

Package ‘LikertMakeR’

November 26, 2025

Type Package

Title Synthesise and Correlate Likert Scale and Rating-Scale Data
Based on Summary Statistics

Version 1.3.0

Description Generate and correlate synthetic Likert and rating-scale data with predefined means, standard deviations, Cronbach's Alpha, Factor Loading table, coefficients, and other summary statistics. Worked examples and documentation are available in the package articles, accessible via the package website, [<https://winzarh.github.io/LikertMakeR/>](https://winzarh.github.io/LikertMakeR/).

License MIT + file LICENSE

URL <https://github.com/WinzarH/LikertMakeR/>,
<https://winzarh.github.io/LikertMakeR/>

BugReports <https://github.com/WinzarH/LikertMakeR/issues>

Depends R (>= 4.2.0)

Imports dplyr, gtools, Matrix, Rcpp

Suggests kableExtra, knitr, psych, psychTools, rmarkdown, rosetta, testthat (>= 3.0.0), tibble, tidyr

LinkingTo Rcpp, RcppArmadillo

VignetteBuilder knitr

Config/Needs/website rmarkdown

Config/testthat/edition 3

Encoding UTF-8

Language en-GB

RoxygenNote 7.3.3

NeedsCompilation yes

Author Hume Winzar [cre, aut] (ORCID: [<https://orcid.org/0000-0001-7475-2641>](https://orcid.org/0000-0001-7475-2641))

Maintainer Hume Winzar <winzar@gmail.com>

Repository CRAN

Date/Publication 2025-11-26 01:22:04 UTC

Contents

alpha	2
correlateScales	3
eigenvalues	5
lcor	6
lexact	7
lfast	8
makeCorrAlpha	10
makeCorrLoadings	12
makeItems	14
makeItemsScale	16
makePaired	19
makeRepeated	21
makeScales	24
makeScalesRegression	27
print.makeScalesRegression	30
Index	31

alpha	<i>Calculate Cronbach's Alpha from a correlation matrix or dataframe</i>
-------	--

Description

alpha() calculates Cronbach's Alpha from a given correlation matrix or a given dataframe.

Usage

```
alpha(cormatrix = NULL, data = NULL)
```

Arguments

cormatrix	(real) a square symmetrical matrix with values ranging from -1 to +1 and '1' in the diagonal
data	(real) a dataframe or matrix

Value

a single value

Examples

```
## Sample data frame
df <- data.frame(
  V1 = c(4, 2, 4, 3, 2, 2, 2, 1),
  V2 = c(4, 1, 3, 4, 4, 3, 2, 3),
  V3 = c(4, 1, 3, 5, 4, 1, 4, 2),
  V4 = c(4, 3, 4, 5, 3, 3, 3, 3)
)

## example correlation matrix
corMat <- matrix(
  c(
    1.00, 0.35, 0.45, 0.70,
    0.35, 1.00, 0.60, 0.55,
    0.45, 0.60, 1.00, 0.65,
    0.70, 0.55, 0.65, 1.00
  ),
  nrow = 4, ncol = 4
)

## apply function examples

alpha(cormatrix = corMat)

alpha(, df)

alpha(corMat, df)
```

correlateScales	<i>Dataframe of correlated scales from different dataframes of scale items</i>
-----------------	--

Description

correlateScales() creates a dataframe of scale items representing correlated constructs, as one might find in a completed questionnaire.

Usage

```
correlateScales(dataframes, scalecors)
```

Arguments

dataframes	a list of 'k' dataframes to be rearranged and combined
scalecors	target correlation matrix - should be a symmetric $k \times k$ positive-semi-definite matrix, where 'k' is the number of dataframes

Details

Correlated rating-scale items generally are summed or averaged to create a measure of an "unobservable", or "latent", construct. `correlateScales()` takes several such dataframes of rating-scale items and rearranges their rows so that the scales are correlated according to a predefined correlation matrix. Univariate statistics for each dataframe of rating-scale items do not change, but their correlations with rating-scale items in other dataframes do.

Value

Returns a dataframe whose columns are taken from the starter dataframes and whose summated values are correlated according to a user-specified correlation matrix

Examples

```
## three attitudes and a behavioural intention
n <- 32
lower <- 1
upper <- 5

### attitude #1
cor_1 <- makeCorrAlpha(items = 4, alpha = 0.90)
means_1 <- c(2.5, 2.5, 3.0, 3.5)
sds_1 <- c(0.9, 1.0, 0.9, 1.0)

Att_1 <- makeItems(
  n = n, means = means_1, sds = sds_1,
  lowerbound = rep(lower, 4), upperbound = rep(upper, 4),
  cormatrix = cor_1
)

### attitude #2
cor_2 <- makeCorrAlpha(items = 5, alpha = 0.85)
means_2 <- c(2.5, 2.5, 3.0, 3.0, 3.5)
sds_2 <- c(1.0, 1.0, 0.9, 1.0, 1.5)

Att_2 <- makeItems(
  n = n, means = means_2, sds = sds_2,
  lowerbound = rep(lower, 5), upperbound = rep(upper, 5),
  cormatrix = cor_2
)

### attitude #3
cor_3 <- makeCorrAlpha(items = 6, alpha = 0.75)
means_3 <- c(2.5, 2.5, 3.0, 3.0, 3.5, 3.5)
sds_3 <- c(1.0, 1.5, 1.0, 1.5, 1.0, 1.5)

Att_3 <- makeItems(
  n = n, means = means_3, sds = sds_3,
  lowerbound = rep(lower, 6), upperbound = rep(upper, 6),
```

```
cormatrix = cor_3
)

### behavioural intention
intent <- lfast(n, mean = 3.0, sd = 3, lowerbound = 0, upperbound = 10) |>
  data.frame()
names(intent) <- "int"

### target scale correlation matrix
scale_cors <- matrix(
  c(
    1.0, 0.6, 0.5, 0.3,
    0.6, 1.0, 0.4, 0.2,
    0.5, 0.4, 1.0, 0.1,
    0.3, 0.2, 0.1, 1.0
  ),
  nrow = 4
)

data_frames <- list("A1" = Att_1, "A2" = Att_2, "A3" = Att_3, "Int" = intent)

### apply the function
my_correlated_scales <- correlateScales(
  dataframes = data_frames,
  scalecors = scale_cors
)
head(my_correlated_scales)
```

