10</td>
</tr>
<tr>
<td>fr4</td>
<td>11</td>
</tr>
<tr>
<td>get_design_long</td>
<td>11</td>
</tr>
<tr>
<td>get_params</td>
<td>12</td>
</tr>
<tr>
<td>interactive_design</td>
<td>13</td>
</tr>
<tr>
<td>is_pos_def</td>
<td>13</td>
</tr>
<tr>
<td>json_design</td>
<td>14</td>
</tr>
<tr>
<td>long2wide</td>
<td>15</td>
</tr>
<tr>
<td>make_id</td>
<td>15</td>
</tr>
<tr>
<td>messy</td>
<td>16</td>
</tr>
<tr>
<td>nested_list</td>
<td>17</td>
</tr>
<tr>
<td>norm2beta</td>
<td>18</td>
</tr>
<tr>
<td>norm2binom</td>
<td>18</td>
</tr>
<tr>
<td>norm2likert</td>
<td>19</td>
</tr>
<tr>
<td>norm2pois</td>
<td>20</td>
</tr>
<tr>
<td>norm2trunc</td>
<td>21</td>
</tr>
<tr>
<td>norm2unif</td>
<td>21</td>
</tr>
<tr>
<td>plot_design</td>
<td>22</td>
</tr>
<tr>
<td>pos_def_limits</td>
<td>23</td>
</tr>
<tr>
<td>print.design</td>
<td>24</td>
</tr>
<tr>
<td>print.nested_list</td>
<td>24</td>
</tr>
<tr>
<td>print.psychds_codebook</td>
<td>25</td>
</tr>
<tr>
<td>readline_check</td>
<td>25</td>
</tr>
<tr>
<td>rnorm_multi</td>
<td>26</td>
</tr>
<tr>
<td>rnorm_pre</td>
<td>27</td>
</tr>
<tr>
<td>sample_from_pop</td>
<td>28</td>
</tr>
<tr>
<td>sim_data</td>
<td>28</td>
</tr>
<tr>
<td>sim_design</td>
<td>29</td>
</tr>
<tr>
<td>sim_df</td>
<td>30</td>
</tr>
<tr>
<td>sim_mixed_cc</td>
<td>31</td>
</tr>
<tr>
<td>sim_mixed_df</td>
<td>32</td>
</tr>
<tr>
<td>std_alpha2average_r</td>
<td>33</td>
</tr>
<tr>
<td>trunc2norm</td>
<td>34</td>
</tr>
<tr>
<td>unif2norm</td>
<td>35</td>
</tr>
<tr>
<td>unique_pairs</td>
<td>35</td>
</tr>
<tr>
<td>wide2long</td>
<td>36</td>
</tr>
</tbody>
</table>

**Index** 37
**Average r to Random Intercept SD**

Description

Average r to Random Intercept SD

Usage

```
average_r2tau_0(average_r, sigma)
```

Arguments

- `average_r`: The average inter-item correlation
- `sigma`: Total error variance

Value

The standard deviation of the random intercept

---

**check_design**

Validates the specified design

Description

Specify any number of within- and between-subject factors with any number of levels.

Usage

```
check_design(
    within = list(),
    between = list(),
    n = 100,
    mu = 0,
    sd = 1,
    r = 0,
    dv = list(y = "value"),
    id = list(id = "id"),
    vardesc = list(),
    plot = faux_options("plot"),
    design = NULL
)
```
check_design

Arguments

within a list of the within-subject factors
between a list of the between-subject factors
n the number of samples required
mu a vector giving the means of the variables
sd the standard deviations of the variables
r the correlations among the variables (can be a single number, full correlation
dv matrix as a matric or vector, or a vector of the upper right triangle of the corre-
dvlation matrix
id the name of the ID column list(id = "id")
vardesc a list of variable descriptions having the names of the within- and be-
plot tween-subject factors
whether to show a plot of the design
design a design list including within, between, n, mu, sd, r, dv, id

Details

Specify n for each between-subject cell; mu and sd for each cell, and r for the within-subject cells
each between-subject cell.

This function returns a validated design list for use in sim_data to simulate a data table with this
design, or to archive your design.

See vignette("sim_design", package = "faux") for details.

Value

list

Examples

within <- list(time = c("day", "night"))
between <- list(pet = c("dog", "cat"))
mu <- list(dog = 10, cat = 5)
vardesc <- list(time = "Time of Day", pet = "Type of Pet")
check_design(within, between, mu = mu, vardesc = vardesc)

between <- list(language = c("dutch", "thai"),
               pet = c("dog", "cat"))
mu <- list(dutch_dog = 12, dutch_cat = 7, thai_dog = 8, thai_cat = 3)
check_design(within, between, mu = mu)
check_mixed_design

Get random intercepts for subjects and items

Description
Get error terms from an existing data table.

