Package ‘petersenlab’

February 3, 2025

Type Package

Title A Collection of R Functions by the Petersen Lab

Version 1.1.0

Maintainer Isaac T. Petersen <isaac-t-petersen@uiowa.edu>

Depends R (>= 4.1.0)

Imports stats, graphics, utils, nlme, Hmisc, digest, dplyr, ggplot2, lavaan, mitools, mix, mvtnorm, psych, stringr, xtable, grDevices, plyr, reshape2, RColorBrewer, viridisLite, tidyselect, scales, purrr

Suggests testthat (>= 3.0.0), waldo, withr

Description A collection of R functions that are widely used by the Petersen Lab. Included are functions for various purposes, including evaluating the accuracy of judgments and predictions, performing scoring of assessments, generating correlation matrices, conversion of data between various types, data management, psychometric evaluation, extensions related to latent variable modeling, various plotting capabilities, and other miscellaneous useful functions. By making the package available, we hope to make our methods reproducible and replicable by others and to help others perform their data processing and analysis methods more easily and efficiently. The codebase is provided in Petersen (2024) <doi:10.5281/zenodo.7602890> and on CRAN: <doi:10.32614/CRAN.package.petersenlab>. The package is described in ``Principles of Psychological Assessment: With Applied Examples in R'' (Petersen, 2024, 2025) <doi:10.1201/9781003357421>, <doi:10.25820/work.007199>, <doi:10.5281/zenodo.6466589>.

URL `https://github.com/DevPsyLab/petersenlab`, `https://devpsylab.github.io/petersenlab/`

BugReports `https://github.com/DevPsyLab/petersenlab/issues`

License MIT + file LICENSE

Encoding UTF-8

RoxygenNote 7.3.2

Config/testthat/edition 3

NeedsCompilation no

Author Isaac T. Petersen [aut, cre] (<https://orcid.org/0000-0003-3072-6673>),
Developmental Psychopathology Lab at the University of Iowa [ctb],
Angela D. Staples [ctb] (<https://orcid.org/0000-0002-7678-5794>),
Johanna Caskey [ctb] (<https://orcid.org/0009-0001-8227-6603>),
Philipp Doebler [ctb] (<https://orcid.org/0000-0002-2946-8526>),
Loreen Sabel [ctb] (<https://orcid.org/0000-0002-9832-8842>)

Repository CRAN

Date/Publication 2025-02-03 08:00:02 UTC

Contents

accuracyAtCutoff 4
accuracyAtEachCutoff 6
accuracyOverall 9
addText 10
apa 11
attenuationCorrelation 12
cleanUpNames 13
columnBindFill 14
complement 15
convert.magic 16
convertHoursAMPM 17
convertToHours 18
convertToMinutes 19
convertToSeconds 20
cor.table 21
crossTimeCorrelation 22
crossTimeCorrelationDF 23
deriv_d_negBinom 24
disattenuationCorrelation 27
discriminationToFactorLoading 28
dropColsWithAllNA 29
dropRowsWithAllNA 30
equiv_chi 31
fourPL 32
getDependencies 33
imputationCombine 34
imputationModelCompare 35
imputationPRV 35
is.nan.data.frame 36
itemInformation 37
kish_ess 38
lmCombine 40
lmeSummary 40
load_or_install 41
make_esem_model 42

meanSum 44
Mode 45
mortgage 46
mySum 46
my_loadings_sorter 47
nomogrammer 48
not_all_na 50
not_any_na 51
optimalCutoff 52
pA 54
partialcor.table 55
percentEffort 57
percentileToTScore 59
plot2WayInteraction 60
pom 63
posttestOdds 65
ppPlot 66
puc 67
pValue 68
read.aes 69
recode_intensity 71
redcapProgressBar 72
reliabilityIRT 73
reliabilityOfDifferenceScore 74
repeatability 75
reverse_score 76
satorraBentlerScaledChiSquareDifferenceTestStatistic 77
semPlotInteraction 78
setLabPath 81
simulateAUC 81
simulateIndirectEffect 82
specify_decimal 84
standardErrorIRT 84
suppressLeadingZero 85
test_info_4PL 86
timesPerInterval 88
update_nested 90
varsDifferentTypes 92
vwReg 93
write.aes 96
%ni% 97

Index **99**

accuracyAtCutoff *Accuracy at a Given Cutoff.*

Description

Find the accuracy at a given cutoff. Actuals should be binary, where `1` = present and `0` = absent.

Usage

```
accuracyAtCutoff(
  predicted,
  actual,
  cutoff,
  UH = NULL,
  UM = NULL,
  UCR = NULL,
  UFA = NULL
)
```

Arguments

`predicted`	vector of continuous predicted values.
`actual`	vector of binary actual values (`1` = present and `0` = absent).
`cutoff`	numeric value at or above which the target condition is considered present.
`UH`	(optional) utility of hits (true positives), specified as a value from 0-1, where 1 is the most highly valued and 0 is the least valued.
`UM`	(optional) utility of misses (false negatives), specified as a value from 0-1, where 1 is the most highly valued and 0 is the least valued.
`UCR`	(optional) utility of correct rejections (true negatives), specified as a value from 0-1, where 1 is the most highly valued and 0 is the least valued.
`UFA`	(optional) utility of false positives (false positives), specified as a value from 0-1, where 1 is the most highly valued and 0 is the least valued.

Details

Compute accuracy indices of predicted values in relation to actual values at a given cutoff by specifying the predicted values, actual values, and cutoff value. The target condition is considered present at or above the cutoff value. Optionally, you can also specify the utility of hits, misses, correct rejections, and false alarms to calculate the overall utility of the cutoff. To compute accuracy at each possible cutoff, see accuracyAtEachCutoff.

Value

- `cutoff` = the cutoff specified
- `TP` = true positives
- `TN` = true negatives
- `FP` = false positives
- `FN` = false negatives
- `SR` = selection ratio
- `BR` = base rate
- `percentAccuracy` = percent accuracy
- `percentAccuracyByChance` = percent accuracy by chance
- `percentAccuracyPredictingFromBaseRate` = percent accuracy from predicting from the base rate
- `RIOC` = relative improvement over chance
- `relativeImprovementOverPredictingFromBaseRate` = relative improvement over predicting from the base rate
- `SN` = sensitivty
- `SP` = specificity
- `TPrate` = true positive rate
- `TNrate` = true negative rate
- `FNrate` = false negative rate
- `FPrate` = false positive rate
- `HR` = hit rate
- `FAR` = false alarm rate
- `PPV` = positive predictive value
- `NPV` = negative predictive value
- `FDR` = false discovery rate
- `FOR` = false omission rate
- `youdenJ` = Youden's J statistic
- `balancedAccuracy` = balanced accuracy
- `f1Score` = F1-score
- `mcc` = Matthews correlation coefficient
- `diagnosticOddsRatio` = diagnostic odds ratio
- `positiveLikelihoodRatio` = positive likelihood ratio
- `negativeLikelhoodRatio` = negative likelihood ratio
- `dPrimeSDT` = d-Prime index from signal detection theory
- `betaSDT` = beta index from signal detection theory
- `cSDT` = c index from signal detection theory

- `aSDT` = a index from signal detection theory
- `bSDT` = b index from signal detection theory
- `differenceBetweenPredictedAndObserved` = difference between predicted and observed values
- `informationGain` = information gain
- `overallUtility` = overall utility (if utilities were specified)

See Also

Other accuracy: `accuracyAtEachCutoff()`, `accuracyOverall()`, `nomogrammer()`, `optimalCutoff()`, `posttestOdds()`

Examples

```
# Prepare Data
data("USArrests")
USArrests$highMurderState <- NA
USArrests$highMurderState[which(USArrests$Murder >= 10)] <- 1
USArrests$highMurderState[which(USArrests$Murder < 10)] <- 0

# Calculate Accuracy
accuracyAtCutoff(predicted = USArrests$Assault,
  actual = USArrests$highMurderState, cutoff = 200)
accuracyAtCutoff(predicted = USArrests$Assault,
  actual = USArrests$highMurderState, cutoff = 200,
  UH = 1, UM = 0, UCR = .9, UFA = 0)
```

`accuracyAtEachCutoff` *Accuracy at Each Cutoff.*

Description

Find the accuracy at each possible cutoff. Actuals should be binary, where `1` = present and `0` = absent.

Usage

```
accuracyAtEachCutoff(
  predicted,
  actual,
  UH = NULL,
  UM = NULL,
  UCR = NULL,
  UFA = NULL
)
```

Arguments

`predicted`	vector of continuous predicted values.
`actual`	vector of binary actual values (`1` = present and `0` = absent).
`UH`	(optional) utility of hits (true positives), specified as a value from 0-1, where 1 is the most highly valued and 0 is the least valued.
`UM`	(optional) utility of misses (false negatives), specified as a value from 0-1, where 1 is the most highly valued and 0 is the least valued.
`UCR`	(optional) utility of correct rejections (true negatives), specified as a value from 0-1, where 1 is the most highly valued and 0 is the least valued.
`UFA`	(optional) utility of false positives (false positives), specified as a value from 0-1, where 1 is the most highly valued and 0 is the least valued.

Details

Compute accuracy indices of predicted values in relation to actual values at each possible cutoff by specifying the predicted values and actual values. The target condition is considered present at or above each cutoff value. Optionally, you can specify the utility of hits, misses, correct rejections, and false alarms to calculate the overall utility of each possible cutoff.

Value

- `cutoff` = the cutoff specified
- `TP` = true positives
- `TN` = true negatives
- `FP` = false positives
- `FN` = false negatives
- `SR` = selection ratio
- `BR` = base rate
- `percentAccuracy` = percent accuracy
- `percentAccuracyByChance` = percent accuracy by chance
- `percentAccuracyPredictingFromBaseRate` = percent accuracy from predicting from the base rate
- `RIOC` = relative improvement over chance
- `relativeImprovementOverPredictingFromBaseRate` = relative improvement over predicting from the base rate
- `SN` = sensitivty
- `SP` = specificity
- `TPrate` = true positive rate
- `TNrate` = true negative rate
- `FNrate` = false negative rate
- `FPrate` = false positive rate

- `HR` = hit rate
- `FAR` = false alarm rate
- `PPV` = positive predictive value
- `NPV` = negative predictive value
- `FDR` = false discovery rate
- `FOR` = false omission rate
- `youdenJ` = Youden's J statistic
- `balancedAccuracy` = balanced accuracy
- `f1Score` = F1-score
- `mcc` = Matthews correlation coefficient
- `diagnosticOddsRatio` = diagnostic odds ratio
- `positiveLikelihoodRatio` = positive likelihood ratio
- `negativeLikelhoodRatio` = negative likelihood ratio
- `dPrimeSDT` = d-Prime index from signal detection theory
- `betaSDT` = beta index from signal detection theory
- `cSDT` = c index from signal detection theory
- `aSDT` = a index from signal detection theory
- `bSDT` = b index from signal detection theory
- `differenceBetweenPredictedAndObserved` = difference between predicted and observed values
- `informationGain` = information gain
- `overallUtility` = overall utility (if utilities were specified)

See Also

Other accuracy: `accuracyAtCutoff()`, `accuracyOverall()`, `nomogrammer()`, `optimalCutoff()`, `posttestOdds()`

Examples

```
# Prepare Data
data("USArrests")
USArrests$highMurderState <- NA
USArrests$highMurderState[which(USArrests$Murder >= 10)] <- 1
USArrests$highMurderState[which(USArrests$Murder < 10)] <- 0

# Calculate Accuracy
accuracyAtEachCutoff(predicted = USArrests$Assault,
  actual = USArrests$highMurderState)
accuracyAtEachCutoff(predicted = USArrests$Assault,
  actual = USArrests$highMurderState,
  UH = 1, UM = 0, UCR = .9, UFA = 0)
```

`accuracyOverall` *Overall Accuracy.*

Description

Find overall accuracy.

Usage

```
accuracyOverall(predicted, actual, dropUndefined = FALSE)

wisdomOfCrowd(predicted, actual, dropUndefined = FALSE)
```

Arguments

`predicted`	vector of continuous predicted values.
`actual`	vector of actual values.
`dropUndefined`	`TRUE` or `FALSE`, indicating whether to drop any undefined values calculated with the accuracy indices.

Details

Compute overall accuracy estimates of predicted values in relation to actual values. Estimates of overall accuracy span all cutoffs. Some accuracy estimates can be undefined under various circumstances. Optionally, you can drop undefined values in the calculation of accuracy indices. Note that dropping undefined values changes the meaning of these indices. Use this option at your own risk!

Value

- `ME` = mean error
- `MAE` = mean absolute error
- `MSE` = mean squared error
- `RMSE` = root mean squared error
- `MPE` = mean percentage error
- `MAPE` = mean absolute percentage error
- `sMAPE` = symmetric mean absolute percentage error
- `MASE` = mean absolute scaled error
- `RMSLE` = root mean squared log error
- `rsquared` = R-squared
- `rsquaredAdj` = adjusted R-squared
- `rsquaredPredictive` = predictive R-squared

See Also

Mean absolute scaled error (MASE):
https://stats.stackexchange.com/questions/108734/alternative-to-mape-when-the-data-is-not-a-time-s
https://stats.stackexchange.com/questions/322276/is-mase-specified-only-to-time-series-data
https://stackoverflow.com/questions/31197726/calculate-mase-with-cross-sectional-non-time-series-da
https://stats.stackexchange.com/questions/401759/how-can-mase-mean-absolute-scaled-error-score-valu

Predictive R-squared:
https://www.r-bloggers.com/2014/05/can-we-do-better-than-r-squared/

Other accuracy: accuracyAtCutoff(), accuracyAtEachCutoff(), nomogrammer(), optimalCutoff(), posttestOdds()

Examples

```
# Prepare Data
data("USArrests")

# Calculate Accuracy
accuracyOverall(predicted = USArrests$Assault, actual = USArrests$Murder)
wisdomOfCrowd(predicted = USArrests$Assault, actual = 200)
```

addText — *Add Correlation to Scatterplot.*

Description

Add correlation text to scatterplot.