eigenvalues

calculate eigenvalues of a correlation matrix with optional scree plot

Description

`eigenvalues()` calculates eigenvalues of a correlation matrix and optionally produces a scree plot.

Usage

```
eigenvalues(cormatrix, scree = FALSE)
```

Arguments

<code>cormatrix</code>	(real, matrix) a correlation matrix
<code>scree</code>	(logical) default = FALSE. If TRUE (or 1), then <code>eigenvalues()</code> produces a scree plot to illustrate the eigenvalues

Value

a vector of eigenvalues
 report on positive-definite status of cormatrix

Examples

```
## define parameters

correlationMatrix <- matrix(
  c(
    1.00, 0.25, 0.35, 0.40,
    0.25, 1.00, 0.70, 0.75,
    0.35, 0.70, 1.00, 0.80,
    0.40, 0.75, 0.80, 1.00
  ),
  nrow = 4, ncol = 4
)

## apply function

evals <- eigenvalues(cormatrix = correlationMatrix)
evals <- eigenvalues(correlationMatrix, 1)
```

lcor	<i>Rearrange elements in each column of a data-frame to fit a predefined correlation matrix</i>
------	---

Description

lcor() rearranges values in each column of a data-frame so that columns are correlated to match a predefined correlation matrix.

Usage

```
lcor(data, target, passes = 10)
```

Arguments

data	dataframe that is to be rearranged
target	target correlation matrix. Must have same dimensions as number of columns in data-frame.
passes	Number of optimization passes (default = 10). Increasing this value <i>MAY</i> improve results if n-columns (target correlation matrix dimensions) are many. Decreasing the value for 'passes' is faster but may decrease accuracy.

Details

Values in a column do not change, so univariate statistics remain the same.

Value

Returns a dataframe whose column-wise correlations approximate a user-specified correlation matrix

Examples

```
## parameters
n <- 32
lowerbound <- 1
upperbound <- 5
items <- 5

mydat3 <- data.frame(
  x1 = lfast(n, 2.5, 0.75, lowerbound, upperbound, items),
  x2 = lfast(n, 3.0, 1.50, lowerbound, upperbound, items),
  x3 = lfast(n, 3.5, 1.00, lowerbound, upperbound, items)
)

cor(mydat3) |> round(3)

tgt3 <- matrix(
  c(
    1.00, 0.50, 0.75,
    0.50, 1.00, 0.25,
    0.75, 0.25, 1.00
  ),
  nrow = 3, ncol = 3
)

## apply function
new3 <- lcor(mydat3, tgt3)

## test output
cor(new3) |> round(3)
```

lexact

Deprecated. Use lfast() instead

Description

lexact is DEPRECATED. Replaced in LikertMakeR Version 0.4.0 by new version of lfast.

lexact remains as a legacy for earlier package users. It is now just a wrapper for lfast

Previously, lexact used a Differential Evolution (DE) algorithm to find an optimum solution with desired mean and standard deviation, but we found that the updated lfast function is much faster and just as accurate.

Also the package is much less bulky.

Usage

```
lexact(n, mean, sd, lowerbound, upperbound, items = 1)
```

Arguments

n	(positive, int) number of observations to generate
mean	(real) target mean
sd	(real) target standard deviation
lowerbound	(positive, int) lower bound
upperbound	(positive, int) upper bound
items	(positive, int) number of items in the rating scale.

Value

a vector of simulated data approximating user-specified conditions.

Examples

```
x <- lexact(  
  n = 256,  
  mean = 4.0,  
  sd = 1.0,  
  lowerbound = 1,  
  upperbound = 7,  
  items = 6  
)  
  
x <- lexact(256, 2, 1.8, 0, 10)
```

lfast

Synthesise rating-scale data with predefined mean and standard deviation

Description

lfast() applies a simple Evolutionary Algorithm to find a vector that best fits the desired moments.

lfast() generates random discrete values from a scaled Beta distribution so the data replicate an ordinal rating scale - for example, a Likert scale made from multiple items (questions) or 0-10 likelihood-of-purchase scale. Data generated are generally consistent with real data, as shown in the *lfast() validation article*.

Usage

```
lfast(n, mean, sd, lowerbound, upperbound, items = 1, precision = 0)
```

Arguments

n	(positive, int) number of observations to generate
mean	(real) target mean, between upper and lower bounds
sd	(positive, real) target standard deviation
lowerbound	(int) lower bound (e.g. '1' for a 1-5 rating scale)
upperbound	(int) upper bound (e.g. '5' for a 1-5 rating scale)
items	(positive, int) number of items in the rating scale. Default = 1
precision	(positive, real) can relax the level of accuracy required. (e.g. '1' generally generates a vector with moments correct within '0.025', '2' generally within '0.05') Default = 0

Value

a vector approximating user-specified conditions.