Usage
check_mixed_design(data, dv = 1, sub_id = 2, item_id = 3, formula = NULL)

Arguments
- data: the existing tbl
- dv: the column name or index containing the DV
- sub_id: the column name or index for the subject IDs
- item_id: the column name or index for the item IDs
- formula: the formula to run in lmer (defaults to null model dv ~ 1 + (1|sub_id) + (1|item_id))

Value
a list of parameters

Examples
```r
des <- check_mixed_design(fr4, "rating", "rater_id", "face_id")
str(des[1:4])
```

codebook
Create PsychDS Codebook from Data

Description
See vignette("codebook", package = "faux") for details.

Usage
codebook(
  data,
  name = NULL,
  vardesc = list(),
  ...
  schemaVersion = "Psych-DS 0.1.0",
  return = c("json", "list", "data"),
  interactive = FALSE
)
**Arguments**

- **data**: The data frame to generate a codebook for
- **name**: The name of this dataset (if NULL, will be the same as 'data', limited to 64 characters)
- **vardesc**: Optional variable properties in the format of a named list of vectors (can be named or unnamed and in the same order as the data) from the options "description", "privacy", "dataType", "identifier", "minValue", "maxValue", "levels", "levelsOrdered", "na", "naValue", "alternateName", "privacy", "unitCode", "unitText"
- **...**: Further dataset properties (e.g., description, license, author, citation, funder, url, identifier, keywords, privacyPolicy)
- **schemaVersion**: defaults to "Psych-DS 0.1.0"
- **return**: Whether the output should be in JSON format (json), a list (list) or the reformatted data with the codebook as an attribute (data)
- **interactive**: Whether the function should prompt the user to describe columns and factor levels

**Value**

a list or json-formatted codebook, or reformatted data with the codebook as an attribute

**Examples**

```r
vardesc = list(
    description = c("Length of the sepal",
                    "Width of the sepal",
                    "Length of the petal",
                    "Width of the petal",
                    "The flower species"),
    type = c("float", "float", "float", "float", "string")
)
codebook(iris, vardesc = vardesc)
```

**Description**

Create a Psych-DS formatted codebook from data by answering questions interactively in the console.

**Usage**

```r
codebook_interactive(data, cb = NULL)
```
Arguments

data The data frame to generate a codebook for

cb The codebook in list format if already generated

Value

codebook list

cormat Make a correlation matrix

Description

cormat makes a correlation matrix from a single number, vars\*vars matrix, vars\*vars vector, or a vars\*{(vars-1)/2} vector.

Usage

cormat(cors = 0, vars = 3)

Arguments

cors the correlations among the variables (can be a single number, vars\*vars matrix, vars\*vars vector, or a vars\*{(vars-1)/2} vector)

vars the number of variables in the matrix

Value

matrix

Examples

cormat(.5, 3)
cormat(c( 1, .2, .3, .4,
    .2, 1, .5, .6,
    .3, .5, 1, .7,
    .4, .6, .7, 1), 4)
cormat(c(.2, .3, .4, .5, .6, .7), 4)
cormat_from_triangle  Make Correlation Matrix from Triangle

Description

cormat_from_triangle makes a correlation matrix from a vector of the upper right triangle

Usage

cormat_from_triangle(cors)

Arguments

cors the correlations among the variables as a vars*(vars-1)/2 vector

Value

matrix

Examples

cormat_from_triangle(c(.2, .3, .4, .5, .6, .7))

faceratings  Attractiveness ratings of faces

Description

A dataset containing attractiveness ratings (on a 1-7 scale from "much less attractiveness than average" to "much more attractive than average") for the neutral front faces from 2513 people (ages 17-90)

Usage

faceratings

Format

A data frame with 256326 rows and 9 variables:

rater_id rater’s ID
rater_sex rater’s sex (female, male, intersex, NA)
rater_age rater’s age (17-90 years)
rater_sexpref rater’s preferred sex for romantic relationships (either, men, neither, women, NA)
**faux**

**Description**

The faux package provides functions for simulating datasets with specified structure.

**faux_options**

*Set/get global faux options*

**Description**

Global afe options are used, for example, to set the default separator for cell names.

**Usage**

`faux_options(...)`

**Arguments**

`...`

One of four: (1) nothing, then returns all options as a list; (2) a name of an option element, then returns its value; (3) a name-value pair which sets the corresponding option to the new value (and returns nothing), (4) a list with option-value pairs which sets all the corresponding arguments.