Usage

```
addText(
  x,
  y,
  xcoord = NULL,
  ycoord = NULL,
  size = 1,
  col = NULL,
  method = "pearson"
)
```

Arguments

x	vector of the variable for the x-axis.
y	vector of the variable for the y-axis.

`xcoord`	x-coordinate for the location of the text.
`ycoord`	y-coordinate for the location of the text.
`size`	size of the text font.
`col`	color of the text font.
`method`	method for calculating the association. One of: • `"pearson"` = Pearson product moment correlation coefficient • `"spearman"` = Spearman's rho • `"kendall"` = Kendall's tau

Details

Adds a correlation coefficient and associated p-value to a scatterplot.

Value

Correlation coefficient, degrees of freedom, and p-value printed on scatterplot.

See Also

Other plot: `plot2WayInteraction()`, `ppPlot()`, `semPlotInteraction()`, `vwReg()`

Other correlations: `cor.table()`, `crossTimeCorrelation()`, `crossTimeCorrelationDF()`, `partialcor.table()`, `vwReg()`

Examples

```
# Prepare Data
data("USArrests")

# Scatterplot
plot(USArrests$Assault, USArrests$Murder)
addText(x = USArrests$Assault, y = USArrests$Murder)
```

`apa` *APA Format*

Description

Format decimals and leading zeroes. Adapted from the MOTE package.

Usage

```
apa(value, decimals = 3, leading = TRUE)
```

Arguments

`value`	A set of numeric values, either a single number, vector, or set of columns.
`decimals`	The number of decimal points desired in the output.
`leading`	Logical value: `TRUE` for leading zeroes on decimals and `FALSE` for no leading zeroes on decimals. The default is `TRUE`.

Details

Formats decimals and leading zeroes for creating reports in scientific style, to be consistent with American Psychological Association (APA) format. This function creates "pretty" character vectors from numeric variables for printing as part of a report. The value can take a single number, matrix, vector, or multiple columns from a data frame, as long as they are numeric. The values will be coerced into numeric if they are characters or logical values, but this process may result in an error if values are truly alphabetical.

Value

Value(s) in the format specified, with the number of decimals places indicated and with or without a leading zero, as indicated.

See Also

`https://github.com/doomlab/MOTE`

Other formatting: `pValue()`, `specify_decimal()`, `suppressLeadingZero()`

Examples

```
apa(value = 0.54674, decimals = 3, leading = TRUE)
```

attenuationCorrelation *Attenuation of True Correlation Due to Measurement Error.*

Description

Estimate the observed association between the predictor and criterion after accounting for the degree to which a true correlation is attenuated due to measurement error.

Usage

```
attenuationCorrelation(
  trueAssociation,
  reliabilityOfPredictor,
  reliabilityOfCriterion
)
```

Arguments

`trueAssociation`	Magnitude of true association (*r* value).
`reliabilityOfPredictor`	Reliability of predictor (from 0 to 1).
`reliabilityOfCriterion`	Reliability of criterion/outcome (from 0 to 1).

Details

Estimate the association that would be observed between the predictor and criterion after accounting for the degree to which a true correlation is attenuated due to random measurement error (unreliability).

Value

Observed correlation between predictor and criterion.

See Also

Other correlation: `disattenuationCorrelation()`

Examples

```
attenuationCorrelation(
  trueAssociation = .7,
  reliabilityOfPredictor = .9,
  reliabilityOfCriterion = .85)
```

`cleanUpNames` *Clean Up Player Names For Merging.*

Description

Cleans up names of players for merging.

Usage

```
cleanUpNames(name)
```

Arguments

`name`	character vector of player names.

Details

Cleans up names of NFL Football players, including making them all-caps, removing common suffixes, punctuation, spaces, etc. This is helpful for merging multiple datasets.

Value

Vector of cleaned player names.

Examples

```
oldNames <- c("Peyton Manning","Tom Brady","Marvin Harrison Jr.")
cleanNames <- cleanUpNames(oldNames)
cleanNames
```

`columnBindFill` *Column Bind and Fill.*

Description

Column bind dataframes and fill with `NA`s.

Usage

```
columnBindFill(...)
```

Arguments

`...`	Names of multiple dataframes.

Details

Binds columns of two or more dataframes together, and fills in missing rows.

Value

Dataframe with columns binded together.

See Also

`https://stackoverflow.com/questions/7962267/cbind-a-dataframe-with-an-empty-dataframe-cbind-fill/7962286#7962286`

Other dataManipulation: `convert.magic()`, `dropColsWithAllNA()`, `dropRowsWithAllNA()`, `varsDifferentTypes()`

Examples

```
# Prepare Data
df1 <- data.frame(a = rnorm(5), b = rnorm(5))
df2 <- data.frame(c = rnorm(4), d = rnorm(4))

# Column Bind and Fill
columnBindFill(df1, df2)
```

complement — *Simulate Complement Variable.*

Description

Simulate data with a specified correlation in relation to an existing variable.

Usage

```
complement(y, rho, x)
```

Arguments

`y`	The existing variable against which to simulate a complement variable.
`rho`	The correlation magnitude, ranging from [-1, 1].
`x`	(optional) Vector with the same length as `y`. Used for calculating the residuals of the least squares regression of `x` against `y`, to remove the `y` component from `x`.

Details

Simulates data with a specified correlation in relation to an existing variable.

Value

Vector of a variable that has a specified correlation in relation to a given variable `y`.

See Also

https://stats.stackexchange.com/a/313138/20338

Other simulation: `simulateAUC()`, `simulateIndirectEffect()`

Examples

```
v1 <- rnorm(100)
complement(y = v1, rho = .5)
complement(y = v1, rho = -.5)

v2 <- complement(y = v1, rho = .85)
plot(v1, v2)
```

`convert.magic` *Convert Variable Types.*

Description

Converts variable types of multiple columns of a dataframe at once.

Usage

```
convert.magic(obj, type)
```

Arguments

`obj`	name of dataframe (object)
`type`	type to convert variables to one of: • `"character"` • `"numeric"` • `"factor"`

Details

Converts variable types of multiple columns of a dataframe at once. Convert variable types to character, numeric, or factor.

Value

Dataframe with columns converted to a particular type.

See Also

https://stackoverflow.com/questions/11261399/function-for-converting-dataframe-column-type/11263399#11263399

Other dataManipulation: `columnBindFill()`, `dropColsWithAllNA()`, `dropRowsWithAllNA()`, `varsDifferentTypes()`

Other conversion: `convertHoursAMPM()`, `convertToHours()`, `convertToMinutes()`, `convertToSeconds()`, `percentileToTScore()`, `pom()`

Examples

```
# Prepare Data
data("USArrests")

# Convert variables to character
convert.magic(USArrests, "character")
```

convertHoursAMPM *Convert AM and PM Hours.*

Description

Convert hours to 24-hour time.

Usage

```
convertHoursAMPM(hours, ampm, am = 0, pm = 1, treatMorningAsLate = FALSE)
```

Arguments

hours	The vector of times in hours.
ampm	Vector indicating whether given times are AM or PM.
am	Value indicating AM in ampm variable.
pm	Value indicating PM in ampm variable.
treatMorningAsLate	TRUE or FALSE indicating whether to treat morning times as late (e.g., 1 AM would be considered a late bedtime, i.e., 25 hours, not an early bedtime).

Details

Convert hours to the number of hours in 24-hour time. You can specify whether to treat morning hours (e.g., 1 AM) as late (25 H), e.g., for specifying late bedtimes

Value

Hours in 24-hour-time.

See Also

Other times: convertToHours(), convertToMinutes(), convertToSeconds()

Other conversion: convert.magic(), convertToHours(), convertToMinutes(), convertToSeconds(), percentileToTScore(), pom()

Examples

```
# Prepare Data
df1 <- data.frame(hours = c(1, 1, 12, 12), ampm = c(0, 0, 1, 1))
df2 <- data.frame(hours = c(1, 1, 12, 12), ampm = c(1, 1, 0, 0))

# Convert AM and PM hours
convertHoursAMPM(hours = df1$hours, ampm = df1$ampm)
convertHoursAMPM(hours = df1$hours, ampm = df1$ampm,
  treatMorningAsLate = TRUE)
```

```
convertHoursAMPM(hours = df2$hours, ampm = df2$ampm, am = 1, pm = 0)
convertHoursAMPM(hours = df2$hours, ampm = df2$ampm, am = 1, pm = 0,
  treatMorningAsLate = TRUE)
```

`convertToHours` *Convert Time to Hours.*

Description

Convert times to hours.

Usage

```
convertToHours(hours, minutes, seconds, HHMMSS, HHMM)
```

Arguments

`hours`	Character vector of the number of hours.
`minutes`	Character vector of the number of minutes.
`seconds`	Character vector of the number of seconds.
`HHMMSS`	Times in HH:MM:SS format.
`HHMM`	Character vector of times in HH:MM format.

Details

Converts times to hours. To convert times to minutes or seconds, see convertToMinutes or convert-ToSeconds.

Value

Vector of times in hours.

See Also

Other times: `convertHoursAMPM()`, `convertToMinutes()`, `convertToSeconds()`

Other conversion: `convert.magic()`, `convertHoursAMPM()`, `convertToMinutes()`, `convertToSeconds()`, `percentileToTScore()`, `pom()`

Examples

```
# Prepare Data
df <- data.frame(hours = c(0,1), minutes = c(15,27), seconds = c(30,13),
  HHMMSS = c("00:15:30","01:27:13"), HHMM = c("00:15","01:27"))

# Convert to Hours
convertToHours(hours = df$hours, minutes = df$minutes, seconds = df$seconds)
convertToHours(HHMMSS = df$HHMMSS)
convertToHours(HHMM = df$HHMM)
```

convertToMinutes *Convert Time to Minutes.*

Description

Convert times to minutes.

Usage

```
convertToMinutes(hours, minutes, seconds, HHMMSS, HHMM, MMSS)
```

Arguments

`hours`	Character vector of the number of hours.
`minutes`	Character vector of the number of minutes.
`seconds`	Character vector of the number of seconds.
`HHMMSS`	Times in HH:MM:SS format.
`HHMM`	Character vector of times in HH:MM format.
`MMSS`	Character vector of times in MM:SS format.

Details

Converts times to minutes. To convert times to hours or seconds, see convertToHours or convert-ToSeconds.

Value

Vector of times in minutes.

See Also

Other times: `convertHoursAMPM()`, `convertToHours()`, `convertToSeconds()`

Other conversion: `convert.magic()`, `convertHoursAMPM()`, `convertToHours()`, `convertToSeconds()`, `percentileToTScore()`, `pom()`

Examples

```
# Prepare Data
df <- data.frame(hours = c(0,1), minutes = c(15,27), seconds = c(30,13),
  HHMMSS = c("00:15:30","01:27:13"), HHMM = c("00:15","01:27"))

# Convert to Minutes
convertToMinutes(hours = df$hours, minutes = df$minutes,
  seconds = df$seconds)
convertToMinutes(HHMMSS = df$HHMMSS)
convertToMinutes(HHMM = df$HHMM)
```

convertToSeconds — *Convert Time to Seconds.*

Description

Convert times to seconds.

Usage

```
convertToSeconds(hours, minutes, seconds, HHMMSS, HHMM, MMSS)
```

Arguments

`hours`	Character vector of the number of hours.
`minutes`	Character vector of the number of minutes.
`seconds`	Character vector of the number of seconds.
`HHMMSS`	Times in HH:MM:SS format.
`HHMM`	Character vector of times in HH:MM format.
`MMSS`	Character vector of times in MM:SS format.

Details

Converts times to seconds. To convert times to hours or minutes, see convertToHours or convert-ToMinutes.

Value

Vector of times in seconds.

See Also

Other times: `convertHoursAMPM()`, `convertToHours()`, `convertToMinutes()`

Other conversion: `convert.magic()`, `convertHoursAMPM()`, `convertToHours()`, `convertToMinutes()`, `percentileToTScore()`, `pom()`

Examples

```
# Prepare Data
df <- data.frame(hours = c(0,1), minutes = c(15,27), seconds = c(30,13),
  HHMMSS = c("00:15:30","01:27:13"), HHMM = c("00:15","01:27"),
  MMSS = c("15:30","87:13"))

# Convert to Minutes
convertToSeconds(hours = df$hours, minutes = df$minutes,
  seconds = df$seconds)
convertToSeconds(HHMMSS = df$HHMMSS)
convertToSeconds(HHMM = df$HHMM)
convertToSeconds(MMSS = df$MMSS)
```

`cor.table` *Correlation Matrix.*

Description

Function that creates a correlation matrix similar to SPSS output.

Usage

```
cor.table(x, y, type = "none", dig = 2, correlation = "pearson")
```

Arguments

`x`	Variable or set of variables in the form of a vector or dataframe to correlate with `y` (if `y` is specified) in an any asymmetric correlation matrix or with itself in a symmetric correlation matrix (if `y` is not specified).
`y`	(optional) Variable or set of variables in the form of a vector or dataframe to correlate with `x`.
`type`	Type of correlation matrix to print. One of: • `"none"` = correlation matrix with *r*, *n*, *p*-values • `"latex"` = generates latex code for correlation matrix with only *r*-values • `"latexSPSS"` = generates latex code for full SPSS-style correlation matrix • `"manuscript"` = only *r*-values, 2 digits; works with `x` only (cannot enter variables for `y`) • `"manuscriptBig"` = only *r*-values, 2 digits, no asterisks; works with `x` only (cannot enter variables for `y`) • `"manuscriptLatex"` = generates latex code for: only *r*-values, 2 digits; works with `x` only (cannot enter variables for `y`) • `"manuscriptBigLatex"` = generates latex code for: only *r*-values, 2 digits, no asterisks; works with `x` only (cannot enter variables for `x`)
`dig`	Number of decimals to print.
`correlation`	Method for calculating the association. One of: • `"pearson"` = Pearson product moment correlation coefficient • `"spearman"` = Spearman's rho • `"kendall"` = Kendall's tau

Details

Co-created by Angela Staples (astaples@emich.edu) and Isaac Petersen (isaac-t-petersen@uiowa.edu). For a partial correlation matrix, see partialcor.table.