See Also

- lfast validation article: https://winzarh.github.io/LikertMakeR/articles/lfast_validation.html
- Package website: <https://winzarh.github.io/LikertMakeR/>

Examples

```
## six-item 1-7 rating scale
x <- lfast(
  n = 256,
  mean = 4.0,
  sd = 1.25,
  lowerbound = 1,
  upperbound = 7,
  items = 6
)

## five-item -3 to +3 rating scale
x <- lfast(
  n = 64,
  mean = 0.025,
  sd = 1.25,
  lowerbound = -3,
  upperbound = 3,
  items = 5
)

## four-item 1-5 rating scale with medium variation
```

```
x <- lfast(
  n = 128,
  mean = 3.0,
  sd = 1.00,
  lowerbound = 1,
  upperbound = 5,
  items = 4,
  precision = 5
)

## eleven-point 'likelihood of purchase' scale
x <- lfast(256, 3, 3.0, 0, 10)
```

makeCorrAlpha

Correlation matrix from Cronbach's Alpha

Description

makeCorrAlpha() generates a random correlation matrix of given dimensions and predefined *Cronbach's Alpha*.

Such a correlation matrix can be applied to the [makeItems\(\)](#) function to generate synthetic data with the predefined alpha.

Usage

```
makeCorrAlpha(
  items,
  alpha,
  variance = 0.5,
  precision = 0,
  sort_cors = FALSE,
  diagnostics = FALSE
)
```

Arguments

items	(positive, int) matrix dimensions: number of rows & columns to generate
alpha	(real) target <i>Cronbach's Alpha</i> (usually positive, must be between about -0.3 and +1)
variance	(positive, real) Default = 0.5. User-provided standard deviation of values sampled from a normally-distributed log transformation. Caution: Larger values increase chance of a non-positive-definite matrix. 'TRUE' is faster, but produces less natural output. Default = FALSE
precision	(positive, real) Default = 0. User-defined value ranging from '0' to '3' to add some random variation around the target <i>Cronbach's Alpha</i> . '0' gives an exact alpha (to two decimal places)

sort_cors	(logical) If 'TRUE', sorts the correlation coefficients in the final correlation matrix. Similar to an earlier version of this function.
diagnostics	(logical) If 'TRUE', returns a list containing the correlation matrix and a diagnostics list (target/achieved alpha, average inter-item correlation, eigenvalues, PD flag, and key arguments). If 'FALSE' (default), returns the correlation matrix only.

Value

If 'diagnostics = FALSE', a $k \times k$ correlation matrix. If 'diagnostics = TRUE', a list with components:

R $k \times k$ correlation matrix

diagnostics list of summary statistics

Note

Random values generated by makeCorrAlpha() are highly volatile. makeCorrAlpha() may not generate a feasible (positive-definite) correlation matrix, especially when

- variance is high relative to
 - desired Alpha, and
 - desired correlation dimensions

makeCorrAlpha() will inform the user if the resulting correlation matrix is positive definite, or not.

If the returned correlation matrix is not positive-definite, a feasible solution may still be possible. The user is encouraged to try again, possibly several times, to find one.

Examples

```
# define parameters
items <- 4
alpha <- 0.85
variance <- 0.5

# apply function
set.seed(42)
cor_matrix <- makeCorrAlpha(
  items = items,
  alpha = alpha,
  variance = variance
)

# test function output
print(cor_matrix)
alpha(cor_matrix)
eigenvalues(cor_matrix, 1)

# higher alpha, more items
cor_matrix2 <- makeCorrAlpha(
  items = 8,
```

```
    alpha = 0.95
  )

# test output
cor_matrix2 |> round(2)
alpha(cor_matrix2) |> round(3)
eigenvalues(cor_matrix2, 1) |> round(3)

# large random variation around alpha
set.seed(42)
cor_matrix3 <- makeCorrAlpha(
  items = 6,
  alpha = 0.85,
  precision = 2
)

# test output
cor_matrix3 |> round(2)
alpha(cor_matrix3) |> round(3)
eigenvalues(cor_matrix3, 1) |> round(3)

# with diagnostics
cor_matrix4 <- makeCorrAlpha(
  items = 4,
  alpha = 0.80,
  diagnostics = TRUE
)

# test output
cor_matrix4
```

makeCorrLoadings

Generate Inter-Item Correlation Matrix from Factor Loadings

Description

Constructs an inter-item correlation matrix based on a user-supplied matrix of standardised factor loadings and (optionally) a factor correlation matrix. The `makeCorrLoadings()` function does a surprisingly good job of reproducing a target correlation matrix when all item-factor loadings are present, as shown in the *makeCorrLoadings() validation article*.

Usage

```
makeCorrLoadings(
  loadings,
  factorCor = NULL,
  uniquenesses = NULL,
```

```

    nearPD = FALSE,
    diagnostics = FALSE
  )

```

Arguments

loadings	Numeric matrix. A $k \times f$ matrix of standardized factor loadings $items \times factors$. Row names and column names are used for diagnostics if present.
factorCor	Optional $f \times f$ matrix of factor correlations (Φ). If NULL, assumes orthogonal factors.
uniquenesses	Optional vector of length k . If NULL, calculated as $1 - rowSums(loadings^2)$.
nearPD	Logical. If TRUE, attempts to coerce non-positive-definite matrices using <code>Matrix::nearPD()</code> .
diagnostics	Logical. If TRUE, returns diagnostics including McDonald's Omega and item-level summaries.