**Value**

a list of options, values of an option, or nothing

---

**face_id** face’s ID

**face_sex** face’s sex (female, male)

**face_age** face’s age (in years)

**face_eth** face’s ethnic group

**rating** attractiveness rating on a scale from 1 (much less attractive than average) to 7 (much more attractive than average)

**Source**

https://figshare.com/articles/Face_Research_Lab_London_Set/5047666
**Examples**

faux_options() # see all options

faux_options("sep") # see value of faux.sep

## Not run:
# changes cell separator (e.g., A1.B2)
faux_options(sep = ".")

# changes cell separator back to default (e.g., A1_B2)
faux_options(sep = "_")

## End(Not run)

---

**fix_name_labels**

---

**Description**

Fixes if a factor list does not have named levels or has special characters in the names

**Usage**

```
fix_name_labels(x, pattern = NA, replacement = ".")
```

**Arguments**

- **x**: the vector or list to fix
- **pattern**: regex pattern to replace; defaults to non-word characters and the value of faux_options("sep") (default = ")
- **replacement**: the character to replace; defaults to . (or _ if faux_options("sep") == ".")

**Value**

a named list with fixed names

**Examples**

```r
source <- list("full.stop", " space ", "under_score", "plus+", "dash-", "tab\t", "line\nbreak")
fix_name_labels(source)
```
Attractiveness rating subset

Description
The faceratings dataset cut down for demos to the first 4 raters of each sex and sexpref and the first 4 faces of each sex and ethnicity with non-NA ages

Usage
fr4

Format
A data frame with 768 rows and 9 variables:
- rater_id  rater’s ID
- rater_sex  rater’s sex (female, male)
- rater_age  rater’s age (17.4-54.3 years)
- rater_sexpref  rater’s preferred sex for romantic relationships (either, men, women)
- face_id  face’s ID
- face_sex  face’s sex (female, male)
- face_age  face’s age (19-47 years)
- face_eth  face’s ethnic group (black, east_asian, west_asian, white)
- rating  attractiveness rating on a scale from 1 (much less attractive than average) to 7 (much more attractive than average)

Source
https://figshare.com/articles/Face_Research_Lab_London_Set/5047666

get_design_long  Get design from long data

Description
Makes a best guess at the design of a long-format data frame.

Usage
getDesignLong(
    data,
    dv = c(y = "score"),
    id = c(id = "id"),
    plot = faux_options("plot")
)
get_params

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>data</td>
<td>the data frame (in long format)</td>
</tr>
<tr>
<td>dv</td>
<td>the column name that identifies the DV</td>
</tr>
<tr>
<td>id</td>
<td>the column name(s) that identify a unit of analysis</td>
</tr>
<tr>
<td>plot</td>
<td>whether to show a plot of the design</td>
</tr>
</tbody>
</table>

Details

Finds all columns that contain a single value per unit of analysis (between factors), all columns that contain the same values per unit of analysis (within factors), and all columns that differ over units of analysis (dv, continuous factors)

Value

a design list

Description

get_params Generates a table of the correlations and means of numeric columns in a data frame

Usage

get_params(data, between = c(), within = c(), dv = "y", id = "id", digits = 2)

check_sim_stats(
  data,
  between = c(),
  within = c(),
  dv = "y",
  id = "id",
  digits = 2
)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>data</td>
<td>the existing tbl</td>
</tr>
<tr>
<td>between</td>
<td>a vector of column names for between-subject factors</td>
</tr>
<tr>
<td>within</td>
<td>a vector of column names for within-subject factors (if data is long)</td>
</tr>
<tr>
<td>dv</td>
<td>the column name of the dv (if data is long)</td>
</tr>
<tr>
<td>id</td>
<td>the column name(s) of the subject ID (if data is long)</td>
</tr>
<tr>
<td>digits</td>
<td>how many digits to round to (default = 2)</td>
</tr>
</tbody>
</table>
**interactive_design**

*Set design interactively*

**Description**

Set design interactively

**Usage**

```
interactive_design(output = c("faux"), plot = faux_options("plot"))
```

**Arguments**

- `output`: what type of design to output (faux)
- `plot`: whether to show a plot of the design

**Value**

list

**Examples**

```
if(interactive()){
  des <- interactive_design()
}
```

---

**is_pos_def**  
*Check a Matrix is Positive Definite*

**Description**

`is_pos_def` makes a correlation matrix from a vector

**Usage**

```
is_pos_def(cor_mat, tol = 1e-08)
```

**Arguments**

- `cor_mat`: a correlation matrix
- `tol`: the tolerance for comparing eigenvalues to 0
json_design