Value

A correlation matrix.

See Also

Other correlations: addText(), crossTimeCorrelation(), crossTimeCorrelationDF(), partialcor.table(), vwReg()

Examples

```
# Prepare Data
data("mtcars")

# Correlation Matrix
cor.table(mtcars[,c("mpg","cyl","disp")])
cor.table(mtcars[,c("mpg","cyl","disp")])
cor.table(mtcars[,c("mpg","cyl","disp")], dig = 3)
cor.table(mtcars[,c("mpg","cyl","disp")], dig = 3, correlation = "spearman")

cor.table(mtcars[,c("mpg","cyl","disp")], type = "manuscript", dig = 3)
cor.table(mtcars[,c("mpg","cyl","disp")], type = "manuscriptBig")

table1 <- cor.table(mtcars[,c("mpg","cyl","disp")], type = "latex")
table2 <- cor.table(mtcars[,c("mpg","cyl","disp")], type = "latexSPSS")
table3 <- cor.table(mtcars[,c("mpg","cyl","disp")], type = "manuscriptLatex")
table4 <- cor.table(mtcars[,c("mpg","cyl","disp")], type = "manuscriptBigLatex")

cor.table(mtcars[,c("mpg","cyl","disp")], mtcars[,c("drat","qsec")])
cor.table(mtcars[,c("mpg","cyl","disp")], mtcars[,c("drat","qsec")], type = "manuscript", dig = 3)
```

crossTimeCorrelation *Cross-Time Correlations.*

Description

Calculate the association of a variable across multiple time points.

Usage

```
crossTimeCorrelation(id = "tcid", time = "age", variable, data)
```

Arguments

id	Name of variable indicating the participant ID.
time	Name of variable indicating the timepoint.
variable	Name of variable to estimate the cross-time correlation.
data	Dataframe.

Details

Calculate the association of a variable across multiple time points. It is especially useful when there are three or more time points.

Value

output of `cor.test()`

See Also

Other correlations: `addText()`, `cor.table()`, `crossTimeCorrelationDF()`, `partialcor.table()`, `vwReg()`

Examples

```
# Prepare Data
df <- expand.grid(ID = 1:100, time = c(1, 2, 3))
df <- df[order(df$ID),]
row.names(df) <- NULL
df$score <- rnorm(nrow(df))

# Cross-Time Correlation
crossTimeCorrelation(id = "ID", time = "time", variable = "score", data = df)
```

`crossTimeCorrelationDF` *Cross-Time Correlations Dataframe.*

Description

Dataframe used to compute cross-time correlations.

Usage

```
crossTimeCorrelationDF(id = "tcid", time = "age", variable, data)
```

Arguments

`id`	Name of variable indicating the participant ID.
`time`	Name of variable indicating the timepoint.
`variable`	Name of variable to estimate the cross-time correlation.
`data`	Dataframe.

Details

Dataframe used to calculate the association of a variable across multiple time points. It is especially useful when there are three or more time points.

Value

dataframe with three columns in the form of: `ID`, `time1`, `time2`

See Also

Other correlations: addText(), cor.table(), crossTimeCorrelation(), partialcor.table(), vwReg()

Examples

```
# Prepare Data
df <- expand.grid(ID = 1:100, time = c(1, 2, 3))
df <- df[order(df$ID),]
row.names(df) <- NULL
df$score <- rnorm(nrow(df))

# Cross-Time Correlation
crossTimeCorrelationDF(id = "ID", time = "time", variable = "score", data = df)
```

---

deriv_d_negBinom *Item and Test Information from Zero-Inflated Negative Binomial Model.*

Description

Estimate item and test information from Bayesian zero-inflated negative binomial model that was fit using the brms package.

Usage

```
deriv_d_negBinom(n, alpha, beta, theta, phi)

d_negBinom(n, alpha, beta, theta, phi)

log_gen_binom(n, phi)

deriv_logd_negBinom(n, alpha, beta, theta, phi)

info_neg_binom_analytical(
  theta = seq(-2.5, 2.5, length.out = 101),
  alpha,
  beta,
  phi,
  varpi
)

item_info_NB_zero_analytical(theta, alpha, beta, phi, varpi)

item_info_NB_analytical(theta, alpha, beta, phi, varpi = NULL, zero = FALSE)

test_info_NB(theta, alpha, beta, phi, varpi = NULL, zero = FALSE)
```

```
error_variance_NB(
  lower = -Inf,
  upper = Inf,
  alpha,
  beta,
  phi,
  varpi = NULL,
  zero = FALSE
)

reliability_NB(alpha, beta, phi, varpi = NULL, zero = FALSE)
```

Arguments

`n`	Integer. The observed count, representing the the event frequency.
`alpha`	Numeric. The slope/discrimination parameter of the item, indicating how steeply the item response changes with the person's (`theta`).
`beta`	Numeric. The intercept/easiness parameter of the item, indicating the expected count at a given level on the construct (`theta`).
`theta`	Numeric. The respondent's level on the latent factor/construct.
`phi`	Numeric. The shape/overdispersion parameter of the negative binomial distribution, indicating the variance beyond what is expected from a negative binomial distribution.
`varpi`	Numeric. The probability of observing a zero count due to a separate zero-inflation process.
`zero`	TRUE/FALSE. Whether the item is a from a zero-inflated model.
`lower`	Numeric. The lower range of theta, for estimating error variance or reliability.
`upper`	Numeric. The upper range of theta, for estimating error variance or reliability.

Details

Created by Philipp Doebler (doebler@statistik.tu-dortmund.de) and Loreen Sabel (loreen.sabel@tu-dortmund.de).

Value

The amount of information for a given item (or the test as a whole) at each of the values of `theta` specified. Based on test information, one can estimate error variance and marginal reliability using `error_variance_NB()` and `reliability_NB()`, respectively.

See Also

Other bayesian: `pA()`

Other IRT: `discriminationToFactorLoading()`, `fourPL()`, `itemInformation()`, `reliabilityIRT()`, `standardErrorIRT()`, `test_info_4PL()`

Examples

```
## Not run:
library("brms")
library("rstan")

coef_bayesianMixedEffectsGRM_gam <- coef(bayesianMixedEffectsGRM_gam)
str(coef_bayesianMixedEffectsGRM_gam)
itempars <- coef_bayesianMixedEffectsGRM_gam$item[,1,1:4]

# define a grid of thetas for the computations:
theta_seq <- seq(-4, 4, length.out = 201)

# item information for all items
# The resulting matrix has length(theta_seq) columns and a row per item.
# We use a loop for the calculations
item_info <- matrix(NA, nrow = nrow(itempars), ncol = length(theta_seq))
for(i in 1:nrow(itempars)){
  item_info[i, ] <- item_info_NB_zero_analytical(
    theta = theta_seq,
    alpha = itempars[i, "alpha_Intercept"],
    beta = itempars[i, "beta_Intercept"],
    phi = exp(itempars[i, "shape_Intercept"]),
    varpi = plogis(itempars[i, "zi_Intercept"]))
}

test_info <- data.frame(
  theta = theta_seq,
  testInformation = colSums(item_info)
)

# Or, alternatively:
test_info_NB(
  theta = compareTestInfo$theta,
  alpha = itempars[,"alpha_Intercept"],
  beta = itempars[,"beta_Intercept"],
  phi = exp(itempars[,"shape_Intercept"]),
  varpi = plogis(itempars[,"zi_Intercept"]),
  zero = TRUE)

# Test Standard Error of Measurement in Different Theta Ranges
error_variance_NB(
  lower = -4,
  upper = 4,
  alpha = itempars[,"alpha_Intercept"],
  beta = itempars[,"beta_Intercept"],
  phi = exp(itempars[,"shape_Intercept"]),
  varpi = plogis(itempars[,"zi_Intercept"]),
  zero = TRUE
)

error_variance_NB(
  lower = -4,
  upper = 0,
  alpha = itempars[,"alpha_Intercept"],
  beta = itempars[,"beta_Intercept"],
  phi = exp(itempars[,"shape_Intercept"]),
  varpi = plogis(itempars[,"zi_Intercept"]),
  zero = TRUE
)

error_variance_NB(
  lower = 0,
  upper = 1.5,
  alpha = itempars[,"alpha_Intercept"],
  beta = itempars[,"beta_Intercept"],
  phi = exp(itempars[,"shape_Intercept"]),
  varpi = plogis(itempars[,"zi_Intercept"]),
  zero = TRUE
)

error_variance_NB(
  lower = 1.5,
  upper = 4,
  alpha = itempars[,"alpha_Intercept"],
  beta = itempars[,"beta_Intercept"],
  phi = exp(itempars[,"shape_Intercept"]),
  varpi = plogis(itempars[,"zi_Intercept"]),
  zero = TRUE
)

# One-Number Summary of Test Reliability
reliability_NB(
  alpha = itempars[,"alpha_Intercept"],
  beta = itempars[,"beta_Intercept"],
  phi = exp(itempars[,"shape_Intercept"]),
  varpi = plogis(itempars[,"zi_Intercept"]),
  zero = TRUE)

## End(Not run)
```

`disattenuationCorrelation` *Disattenuation of Observed Correlation Due to Measurement Error.*

Description

Estimate the true association between the predictor and criterion after accounting for the degree to which a true correlation is attenuated due to measurement error.

Usage

```
disattenuationCorrelation(
    observedAssociation,
    reliabilityOfPredictor,
    reliabilityOfCriterion
  )
```

Arguments

`observedAssociation`
: Magnitude of observed association (r value).

`reliabilityOfPredictor`
: Reliability of predictor (from 0 to 1).

`reliabilityOfCriterion`
: Reliability of criterion/outcome (from 0 to 1).

Details

Estimate the true association between the predictor and criterion after accounting for the degree to which a true correlation is attenuated due to random measurement error (unreliability).

Value

True association between predictor and criterion.

See Also

Other correlation: `attenuationCorrelation()`

Examples

```
disattenuationCorrelation(
  observedAssociation = .7,
  reliabilityOfPredictor = .9,
  reliabilityOfCriterion = .85)
```

`discriminationToFactorLoading`
Discrimination (IRT) to Standardized Factor Loading.

Description

Convert a discrimination parameter in item response theory to a standardized factor loading.

Usage

```
discriminationToFactorLoading(a, model = "probit")
```

Arguments

`a`	Discrimination parameter in item response theory.
`model`	Model type. One of: • `"logit"` • `"probit"`

Details

Convert a discrimination parameter in item response theory to a standardized factor loading

Value

Standardized factor loading.

See Also

https://aidenloe.github.io/introToIRT.html https://stats.stackexchange.com/questions/228629/conversion-of-irt-logit-discrimination-parameter-to-factor-loading-metric

Other IRT: `deriv_d_negBinom()`, `fourPL()`, `itemInformation()`, `reliabilityIRT()`, `standardErrorIRT()`, `test_info_4PL()`

Examples

```
discriminationToFactorLoading(0.5)
discriminationToFactorLoading(1.3)
discriminationToFactorLoading(1.3, model = "logit")
```

`dropColsWithAllNA` *Drop NA columns.*

Description

Drop columns with all missing (`NA`) values.

Usage

```
dropColsWithAllNA(data, ignore = NULL)
```

Arguments

`data`	Dataframe to drop columns from.
`ignore`	Names of columns to ignore for determining whether each row had all missing values.

Details

Drop columns that have no observed values, i.e., all values in the column are missing (NA), excluding the ignored columns.

Value

A dataframe with columns removed that had all missing values in non-ignored columns.

See Also

Other dataManipulation: columnBindFill(), convert.magic(), dropRowsWithAllNA(), varsDifferentTypes()

Other dataEvaluations: dropRowsWithAllNA(), is.nan.data.frame(), not_all_na(), not_any_na()

Examples

```
# Prepare Data
df <- expand.grid(ID = 1:100, time = c(1, 2, 3), rater = c(1, 2),
naCol1 = NA, naCol2 = NA)
df <- df[order(df$ID),]
row.names(df) <- NULL
df$score1 <- rnorm(nrow(df))
df$score2 <- rnorm(nrow(df))
df$score3 <- rnorm(nrow(df))
df[sample(1:nrow(df), size = 100), c("score1","score2","score3")] <- NA

# Drop Rows with All NA in Non-Ignored Columns
dropColsWithAllNA(df)
dropColsWithAllNA(df, ignore = c("naCol2"))
```

---

dropRowsWithAllNA *Drop NA rows.*

Description

Drop rows with all missing (NA) values.

Usage

```
dropRowsWithAllNA(data, ignore = NULL)
```

Arguments

data	Dataframe to drop rows from.
ignore	Names of columns to ignore for determining whether each row had all missing values.

Details

Drop rows that have no observed values, i.e., all values in the row are missing (NA), excluding the ignored columns.

Value

A dataframe with rows removed that had all missing values in non-ignored columns.