Value

If `diagnostics = FALSE`, returns a correlation matrix (class: matrix). If `diagnostics = TRUE`, returns a list with: - R: correlation matrix - Omega: per-factor Omega or adjusted Omega - OmegaTotal: total Omega across all factors - Diagnostics: dataframe of communalities, uniquenesses, and primary factor

See Also

- `makeCorrLoadings()` validation article: https://winzarh.github.io/LikertMakeR/articles/makeCorrLoadings_validate.html
- Package website: <https://winzarh.github.io/LikertMakeR/>

Examples

```

# -----
# Example 1: Basic use without diagnostics
# -----

factorLoadings <- matrix(
  c(
    0.05, 0.20, 0.70,
    0.10, 0.05, 0.80,
    0.05, 0.15, 0.85,
    0.20, 0.85, 0.15,
    0.05, 0.85, 0.10,
    0.10, 0.90, 0.05,
    0.90, 0.15, 0.05,
    0.80, 0.10, 0.10
  ),
  nrow = 8, ncol = 3, byrow = TRUE
)

rownames(factorLoadings) <- paste0("Q", 1:8)
colnames(factorLoadings) <- c("Factor1", "Factor2", "Factor3")

```

```

factorCor <- matrix(
  c(
    1.0, 0.7, 0.6,
    0.7, 1.0, 0.4,
    0.6, 0.4, 1.0
  ),
  nrow = 3, byrow = TRUE
)

itemCor <- makeCorrLoadings(factorLoadings, factorCor)
round(itemCor, 3)

# -----
# Example 2: Diagnostics with factor correlations (Adjusted Omega)
# -----

result_adj <- makeCorrLoadings(
  loadings = factorLoadings,
  factorCor = factorCor,
  diagnostics = TRUE
)

# View outputs
round(result_adj$R, 3) # correlation matrix
round(result_adj$Omega, 3) # adjusted Omega
round(result_adj$OmegaTotal, 3) # total Omega
print(result_adj$Diagnostics) # communality and uniqueness per item

# -----
# Example 3: Diagnostics assuming orthogonal factors (Per-Factor Omega)
# -----

result_orth <- makeCorrLoadings(
  loadings = factorLoadings,
  diagnostics = TRUE
)

round(result_orth$Omega, 3) # per-factor Omega
round(result_orth$OmegaTotal, 3) # total Omega
print(result_orth$Diagnostics)

```

makeItems

*Synthesise rating-scale item data with given first and second moments
and a predefined correlation matrix*

Description

makeItems() generates a dataframe of random discrete values so the data replicate a rating scale, and are correlated close to a predefined correlation matrix.

makeItems() is being deprecated. Use the makeScales() function instead.

makeItems() is wrapper function for:

- `lfast()`, generates a dataframe that best fits the desired moments, and
- `lcor()`, which rearranges values in each column of the dataframe so they closely match the desired correlation matrix.

Usage

```
makeItems(n, means, sds, lowerbound, upperbound, cormatrix)
```

Arguments

<code>n</code>	(positive, int) sample-size - number of observations
<code>means</code>	(real) target means: a vector of length <code>k</code> of mean values for each scale item
<code>sds</code>	(positive, real) target standard deviations: a vector of length <code>k</code> of standard deviation values for each scale item
<code>lowerbound</code>	(positive, int) a vector of length <code>k</code> (same as rows & columns of correlation matrix) of values for lower bound of each scale item (e.g. '1' for a 1-5 rating scale)
<code>upperbound</code>	(positive, int) a vector of length <code>k</code> (same as rows & columns of correlation matrix) of values for upper bound of each scale item (e.g. '5' for a 1-5 rating scale)
<code>cormatrix</code>	(real, matrix) the target correlation matrix: a square symmetric positive-semi-definite matrix of values ranging between -1 and +1, and '1' in the diagonal.

Value

a dataframe of rating-scale values

Examples

```
## define parameters

n <- 16
dfMeans <- c(2.5, 3.0, 3.0, 3.5)
dfSds <- c(1.0, 1.0, 1.5, 0.75)
lowerbound <- rep(1, 4)
upperbound <- rep(5, 4)

corMat <- matrix(
  c(
    1.00, 0.30, 0.40, 0.60,
    0.30, 1.00, 0.50, 0.70,
    0.40, 0.50, 1.00, 0.80,
    0.60, 0.70, 0.80, 1.00
  ),
  nrow = 4, ncol = 4
)

item_names <- c("Q1", "Q2", "Q3", "Q4")
```

```

rownames(corMat) <- item_names
colnames(corMat) <- item_names

## apply function

df <- makeItems(
  n = n, means = dfMeans, sds = dfSds,
  lowerbound = lowerbound, upperbound = upperbound, cormatrix = corMat
)

## test function

str(df)

# means
apply(df, 2, mean) |> round(3)

# standard deviations
apply(df, 2, sd) |> round(3)

# correlations
cor(df) |> round(3)

```

makeItemsScale	<i>Generate scale items from a summated scale, with desired Cronbach's Alpha</i>
----------------	--

Description

makeItemsScale() generates a random dataframe of scale items based on a predefined summated scale (such as created by the `lfast()` function), and a desired *Cronbach's Alpha*.

scale, lowerbound, upperbound, items, alpha, variance

Usage

```

makeItemsScale(
  scale,
  lowerbound,
  upperbound,
  items,
  alpha = 0.8,
  variance = 0.5
)

```

Arguments

scale (int) a vector or dataframe of the summated rating scale. Should range from $lowerbound \times items$ to $upperbound \times items$

lowerbound	(int) lower bound of the scale item (example: '1' in a '1' to '5' rating)
upperbound	(int) upper bound of the scale item (example: '5' in a '1' to '5' rating)
items	(positive, int) k, or number of columns to generate
alpha	(positive, real) desired <i>Cronbach's Alpha</i> for the new dataframe of items. Default = '0.8'. See @details for further information on the alpha parameter
variance	(positive, real) the quantile from which to select items that give given summated scores. Must lie between '0' and '1'. Default = '0.5'. See @details for further information on the variance parameter

Details

alpha:

makeItemsScale() takes each value of a vector of Likert scales and produces a row of 'k' values that average the given scale value, and then rearranges the item values within each row, attempting to give a dataframe of Likert-scale items that produce a predefined *Cronbach's Alpha*.