Value

logical value

Examples

\[
is\_pos\_def(matrix(c(1, 0.5, 0.5, 1), 2)) \# \text{returns TRUE}
\]
\[
is\_pos\_def(matrix(c(1, 0.9, 0.9, 
0.9, 1, -0.2, 
0.9, -0.2, 1), 3)) \# \text{returns FALSE}
\]

json_design

Convert design to JSON

Description

Convert a design list to JSON notation for archiving (e.g. in scienceverse)

Usage

\[
\text{json\_design}(\text{design}, \text{filename} = \text{NULL}, \text{digits} = 8, \text{pretty} = \text{FALSE}, \ldots)
\]

Arguments

design \hspace{1cm} \text{a design list including within, between, n, mu, sd, r, dv, id}
filename \hspace{1cm} \text{option name of file to save the json to}
digits \hspace{1cm} \text{number of digits to save}
pretty \hspace{1cm} \text{whether to print condensed or readable}
\ldots \hspace{1cm} \text{other options to send to jsonlite::toJSON}

Value

\text{a JSON string}

Examples

\[
de\leftarrow \text{check\_design}(2,2)
de\_\text{json\_design}(de)
de\_\text{json\_design}(de, \text{pretty} = \text{TRUE})
\]
long2wide

Convert data from long to wide format

Description
Convert data from long to wide format

Usage
long2wide(data, within = c(), between = c(), dv = "y", id = "id")

Arguments
data the tbl in long format
within the names of the within column(s)
between the names of between column(s) (optional)
dv the name of the DV (value) column
id the names of the column(s) for grouping observations

Value
a tbl in wide format

Examples
df_long <- sim_design(2, 2, long = TRUE)
long2wide(df_long, "A", "B")

make_id
Make ID

Description
Make IDs with fixed length and a prefix (e.g., S001, S002, ..., S100).

Usage
make_id(n = 100, prefix = "S", digits = 0, suffix = "")

Arguments
n the number of IDs to generate (or a vector of numbers)
prefix the prefix to the number (default "S")
digits the number of digits to use for the numeric part. Only used if this is larger than the largest number of digits in n.
suffix the suffix to the number (default ")")
Value

a vector of IDs

Examples

```r
make_id(20, "SUBJECT_")
make_id(10:30, digits = 3)
```

---

**mussy**

*Simulate missing data*

**Description**

Insert NA or another replacement value for some proportion of specified columns to simulate missing data.

**Usage**

```r
mussy(data, prop = 0, ..., replace = NA)
```

**Arguments**

- `data`: the tbl
- `prop`: the proportion of data to mess up
- `...`: the columns to mess up (as a vector of column names or numbers)
- `replace`: the replacement value (defaults to NA)

**Value**

the messed up table

**Examples**

```r
mussy(iris, 0.1, "Species", replace = "NO SPECIES")
mussy(iris, 0.5, 1:4)
```
Output a nested list in RMarkdown list format

Description

Output a nested list in RMarkdown list format

Usage

nested_list(x, pre = "", quote = "")

Arguments

x    The list
pre  Text to prefix to each line (e.g., if you want all lines indented 4 spaces to start, use " ")
quote Text to quote values with (e.g., use "" to make sure values are not parsed as markdown

Value

A character string

Examples

x <- list(
a = list(a1 = "Named", a2 = "List"),
b = list("Unnamed", "List"),
c = c(c1 = "Named", c2 = "Vector"),
d = c("Unnamed", "Vector"),
e = list(e1 = list("A", "B", "C"),
e2 = list(a = "A", b = "B"),
e3 = c("A", "B", "C"),
e4 = 100),
f = "not a list or vector"
)
nested_list(x)
norm2beta  

**Convert normal to beta**

**Description**

Convert normal to beta

**Usage**

```r
norm2beta(x, shape1, shape2, mu = mean(x), sd = stats::sd(x), ...)
```

**Arguments**

- `x`: the normally distributed vector
- `shape1, shape2`: non-negative parameters of the distribution to return
- `mu`: the mean of x (calculated from x if not given)
- `sd`: the SD of x (calculated from x if not given)
- `...`: further arguments to qbeta

**Value**

a vector with a beta distribution

**Examples**

```r
x <- rnorm(10000)
y <- norm2beta(x, 1, 3)
g <- ggplot2::ggplot() + ggplot2::geom_point(ggplot2::aes(x, y))
ggExtra::ggMarginal(g, type = "histogram")
```