See Also

Other dataManipulation: columnBindFill(), convert.magic(), dropColsWithAllNA(), varsDifferentTypes()

Other dataEvaluations: dropColsWithAllNA(), is.nan.data.frame(), not_all_na(), not_any_na()

Examples

```
# Prepare Data
df <- expand.grid(ID = 1:100, time = c(1, 2, 3))
df <- df[order(df$ID),]
row.names(df) <- NULL
df$score1 <- rnorm(nrow(df))
df$score2 <- rnorm(nrow(df))
df$score3 <- rnorm(nrow(df))
df[sample(1:nrow(df), size = 100), c("score1","score2","score3")] <- NA

# Drop Rows with All NA in Non-Ignored Columns
dropRowsWithAllNA(df, ignore = c("ID","time"))
```

---

equiv_chi *Chi-Square Equivalence Test for Structural Equation Models.*

Description

Function that performs a chi-square equivalence test for structural equation models.

Usage

```
equiv_chi(alpha = 0.05, chi, df, m, N_sample, popRMSEA = 0.08)
```

Arguments

alpha	Value of the significance level, which is set to .05 by default.
chi	Value of the observed chi-square test statistic.
df	Number of model (or model difference in) degrees of freedom.
m	Number of groups.
N_sample	Sample size.
popRMSEA	The value of the root-mean square error of approximation (RMSEA) to set for the equivalence bounds, which is set to .08 by default.

Details

Created by Counsell et al. (2020): Counsell, A., Cribbie, R. A., & Flora, D. B. (2020). Evaluating equivalence testing methods for measurement invariance. Multivariate Behavioral Research, 55(2), 312-328. https://doi.org/10.1080/00273171.2019.1633617

Value

p-value indicating whether to reject the null hypothesis that the model is a poor fit to the data.

See Also

Other structural equation modeling: `make_esem_model()`, `puc()`, `satorraBentlerScaledChiSquareDifferenceTestStat` `semPlotInteraction()`

Examples

```
# Prepare Data
data("mtcars")

# Fit structural equation model

# Extract statistics
N1 <- 1222
m <- 1
Tml1 <- 408.793
df1 <- 80

# Equivalence test
equiv_chi(alpha = .05, chi = Tml1, df = df1, m = 1, N_sample = N1, popRMSEA = .08)
```

`fourPL` *4-Parameter Logistic Curve.*

Description

4-parameter logistic curve for item response theory.

Usage

```
fourPL(a = 1, b, c = 0, d = 1, theta)
```

Arguments

`a`	Discrimination parameter (slope).
`b`	Difficulty (severity) parameter (inflection point).
`c`	Guessing parameter (lower asymptote).
`d`	Careless errors parameter (upper asymptote).
`theta`	Person’s level on the construct.

Details

Estimates the probability of item endorsement as function of the four-parameter logistic (4PL) curve and the person's level on the construct (theta).

Value

Probability of item endorsement (or expected value on the item).

See Also

doi:10.1177/0146621613475471

Other IRT: `deriv_d_negBinom()`, `discriminationToFactorLoading()`, `itemInformation()`, `reliabilityIRT()`, `standardErrorIRT()`, `test_info_4PL()`

Examples

```
fourPL(b = 2, theta = -4:4) #1PL
fourPL(b = 2, a = 1.5, theta = -4:4) #2PL
fourPL(b = 2, a = 1.5, c = 0.10, theta = -4:4) #3PL
fourPL(b = 2, a = 1.5, c = 0.10, d = 0.95, theta = -4:4) #4PL
```

`getDependencies` *Package Dependencies.*

Description

Determine package dependencies.

Usage

```
getDependencies(packs)
```

Arguments

`packs`	Character vector of names of target packages.

Details

Determine which packages depend on a target package (or packages).

Value

Vector of packages that depend on the target package(s).

See Also

https://stackoverflow.com/questions/52929114/install-packages-in-r-without-internet-connection-wit
52935020#52935020

Other packages: load_or_install()

Examples

```
old <- options("repos")
options(repos = "https://cran.r-project.org")
getDependencies("tidyverse")
options(old)
```

imputationCombine *Combine Results from Mixed Effect Imputation Models.*

Description

Function that combines lme results across multiple imputation runs.

Usage

```
imputationCombine(model, dig = 3)
```

Arguments

model	name of lme() model object with multiply imputed data.
dig	number of decimals to print in output.

Details

[INSERT].

Value

Summary of model fit and information for mixed effect imputation models.

See Also

Other multipleImputation: imputationModelCompare(), imputationPRV(), lmCombine()

Examples

```
#INSERT
```

imputationModelCompare — *Compare Mixed Effect Imputation Models.*

Description

Function that compares two nested `lme()` models from multiple imputation using likelihood ratio test.

Usage

```
imputationModelCompare(model1, model2)
```

Arguments

`model1`	name of first `lme()` model object with multiply imputed data.
`model2`	name of second `lme()` model object with multiply imputed data.

Details

[INSERT].

Value

Likelihood ratio test for model comparison of two mixed effect imputation models.

See Also

Other multipleImputation: `imputationCombine()`, `imputationPRV()`, `lmCombine()`

Examples

```
#INSERT
```

imputationPRV — *Proportional Reduction of Variance from Imputation Models.*

Description

Calculate the proportional reduction of variance in imputation models.

Usage

```
imputationPRV(baseline, full, baselineTime = 1, fullTime = 1)
```

Arguments

`baseline`	The baseline model object fit with the imputed data.
`full`	The full model object fit with the imputed data.
`baselineTime`	The position of the random effect of time (random slopes) among the random slopes in the baseline model. For example: • `0` = no random slopes • `1` = time is the 1st random effect • `2` = time is the second random effect
`fullTime`	The position of the random effect of time (random slopes) among the random slopes in the full model. For example: • `0` = no random slopes • `1` = time is the 1st random effect • `2` = time is the second random effect

Details

[INSERT].

Value

The proportional reduction of variance from a baseline mixed-effects model to a full mixed effects model.

See Also

Other multipleImputation: `imputationCombine()`, `imputationModelCompare()`, `lmCombine()`

Examples

```
#INSERT
```

`is.nan.data.frame` *NaN (Not a Number).*

Description

Check whether a value is "Not A Number" (`NaN`) in a dataframe.

Usage

```
## S3 method for class 'data.frame'
is.nan(x)
```

Arguments

`x`	Dataframe.

Details

[INSERT].

Value

TRUE or FALSE, indicating whether values in a dataframe are Not a Number (NA).

See Also

https://stackoverflow.com/questions/18142117/how-to-replace-nan-value-with-zero-in-a-huge-data-fra
18143097#18143097

Other dataEvaluations: dropColsWithAllNA(), dropRowsWithAllNA(), not_all_na(), not_any_na()

Examples

```
# Prepare Data
df <- data.frame(item1 = rnorm(1000), item2 = rnorm(1000), item3 = rnorm(1000))
df[sample(1:nrow(df), size = 100), c("item1","item2","item3")] <- NaN

# Calculate Missingness-Adjusted Row Sum
is.nan(df)
```

---

itemInformation *Item Information.*

Description

Item information in item response theory.

Usage

```
itemInformation(a = 1, b, c = 0, d = 1, theta)
```

Arguments

a	Discrimination parameter (slope).
b	Difficulty (severity) parameter (inflection point).
c	Guessing parameter (lower asymptote).
d	Careless errors parameter (upper asymptote).
theta	Person's level on the construct.

Details

Estimates the amount of information provided by a given item as function of the item parameters and the person's level on the construct (theta).

Value

Amount of item information.

See Also

doi:10.1177/0146621613475471

Other IRT: `deriv_d_negBinom()`, `discriminationToFactorLoading()`, `fourPL()`, `reliabilityIRT()`, `standardErrorIRT()`, `test_info_4PL()`

Examples

```
itemInformation(b = 2, theta = -4:4) #1PL
itemInformation(b = 2, a = 1.5, theta = -4:4) #2PL
itemInformation(b = 2, a = 1.5, c = 0.10, theta = -4:4) #3PL
itemInformation(b = 2, a = 1.5, c = 0.10, d = 0.95, theta = -4:4) #4PL
```

`kish_ess` *Weighted Quantiles.*

Description

Computes weighted quantiles. `whdquantile()` uses a weighted Harrell-Davis quantile estimator. `wthdquantile()` uses a weighted trimmed Harrell-Davis quantile estimator. `wquantile()` uses a weighted traditional quantile estimator.

Usage

```
kish_ess(w)

wquantile_generic(x, w, probs, cdf)

whdquantile(x, w, probs)

wthdquantile(x, w, probs, width = 1/sqrt(kish_ess(w)))

wquantile(x, w, probs, type = 7)
```

Arguments

`w`	Numeric vector of weights to give each value. Should be the same length as the vector of values.
`x`	Numeric vector of values of which to determine the quantiles.
`probs`	Numeric vector of the quantiles to retrieve.
`cdf`	Cumulative distribution function.
`width`	Numeric value for the width of the interval in the trimmed Harrell-Davis quantile estimator.
`type`	Numeric value for type of weighted quantile.

Details

Computes weighted quantiles according to Akinshin (2023).

Value

Numeric vector of specified quantiles.

See Also

doi:10.48550/arXiv.2304.07265

Other computations: `Mode`(), `meanSum`(), `mySum`()

Examples

```
mydata <- c(1:100, 1000)
mydataWithNAs <- mydata
mydataWithNAs[c(1,5,7)] <- NA
weights <- rep(1, length(mydata))
quantiles <- c(0.01, 0.05, 0.1, 0.25, 0.5, 0.75, 0.9, 0.95, 0.99)

whdquantile(
  x = mydata,
  w = weights,
  probs = quantiles)

wthdquantile(
  x = mydata,
  w = weights,
  probs = quantiles)

wquantile(
  x = mydata,
  w = weights,
  probs = quantiles)

whdquantile(
  x = mydataWithNAs,
  w = weights,
  probs = quantiles)

wthdquantile(
  x = mydataWithNAs,
  w = weights,
  probs = quantiles)

wquantile(
  x = mydataWithNAs,
  w = weights,
  probs = quantiles)
```

lmCombine *Combine Results from Multiple Regression Imputation Models.*

Description

Function that combines `lm()` results across multiple imputation runs.

Usage

```
lmCombine(model, dig = 3)
```

Arguments

`model`	name of `lm()` model object with multiply imputed data.
`dig`	number of decimals to print in output.

Details

[INSERT].

Value

Summary of multiple regression imputation models.

See Also

Other multipleImputation: `imputationCombine()`, `imputationModelCompare()`, `imputationPRV()`

Other multipleRegression: `plot2WayInteraction()`, `ppPlot()`, `semPlotInteraction()`, `update_nested()`

Examples

```
#INSERT
```

lmeSummary *Summarize mixed effects model.*

Description

Summarizes the results of a model fit by the `lme()` function of the `nlme` package.

Usage

```
lmeSummary(model, dig = 3)
```

Arguments

`model`	name of `lme()` model object.
`dig`	number of decimals to print in output.

Details

Summarizes the results of a model fit by the `lme()` function of the `nlme` package. Includes summary of parameters, pseudo-r-squared, and whether model is positive definite.

Value

Output summary of `lme()` model object.

Examples

```
# Fit Model
library("nlme")
model <- lme(distance ~ age + Sex, data = Orthodont, random = ~ 1 + age)

# Model Summary
summary(model)
lmeSummary(model)
```

`load_or_install` *Load or Install Packages.*

Description

Loads packages or, if not already installed, installs and loads packages.

Usage

```
load_or_install(package_names, ...)
```

Arguments

`package_names`	Character vector of one or more package names.
`...`	Additional arguments for `install.packages()`.

Details

Loads packages that are already installed, and if the packages are not already installed, it installs and then loads them.

Value

Loaded packages.

See Also

https://www.r-bloggers.com/2012/05/loading-andor-installing-packages-programmatically/
https://stackoverflow.com/questions/4090169/elegant-way-to-check-for-missing-packages-and-install-

Other packages: `getDependencies()`

Examples

```
## Not run:
old <- options("repos")
options(repos = "https://cran.r-project.org")
# Warning: the command below installs packages that are not already installed
load_or_install(c("tidyverse","nlme"))
options(old)

## End(Not run)
```

`make_esem_model` *Make ESEM Model.*

Description

Make `lavaan` syntax for exploratory structural equation model (ESEM).

Usage

```
make_esem_model(loadings, anchors)
```

Arguments

`loadings`	Dataframe with three columns from exploratory factor analysis (EFA): • `latent` = name of the latent factor(s) • `item` = name of the item(s)/indicator(s) • `loading` = parameter estimate of the factor loading item factor loading on the latent factor
`anchors`	Dataframe whose names are the latent factors and whose values are the names of the anchor item for each latent factor.

Details

Makes syntax for exploratory structural equation model (ESEM) to be fit in `lavaan`.

Value

`lavaan` model syntax.

See Also

https://msilvestrin.me/post/esem/

Other structural equation modeling: equiv_chi(), puc(), satorraBentlerScaledChiSquareDifferenceTestStatistic(
semPlotInteraction()

Examples

```
# Prepare Data
data("HolzingerSwineford1939", package = "lavaan")

# Specify EFA Syntax
efa_syntax <- '
  # EFA Factor Loadings
  efa("efa1")*f1 +
  efa("efa1")*f2 +
  efa("efa1")*f3 =~ x1 + x2 + x3 + x4 + x5 + x6 + x7 + x8 + x9
'

# Fit EFA Model
mplusRotationArgs <- list(rstarts = 30,
  row.weights = "none",
  algorithm = "gpa",
  orthogonal = FALSE,
  jac.init.rot = TRUE,
  std.ov = TRUE, # row standard = correlation
  geomin.epsilon = 0.0001)

efa_fit <- lavaan::sem(
  efa_syntax,
  data = HolzingerSwineford1939,
  information = "observed",
  missing = "ML",
  estimator = "MLR",
  rotation = "geomin",
  # mimic Mplus
  meanstructure = TRUE,
  rotation.args = mplusRotationArgs)

# Extract Factor Loadings
esem_loadings <- lavaan::parameterEstimates(
  efa_fit,
  standardized = TRUE
) |>
  dplyr::filter(efa == "efa1") |>
  dplyr::select(lhs, rhs, est) |>
  dplyr::rename(item = rhs, latent = lhs, loading = est)

# Specify Anchor Item for Each Latent Factor
anchors <- c(f1 = "x3", f2 = "x5", f3 = "x7")

# Generate ESEM Syntax
esemModel_syntax <- make_esem_model(esem_loadings, anchors)
```

```
# Fit ESEM Model
lavaan::sem(
  esemModel_syntax,
  data = HolzingerSwineford1939,
  missing = "ML",
  estimator = "MLR")
```

meanSum *Mean Sum.*

Description

Compute a missingness-adjusted row sum.