Default value for target alpha is '0.8'.

More extreme values for the 'variance' parameter may reduce the chances of achieving the desired Alpha. So you may need to experiment a little.

variance:

There may be many ways to find a combination of integers that sum to a specific value, and these combinations have different levels of variance:

- low-variance: '3 + 4 = 7'
- high-variance: '1 + 6 = 7'

The 'variance' parameter defines guidelines for the amount of variance among item values that your new dataframe should have.

For example, consider a summated value of '9' on which we apply the makeItemsScale() function to generate three items. With zero variance (variance parameter = '0'), then we see all items with the same value, the mean of '3'. With variance = '1', then we see all items with values that give the maximum variance among those items.

variance	v1	v2	v3	sum
0.0	3	3	3	9
0.2	3	3	3	9
0.4	2	3	4	9
0.6	1	4	4	9
0.8	2	2	5	9
1.0	1	3	5	9

Similarly, the same mean value applied to six items with makeItemsScale() gives the following combinations at different values of the 'variance' parameter.

variance	v1	v2	v3	v4	v5	v6	sum
----------	----	----	----	----	----	----	-----

0.0	3	3	3	3	3	3	18
0.2	1	3	3	3	4	4	18
0.4	1	2	3	4	4	4	18
0.6	1	1	4	4	4	4	18
0.8	1	1	3	4	4	5	18
1.0	1	1	1	5	5	5	18

And a mean value of '3.5' gives the following combinations.

variance	v1	v2	v3	v4	v5	v6	sum
0.0	3	3	3	4	4	4	21
0.2	3	3	3	3	4	5	21
0.4	2	2	4	4	4	5	21
0.6	1	3	4	4	4	5	21
0.8	1	2	4	4	5	5	21
1.0	1	1	4	5	5	5	21

The default value for 'variance' is '0.5' which gives a reasonable range of item values. But if you want 'responses' that are more consistent then choose a lower variance value.

Value

a dataframe with 'items' columns and 'length(scale)' rows

Examples

```
## define parameters
k <- 4
lower <- 1
upper <- 5

## scale properties
n <- 64
mean <- 3.0
sd <- 0.85

## create scale
set.seed(42)
meanScale <- lfast(
  n = n, mean = mean, sd = sd,
  lowerbound = lower, upperbound = upper,
  items = k
)
summatedScale <- meanScale * k

## create new items
newItems <- makeItemsScale(
  scale = summatedScale,
  lowerbound = lower, upperbound = upper,
```

```
    items = k
  )

### test new items
# str(newItems)
# alpha(data = newItems) |> round(2)

## very low variance usually gives higher Cronbach's Alpha
mydat_20 <- makeItemsScale(
  scale = summatedScale,
  lowerbound = lower, upperbound = upper,
  items = k, alpha = 0.8, variance = 0.20
)

### test new data frame
# str(mydat_20)

# moments <- data.frame(
#   means = apply(mydat_20, MARGIN = 2, FUN = mean) |> round(3),
#   sds = apply(mydat_20, MARGIN = 2, FUN = sd) |> round(3)
# ) |> t()

# moments

# cor(mydat_20) |> round(2)
# alpha(data = mydat_20) |> round(2)

## default alpha (0.8) and higher variance (0.8)
mydat_80 <- makeItemsScale(
  scale = summatedScale,
  lowerbound = lower, upperbound = upper,
  items = k, variance = 0.80
)

### test new dataframe
# str(mydat_80)

# moments <- data.frame(
#   means = apply(mydat_80, MARGIN = 2, FUN = mean) |> round(3),
#   sds = apply(mydat_80, MARGIN = 2, FUN = sd) |> round(3)
# ) |> t()

# moments

# cor(mydat_80) |> round(2)
# alpha(data = mydat_80) |> round(2)
```

Description

The function `makePaired()` generates a dataset from paired-sample t -test summary statistics.

`makePaired()` generates correlated values so the data replicate rating scales taken, for example, in a before and after experimental design.

The function is effectively a wrapper function for `lfast()` and `lcor()` with the addition of a t -statistic from which the between-column correlation is inferred.

Paired t -tests apply to observations that are associated with each other. For example: the same people before and after a treatment; the same people rating two different objects; ratings by husband & wife; *etc.*

The paired-samples t -test is defined as:

$$t = \frac{\text{mean}(D)}{\text{sd}(D)/\sqrt{n}}$$

where:

- D = differences in values
- $\text{mean}(D)$ = mean of the differences
- $\text{sd}(D)$ = standard deviation of the differences, where

$$\text{sd}(D)^2 = \text{sd}(X_{\text{before}})^2 + \text{sd}(X_{\text{after}})^2 - 2 \text{cov}(X_{\text{before}}, X_{\text{after}})$$

A paired-sample t -test thus requires an estimate of the covariance between the two sets of observations. `makePaired()` rearranges these formulae so that the covariance is inferred from the t -statistic.