---

norm2binom  

**Convert normal to binomial**

**Description**

Convert normal to binomial

**Usage**

```r
norm2binom(x, size = 1, prob = 0.5, mu = mean(x), sd = stats::sd(x))
```
norm2likert

Arguments

- **x**: the normally distributed vector
- **size**: number of trials (0 or more)
- **prob**: the probability of success on each trial (0 to 1)
- **mu**: the mean of x (calculated from x if not given)
- **sd**: the SD of x (calculated from x if not given)

Value

- a vector with a binomial distribution

Examples

```r
x <- rnorm(10000)
y <- norm2binom(x)
g <- ggplot2::ggplot() + ggplot2::geom_point(ggplot2::aes(x, y))
ggExtra::ggMarginal(g, type = "histogram")
```

Description

Convert normal to likert

Usage

```r
norm2likert(x, prob, mu = mean(x), sd = stats::sd(x))
```

Arguments

- **x**: the normally distributed vector
- **prob**: a vector of probabilities
- **mu**: the mean of x (calculated from x if not given)
- **sd**: the SD of x (calculated from x if not given)

Value

- a vector with a specified distribution
Examples

```r
x <- rnorm(10000)
y <- norm2likert(x, c(.1, .2, .35, .2, .1, .05))
g <- ggplot2::ggplot() + ggplot2::geom_point(ggplot2::aes(x, y))
ggExtra::ggMarginal(g, type = "histogram")

y <- norm2likert(x, c(.4, .3, .2, .1))
g <- ggplot2::ggplot() + ggplot2::geom_point(ggplot2::aes(x, y))
ggExtra::ggMarginal(g, type = "histogram")
```

---

**norm2pois**

Convert normal to poisson

**Description**

Convert normal to poisson

**Usage**

```r
norm2pois(x, lambda, mu = mean(x), sd = stats::sd(x))
```

**Arguments**

- `x`: the normally distributed vector
- `lambda`: the mean of the distribution to return
- `mu`: the mean of `x` (calculated from `x` if not given)
- `sd`: the SD of `x` (calculated from `x` if not given)

**Value**

a vector with a poisson distribution

**Examples**

```r
x <- rnorm(10000)
y <- norm2pois(x, 2)
g <- ggplot2::ggplot() + ggplot2::geom_point(ggplot2::aes(x, y))
ggExtra::ggMarginal(g, type = "histogram")
```
norm2trunc

Convert normal to truncated normal

Description
Convert a normal (gaussian) distribution to a truncated normal distribution with specified minimum and maximum

Usage
norm2trunc(x, min = -Inf, max = Inf, mu = mean(x), sd = stats::sd(x))

Arguments
x the normally distributed vector
min the minimum of the truncated distribution to return
max the maximum of the truncated distribution to return
mu the mean of the distribution to return (calculated from x if not given)
sd the SD of the distribution to return (calculated from x if not given)

Value
a vector with a uniform distribution

Examples
x <- rnorm(10000)
y <- norm2trunc(x, 1, 7, 3.5, 2)
g <- ggplot2::ggplot() + ggplot2::geom_point(ggplot2::aes(x, y))
ggExtra::ggMarginal(g, type = "histogram")

norm2unif

Convert normal to uniform

Description
Convert a normal (gaussian) distribution to a uniform distribution with specified minimum and maximum

Usage
norm2unif(x, min = 0, max = 1, mu = mean(x), sd = stats::sd(x))
Arguments

- **x**: the normally distributed vector
- **min**: the minimum of the uniform distribution to return
- **max**: the maximum of the uniform distribution to return
- **mu**: the mean of x (calculated from x if not given)
- **sd**: the SD of x (calculated from x if not given)

**Value**

A vector with a uniform distribution

**Examples**

```r
x <- rnorm(10000)
y <- norm2unif(x)
g <- ggplot2::ggplot() + ggplot2::geom_point(ggplot2::aes(x, y))
ggExtra::ggMarginal(g, type = "histogram")
```

**Description**

Plots the specified within and between design. See vignette("plots", package = "faux") for examples and details.

**Usage**

```r
plot_design(x, ..., geoms = NULL, palette = "Dark2")
```

```r
### S3 method for class 'design'
plot(x, ...)

### S3 method for class 'faux'
plot(x, ...)
```

**Arguments**

- **x**: A list of design parameters created by check_design() or a data tbl (in long format)
- **...**: A list of factor names to determine visualisation (see vignette)
- **geoms**: A list of ggplot2 geoms to display, defaults to "pointrangeSD" (mean ± 1SD) for designs and c("violin", "box") for data, options are: pointrangeSD, pointrangeSE, violin, box, jitter
- **palette**: A brewer palette, defaults to "Dark2"
Value

plot

Methods (by generic)

• plot: Plotting from a faux design list
• plot: Plotting from a faux data table

Examples

```r
within <- list(time = c("day", "night"))
between <- list(pet = c("dog", "cat"))
des <- check_design(within, between, plot = FALSE)
plot_design(des)

data <- sim_design(within, between, plot = FALSE)
plot_design(data)
```

---

**pos_def_limits**

*Limits on Missing Value for Positive Definite Matrix*

Description

`pos_def_limits` returns min and max possible values for a positive definite matrix with a specified missing value.