Usage

```
meanSum(x)
```

Arguments

x	Matrix or dataframe with participants in the rows and items in the columns.

Details

Take row mean across columns (items) and then multiply by number of items to account for missing (NA) values.

Value

Missingness-adjusted row sum.

See Also

Other computations: Mode(), kish_ess(), mySum()

Examples

```
# Prepare Data
df <- data.frame(item1 = rnorm(1000), item2 = rnorm(1000), item3 = rnorm(1000))

# Calculate Missingness-Adjusted Row Sum
df$missingnessAdjustedSum <- meanSum(df)
```

`Mode` *Statistical Mode.*

Description

Calculate statistical mode.

Usage

```
Mode(x, multipleModes = "all")
```

Arguments

`x`	Numerical vector.
`multipleModes`	How to handle multiple modes. One of: • `"mean"` = if there are multiple modes, take the mean of all modes • `"first"` = if there are multiple modes, select the first mode • `"all"` = if there are multiple modes, return all modes

Details

Calculates statistical mode(s).

Value

Statistical mode(s).

See Also

`https://stackoverflow.com/questions/2547402/how-to-find-the-statistical-mode/8189441#8189441`

Other computations: `kish_ess()`, `meanSum()`, `mySum()`

Examples

```
# Prepare Data
v1 <- c(1, 1, 2, 2, 3)

#Calculate Statistical Mode
Mode(v1)
Mode(v1, multipleModes = "mean")
Mode(v1, multipleModes = "first")
```

`mortgage` *Mortgage Principal and Interest.*

Description

Amount of principal and interest payments on a mortgage.

Usage

```
mortgage(balance, interest, term = 30, n = 12)
```

Arguments

`balance`	Initial mortgage balance.
`interest`	Interest rate.
`term`	Payoff period (in years).
`n`	Number of payments per year.

Details

Calculates the amount of principal and interest payments on a mortgage.

Value

Amount of principal and interest payments.

Examples

```
mortgage(balance = 300000, interest = .05)
mortgage(balance = 300000, interest = .04)
mortgage(balance = 300000, interest = .06)
mortgage(balance = 300000, interest = .05, term = 15)
```

`mySum` *My Sum.*

Description

Compute a row sum and retain `NA`s when all values in the row are `NA`.

Usage

```
mySum(data)
```

Arguments

`data`	dataframe

Details

Compute a row sum and set the row sum to be missing (not zero) when all values in the row are missing (`NA`).

Value

Modified row sum to set row sum to be missing when all values in the row are missing (`NA`).

See Also

Other computations: `Mode()`, `kish_ess()`, `meanSum()`

Examples

```
# Prepare Data
df <- data.frame(item1 = rnorm(1000), item2 = rnorm(1000), item3 = rnorm(1000))
df[sample(1:nrow(df), size = 100), c("item1","item2","item3")] <- NA

# Calculate Missingness-Adjusted Row Sum
df$sum <- mySum(df)
```

`my_loadings_sorter` *Sorts loadings from exploratory factor analysis.*

Description

Sorts items' loadings based on their loadings from exploratory factor analysis fit with the `psych::fa()` function.

Usage

```
my_loadings_sorter(
  fit,
  sort_type = "largest_loading",
  nchar = 40,
  return_blocks = FALSE,
  showlatentcor = TRUE,
  itemLabels = NULL
)
```

Arguments

`fit`	the fitted object from the `psych::fa()` function
`sort_type`	how to sort the loadings. One of: • "largest_loading": sorts items by the largest loading • "largest_loading_but_first": sorts items by the largest loading, ignoring the loading on the first factor • "first": sorts items by the largest loading on the first factor
`nchar`	the limit for the number of characters to display for the item label
`return_blocks`	whether to return the block number that corresponds to each item
`showlatentcor`	whether or not to print the intercorrelation among the latent factors (only possible for models with an oblique rotation)
`itemLabels`	a vector of the item labels

Details

Adapted from code by Philipp Doebler (doebler@statistik.tu-dortmund.de).

Value

Sorted loadings from exploratory factor analysis model.

`nomogrammer` *Create Nomogram.*

Description

Create nomogram plot.

Usage

```
nomogrammer(
  TP = NULL,
  TN = NULL,
  FP = NULL,
  FN = NULL,
  pretestProb = NULL,
  selectionRate = NULL,
  SN = NULL,
  SP = NULL,
  FPR = NULL,
  PLR = NULL,
  NLR = NULL,
  Detail = FALSE,
  NullLine = FALSE,
  LabelSize = (14/5),
  Verbose = FALSE
)
```

Arguments

`TP`	Number of true positive cases.
`TN`	Number of true negative cases.
`FP`	Number of false positive cases.
`FN`	Number of false negative cases.
`pretestProb`	Pretest probability (prevalence/base rate/prior probability) of characteristic, as a number between 0 and 1.
`selectionRate`	Selection rate (marginal probability of positive test), as a number between 0 and 1.
`SN`	Sensitivity of the test at a given cut point, as a number between 0 and 1.
`SP`	Specificity of the test at a given cut point, as a number between 0 and 1.
`FPR`	False positive rate of the test at a given cut point, as a number between 0 and 1.
`PLR`	Positive likelihood ratio of the test at a given cut point.
`NLR`	Positive likelihood ratio of the test at a given cut point.
`Detail`	If `TRUE`, overlay key statistics onto the plot.
`NullLine`	If `TRUE`, add a line from prior prob through LR = 1.
`LabelSize`	Label size.
`Verbose`	Print out relevant metrics in the console.

Details

Create nomogram plot from the following at a given cut point:

- 1) true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN)
- 2) pretest probability (pretestProb), sensitivity (SN), and specificity (SP), OR
- 3) pretest probability (pretestProb), sensitivity (SN), and false positive rate (FPR), OR
- 4) pretest probability (pretestProb), sensitivity (SN), and selection rate (selectionRate), OR
- 5) pretest probability (pretestProb), positive likelihood ratio (PLR), and negative likelihood ratio (NLR)

Value

ggplot object of nomogram plot.

See Also

`https://github.com/achekroud/nomogrammer`

Other accuracy: `accuracyAtCutoff()`, `accuracyAtEachCutoff()`, `accuracyOverall()`, `optimalCutoff()`, `posttestOdds()`

Examples

```
nomogrammer(
  TP = 253,
  TN = 386,
  FP = 14,
  FN = 347)

nomogrammer(
  pretestProb = .60,
  SN = 0.421,
  SP = 0.965)

nomogrammer(
  pretestProb = .60,
  SN = 0.421,
  FPR = 0.035)

nomogrammer(
  pretestProb = .60,
  SN = 0.421,
  selectionRate = 0.267)

nomogrammer(
  pretestProb = .60,
  PLR = 12,
  NLR = 0.6)
```

`not_all_na` *Any Rows Not NA.*

Description

Check if any rows for a column are not `NA`.

Usage

```
not_all_na(x)
```

Arguments

`x`	vector or column

Details

Determine whether any rows for a column (or vector) are not missing (`NA`).

Value

`TRUE` or `FALSE`

See Also

Other dataEvaluations: `dropColsWithAllNA()`, `dropRowsWithAllNA()`, `is.nan.data.frame()`, `not_any_na()`

Examples

```
# Prepare Data
data("USArrests")

# Check if any rows are not NA
not_all_na(USArrests$Murder)
```

`not_any_na` *Not Any NA.*

Description

Check if all rows for a column are `NA`.

Usage

```
not_any_na(x)
```

Arguments

`x`	column vector

Details

[INSERT].

Value

`TRUE` or `FALSE`, indicating whether the whole column does not have any missing values (`NA`).

See Also

Other dataEvaluations: `dropColsWithAllNA()`, `dropRowsWithAllNA()`, `is.nan.data.frame()`, `not_all_na()`

Examples

```
# Prepare Data
df <- data.frame(item1 = rnorm(1000), item2 = rnorm(1000), item3 = rnorm(1000))
df[sample(1:nrow(df), size = 100), "item2"] <- NA
df[,"item3"] <- NA

# Check if Not Any NA
not_any_na(df$item1)
not_any_na(df$item2)
not_any_na(df$item3)
```

---

optimalCutoff *Optimal Cutoff.*

Description

Find the optimal cutoff for different aspects of accuracy. Actuals should be binary, where `1` = present and `0` = absent.

Usage

```
optimalCutoff(predicted, actual, UH = NULL, UM = NULL, UCR = NULL, UFA = NULL)
```

Arguments

`predicted`	vector of continuous predicted values.
`actual`	vector of binary actual values (`1` = present and `0` = absent).
`UH`	(optional) utility of hits (true positives), specified as a value from 0-1, where 1 is the most highly valued and 0 is the least valued.
`UM`	(optional) utility of misses (false negatives), specified as a value from 0-1, where 1 is the most highly valued and 0 is the least valued.
`UCR`	(optional) utility of correct rejections (true negatives), specified as a value from 0-1, where 1 is the most highly valued and 0 is the least valued.
`UFA`	(optional) utility of false positives (false positives), specified as a value from 0-1, where 1 is the most highly valued and 0 is the least valued.

Details

Identify the optimal cutoff for different aspects of accuracy of predicted values in relation to actual values by specifying the predicted values and actual values. Optionally, you can specify the utility of hits, misses, correct rejections, and false alarms to calculate the overall utility of each possible cutoff.

Value

The optimal cutoff and optimal accuracy index at that cutoff based on:

- `percentAccuracy` = percent accuracy
- `percentAccuracyByChance` = percent accuracy by chance
- `RIOC` = relative improvement over chance
- `relativeImprovementOverPredictingFromBaseRate` = relative improvement over predicting from the base rate
- `PPV` = positive predictive value

- `NPV` = negative predictive value
- `youdenJ` = Youden's J statistic
- `balancedAccuracy` = balanced accuracy
- `f1Score` = F1-score
- `mcc` = Matthews correlation coefficient
- `diagnosticOddsRatio` = diagnostic odds ratio
- `positiveLikelihoodRatio` = positive likelihood ratio
- `negativeLikelhoodRatio` = negative likelihood ratio
- `dPrimeSDT` = d-Prime index from signal detection theory
- `betaSDT` = beta index from signal detection theory
- `cSDT` = c index from signal detection theory
- `aSDT` = a index from signal detection theory
- `bSDT` = b index from signal detection theory
- `differenceBetweenPredictedAndObserved` = difference between predicted and observed values
- `informationGain` = information gain
- `overallUtility` = overall utility (if utilities were specified)

See Also

Other accuracy: `accuracyAtCutoff()`, `accuracyAtEachCutoff()`, `accuracyOverall()`, `nomogrammer()`, `posttestOdds()`

Examples

```
# Prepare Data
data("USArrests")
USArrests$highMurderState <- NA
USArrests$highMurderState[which(USArrests$Murder >= 10)] <- 1
USArrests$highMurderState[which(USArrests$Murder < 10)] <- 0

# Determine Optimal Cutoff
optimalCutoff(predicted = USArrests$Assault,
  actual = USArrests$highMurderState)
optimalCutoff(predicted = USArrests$Assault,
  actual = USArrests$highMurderState,
  UH = 1, UM = 0, UCR = .9, UFA = 0)
```

pA — *Bayes' Theorem.*

Description

Estimate marginal and conditional probabilities using Bayes theorem.

Usage

```
pA(pAgivenB, pB, pAgivenNotB)

pB(pBgivenA, pA, pBgivenNotA)

pAgivenB(pBgivenA, pA, pB = NULL, pBgivenNotA = NULL)

pBgivenA(pAgivenB, pB, pA = NULL, pAgivenNotB = NULL)

pAgivenNotB(pAgivenB, pA, pB)

pBgivenNotA(pBgivenA, pA, pB)
```

Arguments

`pAgivenB`	The conditional probability of `A` given `B`.
`pB`	The marginal probability of event `B`.
`pAgivenNotB`	The conditional probability of `A` given NOT `B`.
`pBgivenA`	The conditional probability of `B` given `A`.
`pA`	The marginal probability of event `A`.
`pBgivenNotA`	The conditional probability of `B` given NOT `A`.

Details

Estimates marginal or conditional probabilities using Bayes theorem.

Value

The requested marginal or conditional probability. One of:

- the marginal probability of `A`
- the marginal probability of `B`
- the conditional probability of `A` given `B`
- the conditional probability of `B` given `A`
- the conditional probability of `A` given NOT `B`
- the conditional probability of `B` given NOT `A`

See Also

Other bayesian: `deriv_d_negBinom()`

Examples

```
pA(pAgivenB = .95, pB = .285, pAgivenNotB = .007171515)

pB(pBgivenA = .95, pA = .285, pBgivenNotA = .007171515)

pAgivenB(pBgivenA = .95, pA = .285, pB = .2758776)
pAgivenB(pBgivenA = .95, pA = .285, pBgivenNotA = .007171515)
pAgivenB(pBgivenA = .95, pA = .003, pBgivenNotA = .007171515)

pBgivenA(pAgivenB = .95, pB = .285, pA = .2758776)
pBgivenA(pAgivenB = .95, pB = .285, pAgivenNotB = .007171515)
pBgivenA(pAgivenB = .95, pB = .003, pAgivenNotB = .007171515)

pAgivenNotB(pAgivenB = .95, pB = .003, pA = .01)

pBgivenNotA(pBgivenA = .95, pA = .003, pB = .01)
```

`partialcor.table` *Partial Correlation Matrix.*

Description

Function that creates a partial correlation matrix similar to SPSS output.