Usage

```
makePaired(
  n,
  means,
  sds,
  t_value,
  lowerbound,
  upperbound,
  items = 1,
  precision = 0
)
```

Arguments

<code>n</code>	(positive, integer) sample size
<code>means</code>	(real) 1:2 vector of target means for two before/after measures
<code>sds</code>	(real) 1:2 vector of target standard deviations
<code>t_value</code>	(real) desired paired t -statistic
<code>lowerbound</code>	(integer) lower bound (e.g. '1' for a 1-5 rating scale)

upperbound (integer) upper bound (e.g. '5' for a 1-5 rating scale)
 items (positive, integer) number of items in the rating scale. Default = 1
 precision (positive, real) relaxes the level of accuracy required. Default = 0

Value

a dataframe approximating user-specified conditions.

Note

Larger sample sizes usually result in higher *t*-statistics, and correspondingly small *p*-values.

Small sample sizes with relatively large standard deviations and relatively high *t*-statistics can result in impossible correlation values.

Similarly, large sample sizes with low *t*-statistics can result in impossible correlations. That is, a correlation outside of the -1:+1 range.

If this happens, the function will fail with an *ERROR* message. The user should review the input parameters and insert more realistic values.

Examples

```
n <- 20
pair_m <- c(2.5, 3.0)
pair_s <- c(1.0, 1.5)
lower <- 1
upper <- 5
k <- 6
t <- -2.5

pairedDat <- makePaired(
  n = n, means = pair_m, sds = pair_s,
  t_value = t,
  lowerbound = lower, upperbound = upper, items = k
)

str(pairedDat)
cor(pairedDat) |> round(2)

t.test(pairedDat$X1, pairedDat$X2, paired = TRUE)
```

 makeRepeated

Reproduce Repeated-Measures Data from ANOVA Summary Statistics

Description

Constructs a synthetic dataset and inter-timepoint correlation matrix from a repeated-measures ANOVA result, based on reported means, standard deviations, and an *F*-statistic. This is useful when only summary statistics are available from published studies.

Usage

```

makeRepeated(
  n,
  k,
  means,
  sds,
  f_stat,
  df_between = k - 1,
  df_within = (n - 1) * (k - 1),
  structure = c("cs", "ar1", "toeplitz"),
  names = paste0("time_", 1:k),
  items = 1,
  lowerbound = 1,
  upperbound = 5,
  return_corr_only = FALSE,
  diagnostics = FALSE,
  ...
)

```

Arguments

n	Integer. Sample size used in the original study.
k	Integer. Number of repeated measures (timepoints).
means	Numeric vector of length k. Mean values reported for each timepoint.
sds	Numeric vector of length k. Standard deviations reported for each timepoint.
f_stat	Numeric. The reported repeated-measures ANOVA F-statistic for the within-subjects factor.
df_between	Degrees of freedom between conditions (default: k - 1).
df_within	Degrees of freedom within-subjects (default: (n - 1) * (k - 1)).
structure	Character. Correlation structure to assume: "cs", "ar1", or "toeplitz" (default = "cs").
names	Character vector of length k. Variable names for each timepoint (default: "time_1" to "time_k").
items	Integer. Number of items used to generate each scale score (passed to lfast).
lowerbound	Integer. Lower bounds for Likert-type response scales (default: 1).
upperbound	Integer. upper bounds for Likert-type response scales (default: 5).
return_corr_only	Logical. If TRUE, return only the estimated correlation matrix.
diagnostics	Logical. If TRUE, include diagnostic summaries such as feasible F-statistic range and effect sizes.
...	Reserved for future use.

Details

This function estimates the average correlation between repeated measures by matching the reported F-statistic, under one of three assumed correlation structures:

- "cs" (*Compound Symmetry*): The Default. Assumes all timepoints are equally correlated. Common in standard RM-ANOVA settings.
- "ar1" (*First-Order Autoregressive*): Assumes correlations decay exponentially with time lag.
- "toeplitz" (*Linearly Decreasing*): Assumes correlation declines linearly with time lag - a middle ground between "cs" and "ar1".

The function then generates a data frame of synthetic item-scale ratings using [lfast](#), and adjusts them to match the estimated correlation structure using [lcor](#).

Set `return_corr_only = TRUE` to extract only the estimated correlation matrix.

Value

A named list with components:

`data` A data frame of simulated repeated-measures responses (unless `return_corr_only = TRUE`).

`correlation_matrix` The estimated inter-timepoint correlation matrix.

`structure` The correlation structure assumed.

`achieved_f` The F-statistic produced by the estimated rho value (if `diagnostics = TRUE`).

`feasible_f_range` Minimum and maximum achievable F-values under the chosen structure (shown if diagnostics are requested).

`recommended_f` Conservative, moderate, and strong F-statistic suggestions for similar designs.

`effect_size_raw` Unstandardised effect size across timepoints.

`effect_size_standardised` Effect size standardised by average variance.

See Also

[lfast](#), [lcor](#)

Examples

```
set.seed(42)

out1 <- makeRepeated(
  n = 64,
  k = 3,
  means = c(3.1, 3.5, 3.9),
  sds = c(1.0, 1.1, 1.0),
  items = 4,
  f_stat = 4.87,
  structure = "cs",
  diagnostics = FALSE
)

head(out1$data)
```

```
out1$correlation_matrix

out2 <- makeRepeated(
  n = 32, k = 4,
  means = c(2.75, 3.5, 4.0, 4.4),
  sds = c(0.8, 1.0, 1.2, 1.0),
  f_stat = 16,
  structure = "ar1",
  items = 5,
  lowerbound = 1, upperbound = 7,
  return_corr_only = FALSE,
  diagnostics = TRUE
)

print(out2)

out3 <- makeRepeated(
  n = 64, k = 4,
  means = c(2.0, 2.25, 2.75, 3.0),
  sds = c(0.8, 0.9, 1.0, 0.9),
  items = 4,
  f_stat = 24,
  # structure = "toeplitz",
  diagnostics = TRUE
)

str(out3)
```

makeScales

Synthesise rating-scale data with given first and second moments and a predefined correlation matrix

Description

makeScales() generates a dataframe of random discrete values so the data replicate a rating scale, and are correlated close to a predefined correlation matrix.

makeScales() is wrapper function for:

- [lfast\(\)](#), generates a dataframe that best fits the desired moments, and
- [lcor\(\)](#), which rearranges values in each column of the dataframe so they closely match the desired correlation matrix.