Usage

```r
pos_def_limits(..., steps = 0.01, tol = 1e-08)
```

Arguments

... the correlations among the variables as a `vars*(vars-1)/2` vector
steps the tolerance for min and max values
tol the tolerance for comparing eigenvalues to 0

Value

dataframe with min and max values

Examples

```r
pos_def_limits(.8, .2, NA)
```
### print.design

#### Description

Print Design List

#### Usage

```r
## S3 method for class 'design'
print(x, ...)
```

#### Arguments

- `x` The design list
- `...` Additional parameters for print

---

### print.nested_list

#### Description

Print Nested List

#### Usage

```r
## S3 method for class 'nested_list'
print(x, ...)
```

#### Arguments

- `x` The nested_list string
- `...` Additional parameters for print
print.psychds_codebook

Print Codebook Object

Description
Print Codebook Object

Usage
## S3 method for class 'psychds_codebook'
print(x, ...)

Arguments
x The psychds_codebook list
... Additional parameters for print

readline_check

Check readline input

Description
Check readline input

Usage
readline_check(
prompt,
type = c("numeric", "integer", "length", "grep"),
min = -Inf,
max = Inf,
warning = NULL,
default = NULL,
...
)

Arguments
prompt the prompt for readline
type what type of check to perform, one of c("numeric", "integer", "length", "grep")
min the minimum value
max the maximum value
warning an optional custom warning message
rnorm_multi

Multiple correlated normal distributions

Description

Make normally distributed vectors with specified relationships. See vignette("rnorm_multi", package = "faux") for details.

Usage

rnorm_multi(
  n,
  vars = NULL,
  mu = 0,
  sd = 1,
  r = 0,
  varnames = NULL,
  empirical = FALSE,
  as.matrix = FALSE,
  seed = NULL
)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>n</td>
<td>the number of samples required</td>
</tr>
<tr>
<td>vars</td>
<td>the number of variables to return</td>
</tr>
<tr>
<td>mu</td>
<td>a vector giving the means of the variables (numeric vector of length 1 or vars)</td>
</tr>
<tr>
<td>sd</td>
<td>the standard deviations of the variables (numeric vector of length 1 or vars)</td>
</tr>
</tbody>
</table>

Value

the validated result of readline

Examples

```r
if(interactive()){
  readline_check("Type a number: ", "numeric")
  readline_check("Type two characters: ", "length", min = 2, max = 2)
  readline_check("Type at least 3 characters: ", "length", min = 3)
  readline_check("Type no more than 4 characters: ", "length", max = 44)
  readline_check("Type a letter and a number: ", "grep", pattern = "^[a-zA-Z]\d$")
}
```
### Description

**rnorm_pre** Produces a random normally distributed vector with the specified correlation to an existing vector.

### Usage

```
rnorm_pre(x, mu = 0, sd = 1, r = 0, empirical = FALSE)
```

### Arguments

- **x**: the existing vector
- **mu**: desired mean of returned vector
- **sd**: desired SD of returned vector
- **r**: desired correlation between existing and returned vectors
- **empirical**: logical. If true, mu, sd and r specify the empirical not population mean, sd and covariance

### Value

- **vector**
Examples

```r
v1 <- rnorm(10)
v2 <- rnorm_pre(v1, 0, 1, 0.5)
cor(v1, v2)
```

---

### sample_from_pop

**Sample Parameters from Population Parameters**

**Description**

Sample Parameters from Population Parameters

**Usage**

```r
sample_from_pop(n = 100, mu = 0, sd = 1, r = 0)
```

**Arguments**

- `n`: sample size
- `mu`: population mean
- `sd`: population SD
- `r`: population r

**Value**

list of sample parameters (mu, sd, r)

**Examples**

```r
sample_from_pop(10)
```

---

### sim_data

**Simulate data from design (internal)**

**Description**

Simulate data from design (internal)