Usage

```
partialcor.table(
  x,
  y,
  z = NULL,
  type = "none",
  dig = 2,
  correlation = "pearson"
)
```

Arguments

`x`	Variable or set of variables in the form of a vector or dataframe to correlate with `y` (if `y` is specified) in an any asymmetric correlation matrix or with itself in a symmetric correlation matrix (if `y` is not specified).
`y`	(optional) Variable or set of variables in the form of a vector or dataframe to correlate with `x`.
`z`	Covariate(s) to partial out from association.

`type`	Type of correlation matrix to print. One of: • `"none"` = correlation matrix with *r*, *n*, *p*-values • `"latex"` = generates latex code for correlation matrix with only *r*-values • `"latexSPSS"` = generates latex code for full SPSS-style correlation matrix • `"manuscript"` = only *r*-values, 2 digits; works with `x` only (cannot enter variables for `y`) • `"manuscriptBig"` = only *r*-values, 2 digits, no asterisks; works with `x` only (cannot enter variables for `y`) • `"manuscriptLatex"` = generates latex code for: only *r*-values, 2 digits; works with `x` only (cannot enter variables for `y`) • `"manuscriptBigLatex"` = generates latex code for: only *r*-values, 2 digits, no asterisks; works with `x` only (cannot enter variables for `x`)
`dig`	Number of decimals to print.
`correlation`	Method for calculating the association. One of: • `"pearson"` = Pearson product moment correlation coefficient • `"spearman"` = Spearman's rho • `"kendall"` = Kendall's tau

Details

Co-created by Angela Staples (astaples@emich.edu) and Isaac Petersen (isaac-t-petersen@uiowa.edu). Creates a partial correlation matrix, controlling for one or more covariates. For a standard correlation matrix, see cor.table.

Value

A partial correlation matrix.

See Also

Other correlations: `addText()`, `cor.table()`, `crossTimeCorrelation()`, `crossTimeCorrelationDF()`, `vwReg()`

Examples

```
# Prepare Data
data("mtcars")

#Correlation Matrix
partialcor.table(mtcars[,c("mpg","cyl","disp")], z = mtcars$hp)
partialcor.table(mtcars[,c("mpg","cyl","disp")], z = mtcars[,c("hp","wt")])
partialcor.table(mtcars[,c("mpg","cyl","disp")], z = mtcars[,c("hp","wt")],
  dig = 3)
partialcor.table(mtcars[,c("mpg","cyl","disp")], z = mtcars[,c("hp","wt")],
  dig = 3, correlation = "spearman")

partialcor.table(mtcars[,c("mpg","cyl","disp")], z = mtcars[,c("hp","wt")],
  type = "manuscript", dig = 3)
```

```
partialcor.table(mtcars[,c("mpg","cyl","disp")], z = mtcars[,c("hp","wt")],
  type = "manuscriptBig")

table1 <- partialcor.table(mtcars[,c("mpg","cyl","disp")],
  z = mtcars[,c("hp","wt")], type = "latex")
table2 <- partialcor.table(mtcars[,c("mpg","cyl","disp")],
  z = mtcars[,c("hp","wt")], type = "latexSPSS")
table3 <- partialcor.table(mtcars[,c("mpg","cyl","disp")],
  z = mtcars[,c("hp","wt")], type = "manuscriptLatex")
table4 <- partialcor.table(mtcars[,c("mpg","cyl","disp")],
  z = mtcars[,c("hp","wt")], type = "manuscriptBigLatex")

partialcor.table(mtcars[,c("mpg","cyl","disp")], mtcars[,c("drat","qsec")],
  mtcars[,c("hp","wt")])
partialcor.table(mtcars[,c("mpg","cyl","disp")], mtcars[,c("drat","qsec")],
  mtcars[,c("hp","wt")], type = "manuscript", dig = 3)
```

`percentEffort` *Person Months.*

Description

Calculate perons months for personnel effort in grants.

Usage

```
percentEffort(
  academicMonths = NULL,
  calendarMonths = NULL,
  summerMonths = NULL,
  appointment = 9
)

personMonths(
  academicMonths = NULL,
  calendarMonths = NULL,
  summerMonths = NULL,
  effortAcademic = NULL,
  effortCalendar = NULL,
  effortSummer = NULL,
  appointment = 9
)
```

Arguments

`academicMonths`	The number of academic months.
`calendarMonths`	The number of calendar months.
`summerMonths`	The number of summer months.

`appointment`	The duration (in months) of one's annual appointment; used as the denominator for determining the timeframe out of which the academic months occur. Default is a 9-month appointment.
`effortAcademic`	Percent effort (in proportion) during academic months.
`effortCalendar`	Percent effort (in proportion) during calendar months.
`effortSummer`	Percent effort (in proportion) during summer months.

Details

Calculate person months for personnel effort in grant proposals from academic months, calendar months, and summer months.

Value

The person months of effort.

See Also

https://nexus.od.nih.gov/all/2015/05/27/how-do-you-convert-percent-effort-into-person-months/

Examples

```
# Specify Values
appointmentDuration <- 9 #(in months)

# Specify either Set 1 (months) or Set 2 (percent effort) below:

#Set 1: Months
academicMonths <- 1.3 #AY (academic year) months (should be between 0 to appointmentDuration)
calendarMonths <- 0 #CY (calendar year) months (should be between 0-12)
summerMonths <- 0.5 #SM (summer) months (should be between 0 to [12-appointmentDuration])

# Set 2: Percent Effort
percentEffortAcademic <- 0.1444444 #(a proportion; should be between 0-1)
percentEffortCalendar <- 0 #(a proportion; should be between 0-1)
percentEffortSummer <- 0.1666667 #(a proportion; should be between 0-1)

# Calculations
summerDuration <- 12 - appointmentDuration

# Percent effort (in proportion)
percentEffort(academicMonths = academicMonths)
percentEffort(calendarMonths = calendarMonths)
percentEffort(summerMonths = summerMonths)

# Person-Months From NIH Website
(percentEffort(academicMonths = academicMonths) * appointmentDuration) +
 (percentEffort(calendarMonths = calendarMonths) * 12) +
 (percentEffort(summerMonths = summerMonths) * summerDuration)

# Person-Months from Academic/Calendar/Summer Months
```

```
personMonths(academicMonths = academicMonths,
             calendarMonths = calendarMonths,
             summerMonths = summerMonths)

# Person-Months from Percent Effort
personMonths(effortAcademic = percentEffortAcademic,
             effortCalendar = percentEffortCalendar,
             effortSummer = percentEffortSummer)
```

`percentileToTScore` *Percentile to T-Score Conversion.*

Description

Conversion of percentile ranks to T-scores.

Usage

```
percentileToTScore(percentileRank)
```

Arguments

`percentileRank`	Vector of percentile ranks.

Details

Converts percentile ranks to the equivalent T-scores.

Value

Vector of T-scores.

See Also

Other conversion: `convert.magic()`, `convertHoursAMPM()`, `convertToHours()`, `convertToMinutes()`, `convertToSeconds()`, `pom()`

Examples

```
percentileRanks <- c(1, 10, 20, 30, 40, 50, 60, 70, 80, 90, 99)

percentileToTScore(percentileRanks)
```

`plot2WayInteraction` *Plot 2-way interaction.*

Description

Generates a plot of a 2-way interaction.

Usage

```
plot2WayInteraction(
  predictor,
  outcome,
  moderator,
  predictorLabel = "predictor",
  outcomeLabel = "outcome",
  moderatorLabel = "moderator",
  varList,
  varTypes,
  values = NA,
  interaction = "normal",
  legendLabels = NA,
  legendLocation = "topright",
  ylim = NA,
  pvalues = TRUE,
  data
)
```

Arguments

`predictor`	character name of predictor variable (variable on x-axis).
`outcome`	character name of outcome variable (variable on y-axis).
`moderator`	character name of moderator variable (variable on z-axis).
`predictorLabel`	label on x-axis of plot
`outcomeLabel`	label on y-axis of plot
`moderatorLabel`	label on z-axis of plot
`varList`	names of predictor variables in model
`varTypes`	types of predictor variables in model; one of: • `"mean"` = plots at mean of variable – should be used for ALL covariates (apart from main predictor and moderator) • `"sd"` = plots at +/- 1 sd of variable (for most continuous predictors and moderators) • `"binary"` = plots at values of 0,1 (for binary predictors and moderators) • `"full"` = plots full range of variable (for variables like age when on x-axis) • `"values"` = allows plotting moderator at specific values (e.g., `0`, `1`, `2`)

	• `"factor"` = plots moderator at different categories (e.g., `TRUE`, `FALSE`)
`values`	specifies values at which to plot moderator (must specify varType = `"values"`)
`interaction`	one of: • `"normal"` = standard interaction • `"meancenter"` = calculates the interaction from a mean-centered predictor and moderator (subtracting each individual's value from the variable mean to set the mean of the variable to zero) • `"orthogonalize"` = makes the interaction orthogonal to the predictor and moderator by regressing the interaction on the predictor and outcome and saving the residual
`legendLabels`	vector of 2 labels for the two levels of the moderator; leave as `NA` to see the actual levels of the moderator
`legendLocation`	one of: `"topleft"`, `"topright"`, `"bottomleft"`, or `"bottomright"`
`ylim`	vector of min and max values on y-axis (e,g., `0`, `10`)
`pvalues`	whether to include p-values of each slope in plot (`TRUE` or `FALSE`)
`data`	name of data object

Details

Generates a plot of a 2-way interaction: the association between a predictor and an outcome at two levels of the moderator.

Value

Plot of two-way interaction.

See Also

Other plot: `addText()`, `ppPlot()`, `semPlotInteraction()`, `vwReg()`

Other multipleRegression: `lmCombine()`, `ppPlot()`, `semPlotInteraction()`, `update_nested()`

Examples

```
# Prepare Data
predictor <- rnorm(1000, 10, 3)
moderator <- rnorm(1000, 50, 10)
outcome <- (1.7 * predictor) + (1.3 * moderator) +
  (1.5 * predictor * moderator) + rnorm(1000, sd = 3)
covariate <- rnorm(1000)
df <- data.frame(predictor, moderator, outcome, covariate)

# Linear Regression
lmModel <- lm(outcome ~ predictor + moderator + predictor:moderator,
  data = df, na.action = "na.exclude")
summary(lmModel)

# 1. Plot 2-Way Interaction
```

```
plot2WayInteraction(predictor = "predictor",
                    outcome = "outcome",
                    moderator = "moderator",
                    varList = c("predictor","moderator","covariate"),
                    varTypes = c("sd","binary","mean"),
                    data = df)

# 2. Specify y-axis Range
plot2WayInteraction(predictor = "predictor",
                    outcome = "outcome",
                    moderator = "moderator",
                    varList = c("predictor","moderator","covariate"),
                    varTypes = c("sd","binary","mean"),
                    ylim = c(10,50),                                #new
                    data = df)

# 3. Add Variable Labels
plot2WayInteraction(predictor = "predictor",
                    outcome = "outcome",
                    moderator = "moderator",
                    varList = c("predictor","moderator","covariate"),
                    varTypes = c("sd","binary","mean"),
                    ylim = c(10,50),
                    predictorLabel = "Stress",                      #new
                    outcomeLabel = "Aggression",                    #new
                    moderatorLabel = "Gender",                      #new
                    data = df)

# 4. Change Legend Labels
plot2WayInteraction(predictor = "predictor",
                    outcome = "outcome",
                    moderator = "moderator",
                    varList = c("predictor","moderator","covariate"),
                    varTypes = c("sd","binary","mean"),
                    ylim = c(10,50),
                    predictorLabel = "Stress",
                    outcomeLabel = "Aggression",
                    moderatorLabel = "Gender",
                    legendLabels = c("Boys","Girls"),               #new
                    data = df)

# 5. Move Legend Location
plot2WayInteraction(predictor = "predictor",
                    outcome = "outcome",
                    moderator = "moderator",
                    varList = c("predictor","moderator","covariate"),
                    varTypes = c("sd","binary","mean"),
                    ylim = c(10,50),
                    predictorLabel = "Stress",
                    outcomeLabel = "Aggression",
                    moderatorLabel = "Gender",
                    legendLabels = c("Boys","Girls"),
                    legendLocation = "topleft",                     #new
                    data = df)

#6. Turn Off p-Values
plot2WayInteraction(predictor = "predictor",
                    outcome = "outcome",
                    moderator = "moderator",
                    varList = c("predictor","moderator","covariate"),
                    varTypes = c("sd","binary","mean"),
                    ylim = c(10,50),
                    predictorLabel = "Stress",
                    outcomeLabel = "Aggression",
                    moderatorLabel = "Gender",
                    legendLabels = c("Boys","Girls"),
                    legendLocation = "topleft",
                    pvalues = FALSE,                              #new
                    data = df)

#7. Get Regression Output from Mean-Centered Predictor and Moderator
plot2WayInteraction(predictor = "predictor",
                    outcome = "outcome",
                    moderator = "moderator",
                    varList = c("predictor","moderator","covariate"),
                    varTypes = c("sd","binary","mean"),
                    ylim = c(10,50),
                    predictorLabel = "Stress",
                    outcomeLabel = "Aggression",
                    moderatorLabel = "Gender",
                    legendLabels = c("Boys","Girls"),
                    legendLocation = "topleft",
                    interaction = "meancenter",                   #new
                    data = df)

#8. Get Regression Output from Orthogonalized Interaction Term
plot2WayInteraction(predictor = "predictor",
                    outcome = "outcome",
                    moderator = "moderator",
                    varList = c("predictor","moderator","covariate"),
                    varTypes = c("sd","binary","mean"),
                    ylim = c(10,50),
                    predictorLabel = "Stress",
                    outcomeLabel = "Aggression",
                    moderatorLabel = "Gender",
                    legendLabels = c("Boys","Girls"),
                    legendLocation = "topleft",
                    interaction = "orthogonalize",                #new
                    data = df)
```

pom *Proportion of Maximum (POM).*

Description

Calculate the proportion of maximum (POM) score given a minimum and maximum score.