Usage

```
makeScales(n, means, sds, lowerbound = 1, upperbound = 5, items = 1, cormatrix)
```


Arguments

n	(positive, int) sample-size - number of observations
means	(real) target means: a vector of length k of mean values for each scale item
sds	(positive, real) target standard deviations: a vector of length k of standard deviation values for each scale item
lowerbound	(positive, int) a vector of length k (same as rows & columns of correlation matrix) of values for lower bound of each scale item (e.g. '1' for a 1-5 rating scale). Default = 1.
upperbound	(positive, int) a vector of length k (same as rows & columns of correlation matrix) of values for upper bound of each scale item (e.g. '5' for a 1-5 rating scale). Default = 5.
items	(positive, int) a vector of length k of number of items in each scale. Default = 1.
cormatrix	(real, matrix) the target correlation matrix: a square symmetric positive-semi-definite matrix of values ranging between -1 and +1, and '1' in the diagonal.

Value

a dataframe of rating-scale values

Examples

```
## Example 1: four correlated items (questions)

### define parameters

n <- 16
dfMeans <- c(2.5, 3.0, 3.0, 3.5)
dfSds <- c(1.0, 1.0, 1.5, 0.75)
lowerbound <- rep(1, 4)
upperbound <- rep(5, 4)

corMat <- matrix(
  c(
    1.00, 0.30, 0.40, 0.60,
    0.30, 1.00, 0.50, 0.70,
    0.40, 0.50, 1.00, 0.80,
    0.60, 0.70, 0.80, 1.00
  ),
  nrow = 4, ncol = 4
)

scale_names <- c("Q1", "Q2", "Q3", "Q4")
rownames(corMat) <- scale_names
colnames(corMat) <- scale_names

### apply function

df1 <- makeScales(
  n = n, means = dfMeans, sds = dfSds,
```

```

    lowerbound = lowerbound, upperbound = upperbound, cormatrix = corMat
  )

### test function

str(df1)

#### means
apply(df1, 2, mean) |> round(3)

#### standard deviations
apply(df1, 2, sd) |> round(3)

#### correlations
cor(df1) |> round(3)

## Example 2: five correlated Likert scales

### a study on employee engagement and organizational climate:
# Job Satisfaction (JS)
# Organizational Commitment (OC)
# Perceived Supervisor Support (PSS)
# Work Engagement (WE)
# Turnover Intention (TI) (reverse-related to others).

### define parameters

n <- 128
dfMeans <- c(3.8, 3.6, 3.7, 3.9, 2.2)
dfSds <- c(0.7, 0.8, 0.7, 0.6, 0.9)
lowerbound <- rep(1, 5)
upperbound <- rep(5, 5)
items <- c(4, 4, 3, 3, 3)

corMat <- matrix(
  c(
    1.00, 0.72, 0.58, 0.65, -0.55,
    0.72, 1.00, 0.54, 0.60, -0.60,
    0.58, 0.54, 1.00, 0.57, -0.45,
    0.65, 0.60, 0.57, 1.00, -0.50,
    -0.55, -0.60, -0.45, -0.50, 1.00
  ),
  nrow = 5, ncol = 5
)

scale_names <- c("JS", "OC", "PSS", "WE", "TI")
rownames(corMat) <- scale_names
colnames(corMat) <- scale_names

### apply function

```

```
df2 <- makeScales(  
  n = n, means = dfMeans, sds = dfSds,  
  lowerbound = lowerbound, upperbound = upperbound,  
  items = items, cormatrix = corMat  
)  
  
### test function  
  
str(df2)  
  
#### means  
apply(df2, 2, mean) |> round(3)  
  
#### standard deviations  
apply(df2, 2, sd) |> round(3)  
  
#### correlations  
cor(df2) |> round(3)
```

makeScalesRegression *Generate Data from Multiple-Regression Summary Statistics*

Description

Generates synthetic rating-scale data that replicates reported regression results. This function is useful for reproducing analyses from published research where only summary statistics (standardised regression coefficients and R-squared) are reported.

Usage

```
makeScalesRegression(  
  n,  
  beta_std,  
  r_squared,  
  iv_cormatrix = NULL,  
  iv_cor_mean = 0.3,  
  iv_cor_variance = 0.01,  
  iv_cor_range = c(-0.7, 0.7),  
  iv_means,  
  iv_sds,  
  dv_mean,  
  dv_sd,  
  lowerbound_iv,  
  upperbound_iv,  
  lowerbound_dv,  
  upperbound_dv,  
  items_iv = 1,
```

```

    items_dv = 1,
    var_names = NULL,
    tolerance = 0.005
)

