

# Package: LikertMakeR (via r-universe)

October 17, 2024

**Type** Package

**Title** Synthesise and Correlate Rating-Scale Data

**Version** 0.3.0

**Description** Synthesise rating-scale data with predefined first & second moments (mean & standard deviation) and, optionally, correlate multiple vectors with predefined correlation matrix. Also generate synthetic rating-scale data with predefined Cronbach's Alpha, or generate rating-scale items from a predefined scale.

**License** MIT + file LICENSE

**URL** <https://github.com/WinzarH/LikertMakeR>

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

**Depends** R (>= 4.2.0)

**Imports** dplyr, gtools, Rcpp

**Suggests** knitr, rmarkdown, testthat (>= 3.0.0)

**LinkingTo** Rcpp, RcppArmadillo

**VignetteBuilder** knitr

**Config/testthat/edition** 3

**Encoding** UTF-8

**Language** en-GB

**RoxygenNote** 7.3.1

**NeedsCompilation** yes

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

**Maintainer** Hume Winzar <winzar@gmail.com>

**Repository** CRAN

**Date/Publication** 2024-05-19 13:00:06 UTC

## Contents

alpha . . . . .	2
correlateScales . . . . .	3
eigenvalues . . . . .	5
lcor . . . . .	6
lexact . . . . .	7
lfast . . . . .	8
makeCorrAlpha . . . . .	10
makeItems . . . . .	11
makeItemsScale . . . . .	13

<b>Index</b>	<b>16</b>
--------------	-----------

---

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

---

### Description

alpha() calculate 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>Create a 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*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()` calculate 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 columns in a data-frame to fit a predefined correlation matrix</i>
------	---

---

**Description**

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

**Usage**

```
lcor(data, target)
```

**Arguments**

data	data-frame that is to be rearranged
target	target correlation matrix - should be a symmetric k*k positive-semi-definite matrix

**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 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 (e.g. '1' for a 1-5 rating scale)
upperbound	(positive, int) upper bound (e.g. '5' for a 1-5 rating scale)
items	(positive, int) number of items in the rating scale. Default = 1

**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 a rating scale - for example, a 1-5 Likert scale made from 5 items (questions) or 0-10 likelihood-of-purchase scale.

**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	(positive, int) lower bound (e.g. '1' for a 1-5 rating scale)
upperbound	(positive, 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.

**Examples**

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

## 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	<i>Correlation matrix from Cronbach's Alpha</i>
---------------	---

---

**Description**

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

**Usage**

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

**Arguments**

items	(positive, int) matrix dimensions: number of rows & columns to generate
alpha	(real) target Cronbach's Alpha (usually positive, must be between -1 and +1)
variance	(positive, real) Default = 0.5. User-provided standard deviation of values sampled from a normally-distributed log transformation.
precision	(positive, real) Default = 0. User-defined value ranging from '0' to '3' to add some random variation around the target Cronbach's Alpha. '0' gives an exact alpha (to two decimal places)

**Value**

a correlation matrix

**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)
```

---

makeItems

*Synthetic rating-scale 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 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**

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)
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)
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**

```
## 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
)

## 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      scale items from a summated scale
```

---

### Description

makeItemsScale() generates a random dataframe of scale items based on a predefined summated scale, such as created by the lfast() function.

scale, lowerbound, upperbound, items

### Usage

```
makeItemsScale(scale, lowerbound, upperbound, items, variance = 0.5)
```

### Arguments

scale	(int) a vector or dataframe of the summated rating scale. Should range from ('lowerbound' * 'items') to ('upperbound' * '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
variance	(positive, real) standard deviation of values sampled from a normally-distributed log transformation. Default = '0.5'. A value of '0' makes all values in the correlation matrix the same, equal to the mean correlation needed to produce the desired <i>Cronbach's Alpha</i> . A value of '2', or more, risks producing a matrix that is not positive-definite, so not feasible.

### Value

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

### Examples

```
## define parameters
items <- 4
lowerbound <- 1
upperbound <- 5

## scale properties
n <- 64
```

```
mean <- 3.5
sd <- 1.00

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

## create items
newItems <- makeItemsScale(
  scale = summatedScale,
  lowerbound = lowerbound, upperbound = upperbound,
  items = items
)
str(newItems)

##
## Testing Lowest value to Highest value of a scale
##
lowerbound <- 1
upperbound <- 5
items <- 6

# lowest to highest values
myvalues <- c((lowerbound * items):(upperbound * items))

## Low variance usually gives higher Cronbach's Alpha
mydat_20 <- makeItemsScale(
  scale = myvalues,
  lowerbound = lowerbound, upperbound = upperbound,
  items = items, variance = 0.20
)

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(mydat_20) |> round(2)

## default variance
mydat_50 <- makeItemsScale(
  scale = myvalues,
```

```
    lowerbound = lowerbound, upperbound = upperbound,
    items = items, variance = 0.50
)

str(mydat_50)

moments <- data.frame(
  means = apply(mydat_50, MARGIN = 2, FUN = mean) |> round(3),
  sds = apply(mydat_50, MARGIN = 2, FUN = sd) |> round(3)
) |> t()

moments

cor(mydat_50) |> round(2)
alpha(mydat_50) |> round(2)

## higher variance usually gives lower Cronbach's Alpha
mydat_80 <- makeItemsScale(
  scale = myvalues,
  lowerbound = lowerbound, upperbound = upperbound,
  items = items, variance = 0.80
)

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(mydat_80) |> round(2)
```

# Index

alpha, [2](#)

correlateScales, [3](#)

eigenvalues, [5](#)

lcor, [6](#)

lexact, [7](#)

lfast, [8](#)

makeCorrAlpha, [10](#)

makeItems, [11](#)

makeItemsScale, [13](#)