Usage

```
pom(data, min = NULL, max = NULL)
```

Arguments

`data`	The vector of data.
`min`	The minimum possible or observed value.
`max`	The maximum possible or observed value.

Details

The minimum and maximum score for calculating the proportion of maximum could be the possible or observed minimum and maximum, respectively. Using the possible minimum and maximum would yield the proportion of maximum possible score. Using the observed minimum and maximum would yield the proportion of minimum and maximum observed score. If the minimum and maximum possible scores are not specified, the observed minimum and maximum are used.

Value

Proportion of maximum possible or observed values.

See Also

Other conversion: `convert.magic()`, `convertHoursAMPM()`, `convertToHours()`, `convertToMinutes()`, `convertToSeconds()`, `percentileToTScore()`

Examples

```
# Prepare Data
v1 <- sample(1:9, size = 1000, replace = TRUE)

# Calculate Proportion of Maximum Possible (by specifying the minimum and maximum possible)
pom(v1, min = 0, max = 10)

# Calculate Proportion of Maximum Observed
pom(v1)
```

posttestOdds *Posttest Odds & Probability.*

Description

Estimate posttest odds and posttest probability.

Usage

```
posttestOdds(
  TP,
  TN,
  FP,
  FN,
  pretestProb = NULL,
  SN = NULL,
  SP = NULL,
  likelihoodRatio = NULL
)

posttestProbability(
  TP,
  TN,
  FP,
  FN,
  pretestProb = NULL,
  SN = NULL,
  SP = NULL,
  likelihoodRatio = NULL
)
```

Arguments

`TP`	Number of true positive cases.
`TN`	Number of true negative cases.
`FP`	Number of false positive cases.
`FN`	Number of false negative cases.
`pretestProb`	Pretest probability (prevalence/base rate/prior probability) of characteristic, as a number between 0 and 1.
`SN`	Sensitivity of the test at a given cut point, as a number between 0 and 1.
`SP`	Specificity of the test at a given cut point, as a number between 0 and 1.
`likelihoodRatio`	Likelihood ratio of the test at a given cut point.

Details

Estimates posttest odds or posttest probability.

Value

The requested posttest odds or pottest probability.

See Also

Other accuracy: `accuracyAtCutoff()`, `accuracyAtEachCutoff()`, `accuracyOverall()`, `nomogrammer()`, `optimalCutoff()`

Examples

```
posttestOdds(
  TP = 26,
  TN = 56,
  FP = 14,
  FN = 14)

posttestOdds(
  pretestProb = 0.3636364,
  SN = 0.65,
  SP = 0.80)

posttestOdds(
  pretestProb = 0.3636364,
  likelihoodRatio = 3.25)

posttestProbability(
  TP = 26,
  TN = 56,
  FP = 14,
  FN = 14)

posttestProbability(
  pretestProb = 0.3636364,
  SN = 0.65,
  SP = 0.80)

posttestProbability(
  pretestProb = 0.3636364,
  likelihoodRatio = 3.25)
```

`ppPlot` *PP Plot.*

Description

Normal Probability (P-P) Plot.

Usage

```
ppPlot(model)
```

Arguments

`model`	The model object of a linear regression model fit using the `lm()` function.

Details

A normal probability (P-P) plot compares the empirical cumulative distribution to the theoretical cumulative distribution.

Value

Normal probability (P-P) plot.

See Also

https://www.r-bloggers.com/2009/12/r-tutorial-series-graphic-analysis-of-regression-assumptions/

Other plot: `addText()`, `plot2WayInteraction()`, `semPlotInteraction()`, `vwReg()`

Other multipleRegression: `lmCombine()`, `plot2WayInteraction()`, `semPlotInteraction()`, `update_nested()`

Examples

```
# Prepare Data
predictor1 <- rnorm(100)
predictor2 <- rnorm(100)
outcome <- rnorm(100)

# Fit Model
lmModel <- lm(outcome ~ predictor1 + predictor2)

# P-P Plot
ppPlot(lmModel)
```

puc *Percent of Uncontaminated Correlations (PUC).*

Description

Percent of uncontaminated correlations (PUC) from bifactor model.

Usage

```
puc(numItems, numSpecificFactors)
```

Arguments

`numItems`	Number of items (or indicators).
`numSpecificFactors`	Number of specific factors.

Details

Estimates the percent of uncontaminated correlations (PUC) from a bifactor model. The PUC represents the percentage of correlations (i.e., covariance terms) that reflect variance from only the general factor (i.e., not variance from a specific factor). Correlations that are explained by the specific factors are considered "contaminated" by multidimensionality.

Value

Percent of Uncontaminated Correlations (PUC).

See Also

doi:10.31234/osf.io/6tf7j doi:10.1177/0013164412449831 doi:10.1037/met0000045

Other structural equation modeling: `equiv_chi()`, `make_esem_model()`, `satorraBentlerScaledChiSquareDifferenceTe`
`semPlotInteraction()`

Examples

```
puc(
  numItems = 9,
  numSpecificFactors = 3
)

mydata <- data.frame(
  numItems = c(9,18,18,36,36,36),
  numSpecificFactors = c(3,3,6,3,6,12)
)

puc(
  numItems = mydata$numItems,
  numSpecificFactors = mydata$numSpecificFactors
)
```

`pValue` *p-values.*

Description

Suppress the leading zero when printing p-values.

Usage

```
pValue(value, digits = 3)
```

Arguments

`value`	The p-value.
`digits`	Number of decimal digits for printing the p-value.

Details

[INSERT].

Value

p-value.

See Also

Other formatting: `apa()`, `specify_decimal()`, `suppressLeadingZero()`

Examples

```
pValue(0.70)
pValue(0.04)
pValue(0.00002)
```

`read.aes` *Read Encrypted Data.*

Description

Read data from encrypted file.

Usage

```
read.aes(filename, key)
```

Arguments

`filename`	Location of encrypted data.
`key`	Encryption key.

Details

Reads data from an encrypted file. To write an data to an encrypted file, see write.aes.

Value

Unencrypted data.

See Also

https://stackoverflow.com/questions/25318800/how-do-i-read-an-encrypted-file-from-disk-with-r/25321586#25321586

Other encrypted: `write.aes()`

Examples

```
# Location of Encryption Key on Local Computer (where only you should have access to it)
#encryptionKeyLocation <- file.path(getwd(), "/encryptionKey.RData",
#  fsep = "") #Can change to a different path, e.g.: "C:/Users/[USERNAME]/"

# Generate a Temporary File Path for Encryption Key
encryptionKeyLocation <- tempfile(fileext = ".RData")

# Generate Encryption Key
key <- as.raw(sample(1:16, 16))

# Save Encryption Key
save(key, file = encryptionKeyLocation)

# Specify Credentials
credentials <- "Insert My Credentials Here"

# Generate a Temporary File Path for Encrypted Credentials
encryptedCredentialsLocation <- tempfile(fileext = ".txt")

# Save Encrypted Credentials
#write.aes(
#  df = credentials,
#  filename = file.path(getwd(), "/encrypytedCredentials.txt", fsep = ""),
#  key = key) # Change the file location to save this on the lab drive

write.aes(
  df = credentials,
  filename = encryptedCredentialsLocation,
  key = key)

rm(credentials)
rm(key)

# Read and Unencrypt the Credentials Using the Encryption Key
load(encryptionKeyLocation)

#credentials <- read.aes(
#  filename = file.path(getwd(), "/encrypytedCredentials.txt", fsep = ""),
#  key = key)
```

```
credentials <- read.aes(
  filename = encryptedCredentialsLocation,
  key = key)
```

`recode_intensity` *Recode Intensity.*

Description

Recode intensity of behavior based on frequency of behavior.

Usage

```
recode_intensity(intensity, did_not_occur = NULL, frequency = NULL)

mark_intensity_as_zero(
  item_names,
  data,
  did_not_occur_vars = NULL,
  frequency_vars = NULL
)
```

Arguments

`intensity`	The intensity of the behavior.
`did_not_occur`	Whether or not the behavior did NOT occur. If `0`, the behavior did occur (in the given timeframe). If `1`, the behavior did not occur in (in the given timeframe).
`frequency`	The frequency of the behavior.
`item_names`	The names of the questionnaire items.
`data`	The data object.
`did_not_occur_vars`	The name(s) of the variables corresponding to whether the behavior did not occur in the past year (`did_not_occur`).
`frequency_vars`	The name(s) of the variables corresponding to the number of occurrences (`num_occurrences`).

Details

Recodes the intensity of behavior to zero if the frequency of the behavior is zero (i.e., if the behavior has not occurred).

Value

The intensity of the behavior.

redcapProgressBar *Progress Bar for REDCap.*

Description

Function that identifies the values for a progress bar in REDCap.

Usage

```
redcapProgressBar(numSurveys, beginning = 2, end = 99)
```

Arguments

numSurveys	the number of surveys to establish progress.
beginning	the first value to use in the sequence.
end	the last value to use in the sequence.

Details

A progress bar in REDCap can be created using the following code:

```
Progress:
<div style="width:100%;border:0;margin:0;padding:0;background-color:
#A9BAD1;text-align:center;"><div style="width:2%;border: 0;margin:0;
padding:0;background-color:#8491A2"><span style="color:#8491A2">.
</span></div></div>
```

where `width:2%` specifies the progress (out of 100%).

Value

sequence of numbers for the progress bar in REDCap.

Examples

```
redcapProgressBar(numSurveys = 6)
redcapProgressBar(6)
redcapProgressBar(4)
redcapProgressBar(numSurveys = 7, beginning = 1, end = 99)
```

`reliabilityIRT` *Reliability (IRT).*

Description

Estimate the reliability in item response theory.

Usage

```
reliabilityIRT(information, varTheta = 1)
```

Arguments

`information`	Test information.
`varTheta`	Variance of theta.

Details

Estimate the reliability in item response theory using the test information (i.e., the sum of all items' information).

Value

Reliability for that amount of test information.

See Also

`https://groups.google.com/g/mirt-package/c/ZAgpt6nq5V8/m/R3OEeEqdAQAJ`

Other IRT: `deriv_d_negBinom()`, `discriminationToFactorLoading()`, `fourPL()`, `itemInformation()`, `standardErrorIRT()`, `test_info_4PL()`

Examples

```
# Calculate information for 4 items
item1 <- itemInformation(b = -2, a = 0.6, theta = -4:4)
item2 <- itemInformation(b = -1, a = 1.2, theta = -4:4)
item3 <- itemInformation(b = 1, a = 1.5, theta = -4:4)
item4 <- itemInformation(b = 2, a = 2, theta = -4:4)

items <- data.frame(item1, item2, item3, item4)

# Calculate test information
items$testInformation <- rowSums(items)

# Estimate reliability
reliabilityIRT(items$testInformation)
```

reliabilityOfDifferenceScore

Reliability of Difference Score.

Description

Estimate the reliability of a difference score.

Usage

```
reliabilityOfDifferenceScore(x, y, reliabilityX, reliabilityY)
```

Arguments

x	Vector of one variable that is used in the computation of difference score.
y	Vector of second variable that is used in the computation of the difference score.
reliabilityX	The reliability of the x variable.
reliabilityY	The reliability of the y variable.

Details

Estimates the reliability of a difference score.

Value

Reliability of the difference score that is computed from the difference of x and y.

See Also

Other reliability: repeatability()

Examples

```
v1 <- rnorm(1000, mean = 100, sd = 15)
v2 <- v1 + rnorm(1000, mean = 1, sd = 15)
reliabilityOfDifferenceScore(x = v1, y = v2,
 reliabilityX = .7, reliabilityY = .8)
```

`repeatability` *Repeatability.*

Description

Estimate the repeatability of a measure's scores across two time points.

Usage

```
repeatability(measure1, measure2)
```

Arguments

`measure1`	Vector of scores from the measure at time 1.
`measure2`	Vector of scores from the measure at time 2.

Details

Estimates the coefficient of repeatability (CR), bias, and the lower and upper limits of agreement (LOA).

Value

Dataframe with the coefficient of repeatability (`CR`), bias, the lower limit of agreement (`lowerLOA`), and the upper limit of agreement (`upperLOA`). Also generates a Bland-Altman plot with a solid black reference line (indicating a difference of zero), a dashed red line indicating the bias, and dashed blue lines indicating the limits of agreement.

See Also

Other reliability: `reliabilityOfDifferenceScore()`

Examples

```
v1 <- rnorm(1000, mean = 100, sd = 15)
v2 <- v1 + rnorm(1000, mean = 1, sd = 3)
repeatability(v1, v2)
```

`reverse_score` *Reverse Score Variables.*

Description

Reverse score variables using either the theoretical min and max, or the observed max.

Usage

```
reverse_score(
  data,
  variables,
  theoretical_max = NULL,
  theoretical_min = NULL,
  append_string = NULL
)
```

Arguments

`data`	Data object.
`variables`	Names of variables to reverse score.
`theoretical_max`	(Optional): the theoretical maximum score.
`theoretical_min`	(Optional): the theoretical minimum score.
`append_string`	(Optional): a string to append to each variable name.

Details

Reverse scores variables using either the theoretical min and max (by subtracting the theoretical maximum from each score and adding the theoretical minimum to each score) or by subtracting each score from the maximum score for that variable.

Value

Dataframe with reverse-scored variables.