**Usage**

```r
sim_data(
  design,
  empirical = FALSE,
  long = FALSE,
  rep = 1,
  sep = faux_options("sep"),
  seed = NULL
)
```
### Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>design</code></td>
<td>A list of design parameters created by <code>check_design()</code></td>
</tr>
<tr>
<td><code>empirical</code></td>
<td>logical. If true, mu, sd and r specify the empirical not population mean, sd and covariance</td>
</tr>
<tr>
<td><code>long</code></td>
<td>Whether the returned tbl is in wide (default = FALSE) or long (TRUE) format</td>
</tr>
<tr>
<td><code>rep</code></td>
<td>the number of data frames to return (default 1); if greater than 1, the returned data frame is nested by rep</td>
</tr>
<tr>
<td><code>sep</code></td>
<td>separator for within-columns, defaults to <code>_</code></td>
</tr>
<tr>
<td><code>seed</code></td>
<td>DEPRECATED use <code>set.seed()</code> instead</td>
</tr>
</tbody>
</table>

### Value

a tbl

---

### `sim_design`

*Simulate data from design*

### Description

Generates a data table with a specified within and between design. See `vignette("sim_design", package = "faux")` for examples and details.

### Usage

```r
sim_design(
  within = list(),
  between = list(),
  n = 100,
  mu = 0,
  sd = 1,
  r = 0,
  empirical = FALSE,
  long = FALSE,
  dv = list(y = "value"),
  id = list(id = "id"),
  plot = faux_options("plot"),
  interactive = FALSE,
  design = NULL,
  rep = 1,
  seed = NULL
)
```
**Arguments**

- `within`: a list of the within-subject factors
- `between`: a list of the between-subject factors
- `n`: the number of samples required
- `mu`: the means of the variables
- `sd`: the standard deviations of the variables
- `r`: the correlations among the variables (can be a single number, full correlation matrix as a matrix or vector, or a vector of the upper right triangle of the correlation matrix)
- `empirical`: logical. If true, mu, sd and r specify the empirical not population mean, sd and covariance
- `long`: Whether the returned tbl is in wide (default = FALSE) or long (TRUE) format
- `dv`: the name of the dv for long plots (defaults to y)
- `id`: the name of the id column (defaults to id)
- `plot`: whether to show a plot of the design
- `interactive`: whether to run the function interactively
- `design`: a design list including within, between, n, mu, sd, r, dv, id
- `rep`: the number of data frames to return (default 1); if greater than 1, the returned data frame is nested by rep
- `seed`: DEPRECATED use set.seed() instead

**Value**

- a tbl

---

**sim_df**

*Simulate an existing dataframe*

**Description**

Produces a data table with the same distributions and correlations as an existing data table. Only returns numeric columns and simulates all numeric variables from a continuous normal distribution (for now).

**Usage**

```r
sim_df(
  data,
  n = 100,
  within = c(),
  between = c(),
  id = "id",
)```
Arguments

data the existing tbl (must be in wide format)
n the number of samples to return per group
within a list of the within-subject columns
between a list of the between-subject columns
id the names of the column(s) for grouping observations
dv the name of the DV (value) column
empirical Should the returned data have these exact parameters? (versus be sampled from a population with these parameters)
long whether to return the data table in long format
seed DEPRECATED use set.seed() instead

Details

See vignette("sim_df", package = "faux") for details.

Value

a tbl

Examples

iris100 <- sim_df(iris, 100)
iris_species <- sim_df(iris, 100, between = "Species")

Description

Makes a basic cross-classified design with random intercepts for subjects and items. See vignette("sim_mixed", package = "faux") for examples and details.
Usage

```r
sim_mixed_cc(
  sub_n = 100,
  item_n = 20,
  grand_i = 0,
  sub_sd = 1,
  item_sd = 1,
  error_sd = 1,
  empirical = FALSE,
  seed = NULL
)
```

Arguments

- `sub_n`: the number of subjects
- `item_n`: the number of items
- `grand_i`: the grand intercept (overall mean)
- `sub_sd`: the SD of subject random intercepts (or a `sub_n`-length named vector of random intercepts for each subject)
- `item_sd`: the SD of item random intercepts (or an `item_n`-length named vector of random intercepts for each item)
- `error_sd`: the SD of the error term
- `empirical`: Should the returned data have these exact parameters? (versus be sampled from a population with these parameters)
- `seed`: DEPRECATED use `set.seed()` instead

Value

a tbl

Examples