```

Arguments

<code>n</code>	Integer. Sample size
<code>beta_std</code>	Numeric vector of standardised regression coefficients (length <code>k</code>)
<code>r_squared</code>	Numeric. R-squared from regression (-1 to 1)
<code>iv_cormatrix</code>	<code>k</code> x <code>k</code> correlation matrix of independent variables. If missing (NULL), will be optimised.
<code>iv_cor_mean</code>	Numeric. Mean correlation among IVs when optimising (ignored if <code>iv_cormatrix</code> provided). Default = 0.3
<code>iv_cor_variance</code>	Numeric. Variance of correlations when optimising (ignored if <code>iv_cormatrix</code> provided). Default = 0.01
<code>iv_cor_range</code>	Numeric vector of length 2. Min and max constraints on correlations when optimising. Default = <code>c(-0.7, 0.7)</code>
<code>iv_means</code>	Numeric vector of means for IVs (length <code>k</code>)
<code>iv_sds</code>	Numeric vector of standard deviations for IVs (length <code>k</code>)
<code>dv_mean</code>	Numeric. Mean of dependent variable
<code>dv_sd</code>	Numeric. Standard deviation of dependent variable
<code>lowerbound_iv</code>	Numeric vector of lower bounds for each IV scale (or single value for all)
<code>upperbound_iv</code>	Numeric vector of upper bounds for each IV scale (or single value for all)
<code>lowerbound_dv</code>	Numeric. Lower bound for DV scale
<code>upperbound_dv</code>	Numeric. Upper bound for DV scale
<code>items_iv</code>	Integer vector of number of items per IV scale (or single value for all). Default = 1
<code>items_dv</code>	Integer. Number of items in DV scale. Default = 1
<code>var_names</code>	Character vector of variable names (length <code>k+1</code> : IVs then DV)
<code>tolerance</code>	Numeric. Acceptable deviation from target R-squared (default 0.005)

Details

Generate regression data from summary statistics

The function can operate in two modes:

Mode 1: With IV correlation matrix provided

When `iv_cormatrix` is provided, the function uses the given correlation structure among independent variables and calculates the implied IV-DV correlations from the regression coefficients.

Mode 2: With optimisation (IV correlation matrix not provided)

When `iv_cormatrix = NULL`, the function optimises to find a plausible correlation structure among independent variables that matches the reported regression statistics. Initial correlations are sampled using Fisher's z-transformation to ensure proper distribution, then iteratively adjusted to match the target R-squared.

The function generates Likert-scale data (not individual items) using `lfast()` for each variable with specified moments, then correlates them using `lcor()`. Generated data are verified by running a regression and comparing achieved statistics with targets.

Value

A list containing:

<code>data</code>	Generated dataframe with k IVs and 1 DV
<code>target_stats</code>	List of target statistics provided
<code>achieved_stats</code>	List of achieved statistics from generated data
<code>diagnostics</code>	Comparison of target vs achieved
<code>iv_dv_cors</code>	Calculated correlations between IVs and DV
<code>full_cormatrix</code>	The complete (k+1) x (k+1) correlation matrix used
<code>optimisation_info</code>	If IV correlations were optimised, details about the optimisation

See Also

[lfast](#) for generating individual rating-scale vectors with exact moments.

[lcor](#) for rearranging values to achieve target correlations.

[makeCorrAlpha](#) for generating correlation matrices from Cronbach's Alpha.

Examples

```
# Example 1: With provided IV correlation matrix
set.seed(123)
iv_corr <- matrix(c(1.0, 0.3, 0.3, 1.0), nrow = 2)

result1 <- makeScalesRegression(
  n = 64,
  beta_std = c(0.4, 0.3),
  r_squared = 0.35,
  iv_cormatrix = iv_corr,
  iv_means = c(3.0, 3.5),
  iv_sds = c(1.0, 0.9),
  dv_mean = 3.8,
  dv_sd = 1.1,
  lowerbound_iv = 1,
  upperbound_iv = 5,
  lowerbound_dv = 1,
  upperbound_dv = 5,
  items_iv = 4,
  items_dv = 4,
```

```
    var_names = c("Attitude", "Intention", "Behaviour")
  )

print(result1)
head(result1$data)

# Example 2: With optimisation (no IV correlation matrix)
set.seed(456)
result2 <- makeScalesRegression(
  n = 128,
  beta_std = c(0.3, 0.25, 0.2),
  r_squared = 0.40,
  iv_cormatrix = NULL, # Will be optimised
  iv_cor_mean = 0.3,
  iv_cor_variance = 0.02,
  iv_means = c(3.0, 3.2, 2.8),
  iv_sds = c(1.0, 0.9, 1.1),
  dv_mean = 3.5,
  dv_sd = 1.0,
  lowerbound_iv = 1,
  upperbound_iv = 5,
  lowerbound_dv = 1,
  upperbound_dv = 5,
  items_iv = 4,
  items_dv = 5
)

# View optimised correlation matrix
print(result2$target_stats$iv_cormatrix)
print(result2$optimisation_info)
```

print.makeScalesRegression

Print method for makeScalesRegression objects

Description

Print method for makeScalesRegression objects

Usage

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

Arguments

x	An object of class "makeScalesRegression"
...	Additional arguments (currently unused)

Index

alpha, [2](#)

correlateScales, [3](#)

eigenvalues, [5](#)

lcor, [6](#), [23](#), [29](#)
lcor(), [15](#), [24](#)
lexact, [7](#)
lfast, [8](#), [22](#), [23](#), [29](#)
lfast(), [15](#), [16](#), [24](#)

makeCorrAlpha, [10](#), [29](#)
makeCorrLoadings, [12](#)
makeItems, [14](#)
makeItems(), [10](#)
makeItemsScale, [16](#)
makePaired, [19](#)
makeRepeated, [21](#)
makeScales, [24](#)
makeScalesRegression, [27](#)
Matrix::nearPD(), [13](#)

print.makeScalesRegression, [30](#)