```r
sim_mixed_cc(10, 10)
```

---

**sim_mixed_df**

*Generate a mixed design from existing data*

**Description**

`sim_mixed_df()` produces a data table with the same distributions of by-subject and by-item random intercepts as an existing data table.
Usage
```
sim_mixed_df(
    data,
    sub_n = NULL,
    item_n = NULL,
    dv = "y",
    sub_id = "sub_id",
    item_id = "item_id"
)
```

Arguments
- **data**: the existing tbl
- **sub_n**: the number of subjects to simulate (if NULL, returns data for the same subjects)
- **item_n**: the number of items to simulate (if NULL, returns data for the same items)
- **dv**: the column name or index containing the DV
- **sub_id**: the column name or index for the subject IDs
- **item_id**: the column name or index for the item IDs

Value
- a tbl

Examples
```
sim_mixed_df(faceratings, 10, 10, "rating", "rater_id", "face_id")
```

---

**std_alpha2average_r**  
*Standardized Alpha to Average R*

Description

Standardized Alpha to Average R

Usage
```
std_alpha2average_r(std_alpha, n)
```

Arguments
- **std_alpha**: The standarized alpha
- **n**: The number of items

Value

The average inter-item correlation
Examples

std_alpha2average_r(.8, 10)

```
trunc2norm
Convert truncated normal to normal
```

Description

Convert a truncated normal distribution to a normal (gaussian) distribution

Usage

trunc2norm(x, min = NA, max = NA, mu = NA, sd = NA)

Arguments

- **x**: the truncated normally distributed vector
- **min**: the minimum of the truncated distribution (calculated from x if not given)
- **max**: the maximum of the truncated distribution (calculated from x if not given)
- **mu**: the mean of the distribution to return (calculated from x if not given)
- **sd**: the SD of the distribution to return (calculated from x if not given)

Value

a vector with a uniform distribution

Examples

x <- truncnorm::rtruncnorm(10000, 1, 7, 3.5, 2)
y <- trunc2norm(x, 1, 7)
g <- ggplot2::ggplot() + ggplot2::geom_point(ggplot2::aes(x, y))
ggExtra::ggMarginal(g, type = "histogram")
unif2norm

Convert normal to uniform

Description

Convert a uniform distribution to a normal (gaussian) distribution with specified mu and sd

Usage

unif2norm(x, mu = 0, sd = 1, min = NULL, max = NULL)

Arguments

x the uniformly distributed vector
mu the mean of the normal distribution to return
sd the SD of the normal distribution to return
min the minimum possible value of x (calculated from x if not given)
max the maximum possible value of x (calculated from x if not given)

Value

a vector with a gaussian distribution

Examples

x <- runif(10000)
y <- unif2norm(x)
g <- ggplot2::ggplot() + ggplot2::geom_point(ggplot2::aes(x, y))
ggExtra::ggMarginal(g, type = "histogram")

unique_pairs

Make unique pairs of level names for correlations

Description

Make unique pairs of level names for correlations

Usage

unique_pairs(v)

Arguments

v a vector of level names or a number of levels
Value
a vector of all unique pairs

Examples
unique_pairs(c("O", "C", "E", "A", "N"))
unique_pairs(3)

---

wide2long

Convert data from wide to long format

Description
Convert data from wide to long format

Usage
wide2long(
  data,
  within_factors = c(),
  within_cols = c(),
  dv = "y",
  id = "id",
  sep = faux_options("sep")
)

Arguments
data the tbl in wide format
within_factors the names of the within factors
within_cols the names (or indices) of the within-subject (value) columns
dv the name of the dv column (defaults to "y")
id the name of the ID column(s) if they don’t exist, a new column will be made (defaults to ("id")
sep separator for within-columns (to be used in strsplit, so can be regex), defaults to ""

Value
a tbl in long format

Examples
wide2long(iris, c("Feature", "Measure"), 1:4, sep = "\\")
Index

* datasets
  faceratings, 8
  fr4, 11

* package
  faux, 9

average_r2tau_0, 3
check_design, 3
check_mixed_design, 5
check_sim_stats (get_params), 12
codebook, 5
codebook_interactive, 6
cormat, 7
cormat_from_triangle, 8
faceratings, 8
fau, 9
faux_options, 9
fix_name_labels, 10
fr4, 11
get_design_long, 11
get_params, 12
interactive_design, 13
is_pos_def, 13
json_design, 14
long2wide, 15
make_id, 15
messy, 16
nested_list, 17
norm2beta, 18
norm2binom, 18
norm2likert, 19
norm2pois, 20
norm2trunc, 21
norm2unif, 21
plot.design (plot_design), 22
plot.faux (plot_design), 22
plot_design, 22
pos_def_limits, 23
print.design, 24
print.nested_list, 24
print.psychds_codebook, 25
readline_check, 25
rnorm_multi, 26
rnorm_pre, 27
sample_from_pop, 28
sim_data, 28
sim_design, 29
sim_df, 30
sim_mixed_cc, 31
sim_mixed_df, 32
std_alpha2average_r, 33
trunc2norm, 34
unif2norm, 35
unique_pairs, 35
wide2long, 36