Examples

```
mydata <- data.frame(
  var1 = c(1, 2, NA, 4, 5),
  var2 = c(NA, 4, 3, 2, 1)
)

variables_to_reverse_score <- c("var1", "var2")

reverse_score(
  mydata,
  variables = variables_to_reverse_score)

reverse_score(
  mydata,
  variables = variables_to_reverse_score,
  append_string = ".R")

reverse_score(
  mydata,
  variables = variables_to_reverse_score,
  theoretical_max = 7)

reverse_score(
  mydata,
  variables = variables_to_reverse_score,
  theoretical_max = 7,
  theoretical_min = 1)
```

`satorraBentlerScaledChiSquareDifferenceTestStatistic` *Satorra-Bentler Scaled Chi-Square Difference Test Statistic.*

Description

Function that computes the Satorra-Bentler Scaled Chi-Square Difference Test statistic.

Usage

```
satorraBentlerScaledChiSquareDifferenceTestStatistic(T0, c0, d0, T1, c1, d1)
```

Arguments

`T0`	Value of the chi-square statistic for the nested model.
`c0`	Value of the scaling correction factor for the nested model.
`d0`	Number of model degrees of freedom for the nested model.
`T1`	Value of the chi-square statistic for the comparison model.
`c1`	Value of the scaling correction factor for the comparison model.
`d1`	Number of model degrees of freedom for the comparison model.

Details

Computes the Satorra-Bentler Scaled Chi-Square Difference Test statistic between two structural equation models.

Value

Satorra-Bentler Scaled Chi-Square Difference Test statistic.

See Also

Other structural equation modeling: equiv_chi(), make_esem_model(), puc(), semPlotInteraction()

Examples

```
# Fit structural equation model
HS.model <- '
 visual =~ x1 + x2 + x3
 textual =~ x4 + x5 + x6
 speed =~ x7 + x8 + x9
'

fit1 <- lavaan::cfa(HS.model, data = lavaan::HolzingerSwineford1939,
 estimator = "MLR")
fit0 <- lavaan::cfa(HS.model, data = lavaan::HolzingerSwineford1939,
 orthogonal = TRUE, estimator = "MLR")

# Chi-square difference test
# lavaan::anova(fit1, fit0)
satorraBentlerScaledChiSquareDifferenceTestStatistic(
 T0 = lavaan::fitMeasures(fit0)["chisq.scaled"],
 c0 = lavaan::fitMeasures(fit0)["chisq.scaling.factor"],
 d0 = lavaan::fitMeasures(fit0)["df.scaled"],
 T1 = lavaan::fitMeasures(fit1)["chisq.scaled"],
 c1 = lavaan::fitMeasures(fit1)["chisq.scaling.factor"],
 d1 = lavaan::fitMeasures(fit1)["df.scaled"])
```

---

semPlotInteraction *Plot interaction from SEM model.*

Description

Generates a plot of a 2-way interaction from a structural equation model (SEM) that was estimated using the lavaan package.

Usage

```
semPlotInteraction(
  data,
  fit,
  predictor,
  centered_predictor,
  moderator,
  centered_moderator,
  interaction,
  outcome,
  covariates = NULL,
  predStr = NULL,
  modStr = NULL,
  outStr = NULL
)
```

Arguments

`data`	the dataframe object from which the model was derived
`fit`	the fitted model lavaan object
`predictor`	the variable name of the predictor variable that is in its raw metric (in quotes)
`centered_predictor`	the variable name of the mean-centered predictor variable as it appears in the model object syntax in lavaan (in quotes)
`moderator`	the variable name of the moderator variable that is in its raw metric (in quotes)
`centered_moderator`	the variable name of the mean-centered moderator variable that as it appears in the model object syntax in lavaan (in quotes)
`interaction`	the variable name of the interaction term as it appears in the model object syntax in lavaan (in quotes)
`outcome`	the variable name of the outcome variable as it appears in the model object syntax in lavaan (in quotes)
`covariates`	default NULL; a vector of the names of the covariate variables as they appear in the model object syntax in lavaan (each in quotes)
`predStr`	default NULL; optional addition of an x-axis title for the name of the predictor variable (in quotes); if left unset, plot label will default to "Predictor"
`modStr`	default NULL; optional addition of an z-axis title for the name of the moderator variable (in quotes); if left unset, plot label will default to "Moderator"
`outStr`	default NULL; optional addition of an x-axis title for the name of the outcome variable (in quotes); if left unset, plot label will default to "Outcome"

Details

Created by Johanna Caskey (johanna-caskey@uiowa.edu).

Value

Plot of two-way interaction from structural equation model.

See Also

Other plot: `addText()`, `plot2WayInteraction()`, `ppPlot()`, `vwReg()`

Other multipleRegression: `lmCombine()`, `plot2WayInteraction()`, `ppPlot()`, `update_nested()`

Other structural equation modeling: `equiv_chi()`, `make_esem_model()`, `puc()`, `satorraBentlerScaledChiSquareDiffer`

Examples

```
states <- as.data.frame(state.x77)
names(states)[which(names(states) == "HS Grad")] <- "HS.Grad"
states$Income_rescaled <- states$Income/100

# Mean Center Predictors
states$Illiteracy_centered <- scale(states$Illiteracy, scale = FALSE)
states$Murder_centered <- scale(states$Murder, scale = FALSE)

# Compute Interaction Term
states$interaction <- states$Illiteracy_centered * states$Murder_centered

# Specify model syntax
moderationModel <- '
  Income_rescaled ~ Illiteracy_centered + Murder_centered + interaction +
  HS.Grad
'

# Fit the model
moderationFit <- lavaan::sem(
  moderationModel,
  data = states,
  missing = "ML",
  estimator = "MLR",
  fixed.x = FALSE)

# Pass model to function (unlabeled plot)
semPlotInteraction(
  data = states,
  fit = moderationFit,
  predictor = "Illiteracy",
  centered_predictor = "Illiteracy_centered",
  moderator = "Murder",
  centered_moderator = "Murder_centered",
  interaction = "interaction",
  outcome = "Income_rescaled",
  covariates = "HS.Grad")

# Pass model to function (labeled plot)
semPlotInteraction(
  data = states,
  fit = moderationFit,
  predictor = "Illiteracy",
  centered_predictor = "Illiteracy_centered",
  moderator = "Murder",
  centered_moderator = "Murder_centered",
  interaction = "interaction",
  outcome = "Income_rescaled",
  covariates = "HS.Grad",
  predStr = "Illiteracy Level",
  modStr = "Murder Rate",
  outStr = "Income")
```

`setLabPath` *Set Lab Path.*

Description

Sets the path directory to the lab drive.

Usage

```
setLabPath()
```

Details

Sets the path directory to the lab drive, and saves it in the object `petersenLab`.

Value

The object `petersenLab` with containing the path directory to the lab drive.

Examples

```
petersenLabPath <- setLabPath()
```

`simulateAUC` *Simulate Area Under the ROC Curve (AUC).*

Description

Simulate data with a specified area under the receiver operating characteristic curve—i.e., the AUC of an ROC curve.

Usage

```
simulateAUC(auc, n)
```

Arguments

`auc`	The area under the receiver operating characteristic (ROC) curve.
`n`	The number of observations to simulate.

Details

Simulates data with a specified area under the receiver operating characteristic curve—i.e., the AUC of an ROC curve.

Value

Dataframe with two columns:

- x is the predictor variable.
- y is the dichotomous criterion variable.

See Also

https://stats.stackexchange.com/questions/422926/generate-synthetic-data-given-auc/424213

Other simulation: complement(), simulateIndirectEffect()

Examples

```
simulateAUC(.60, 50000)
simulateAUC(.70, 50000)
simulateAUC(.80, 50000)
simulateAUC(.90, 50000)
simulateAUC(.95, 50000)
simulateAUC(.99, 50000)
```

---

simulateIndirectEffect *Simulate Indirect Effect.*

Description

Simulate indirect effect from mediation analyses.

Usage

```
simulateIndirectEffect(
  N = NA,
  x = NA,
  m = NA,
  XcorM = NA,
  McorY = NA,
  corTotal = NA,
  proportionMediated = NA,
  seed = NA
)
```

Arguments

`N`	Sample size.
`x`	Vector for the predictor variable.
`m`	Vector for the mediating variable.
`XcorM`	Coefficient of the correlation between the predictor variable and mediating variable.
`McorY`	Coefficient of the correlation between the mediating variable and outcome variable.
`corTotal`	Size of total effect.
`proportionMediated`	The proportion of the total effect that is mediated.
`seed`	Seed for replicability.

Details

Co-created by Robert G. Moulder Jr. and Isaac T. Petersen

Value

- the correlation between the predictor variable (`x`) and the mediating variable (`m`).
- the correlation between the mediating variable (`m`) and the outcome variable (`Y`).
- the correlation between the predictor variable (`x`) and the outcome variable (`Y`).
- the direct correlation between the predictor variable (`x`) and the outcome variable (`Y`), while controlling for the mediating variable (`m`).
- the indirect correlation between the predictor variable (`x`) and the outcome variable (`Y`) through the mediating variable (`m`).
- the total correlation between the predictor variable (`x`) and the outcome variable (`Y`): i.e., the sum of the direct correlation and the indirect correlation.
- the proportion of the correlation between the predictor variable (`x`) and the outcome variable (`Y`) that is mediated through the mediating variable (`m`).

See Also

Other simulation: `complement()`, `simulateAUC()`

Examples

```
#INSERT
```

`specify_decimal` *Specify Decimals.*

Description

Specify the number of decimals to print.

Usage

```
specify_decimal(x, k)
```

Arguments

`x`	Numeric vector.
`k`	Number of decimals to print.

Details

[INSERT].

Value

Character vector of numbers with the specified number of decimal places.

See Also

Other formatting: `apa()`, `pValue()`, `suppressLeadingZero()`

Examples

```
# Prepare Data
v1 <- rnorm(1000)

# Specify Decimals
specify_decimal(v1, 2)
```

`standardErrorIRT` *Standard Error of Measurement (IRT).*

Description

Estimate the standard error of measurement in item response theory.

Usage

```
standardErrorIRT(information)
```

Arguments

`information`	Test information.

Details

Estimate the standard error of measurement in item response theory using the test information (i.e., the sum of all items' information).

Value

Standard error of measurement for that amount of test information.

See Also

doi:10.1177/0146621613475471

Other IRT: `deriv_d_negBinom()`, `discriminationToFactorLoading()`, `fourPL()`, `itemInformation()`, `reliabilityIRT()`, `test_info_4PL()`

Examples

```
# Calculate information for 4 items
item1 <- itemInformation(b = -2, a = 0.6, theta = -4:4)
item2 <- itemInformation(b = -1, a = 1.2, theta = -4:4)
item3 <- itemInformation(b = 1, a = 1.5, theta = -4:4)
item4 <- itemInformation(b = 2, a = 2, theta = -4:4)

items <- data.frame(item1, item2, item3, item4)

# Calculate test information
items$testInformation <- rowSums(items)

# Calculate standard error of measurement
standardErrorIRT(items$testInformation)
```

`suppressLeadingZero` *Suppress Leading Zero.*

Description

Suppress leading zero of numbers.

Usage

```
suppressLeadingZero(value)
```

Arguments

`value`	Numeric vector.

Details

[INSERT].

Value

Character vector of numbers without leading zeros.

See Also

Other formatting: `apa()`, `pValue()`, `specify_decimal()`

Examples

```
# Prepare Data
v1 <- rnorm(1000)

# Suppress Leading Zero
suppressLeadingZero(v1)
```

`test_info_4PL` *Test Information from Logistic IRT Model.*

Description

Estimate test information from logistic item response theory model.

Usage

```
test_info_4PL(
  theta,
  alpha,
  beta,
  gamma = rep(0, length(alpha)),
  delta = rep(1, length(alpha))
)

error_variance_4PL(
  lower = -Inf,
  upper = Inf,
  alpha,
  beta,
  gamma = rep(0, length(alpha)),
  delta = rep(1, length(alpha)),
  mean = 0,
  sd = 1,
  density_cutoff = 1e-10
)
```

```
reliability_4PL(
  alpha,
  beta,
  gamma = rep(0, length(alpha)),
  delta = rep(1, length(alpha))
)
```

Arguments

`theta`	Numeric. The respondent's level on the latent factor/construct.
`alpha`	Numeric. The discrimination parameter of the item, indicating how steeply the item response changes with the person's (`theta`).
`beta`	Numeric. The difficulty parameter of the item, indicating the expected count at a given level on the construct (`theta`).
`gamma`	Numeric. The lower asymptote.
`delta`	Numeric. The upper asymptote.
`lower`	Numeric. The lower range of theta, for estimating error variance or reliability.
`upper`	Numeric. The upper range of theta, for estimating error variance or reliability.
`mean`	Numeric. Mean of normal latent variable.
`sd`	Numeric. Standard deviation of normal latent variable.
`density_cutoff`	Numeric. Cut-off value for very large or very small bounds needed for numerical stability.

Details

Created by Philipp Doebler (doebler@statistik.tu-dortmund.de) and Loreen Sabel (loreen.sabel@tu-dortmund.de).

Value

The amount of information for a given the test as a whole at each of the values of `theta` specified. Based on test information, one can estimate error variance and marginal reliability using `error_variance_4PL()` and `reliability_4PL()`, respectively.

See Also

Other IRT: `deriv_d_negBinom()`, `discriminationToFactorLoading()`, `fourPL()`, `itemInformation()`, `reliabilityIRT()`, `standardErrorIRT()`

Examples

```
test_info_4PL(0,1,0,0,1) # 0.25
test_info_4PL(-0.849, 1.1, -1, 0.2, 0.95) # Magis, 2013, Fig. 2
optimize(function(x)- test_info_4PL(x, 1.1, -1, 0.2, 0.95), c(-3, 3))

# test
```

```
set.seed(23)
# parameters (some are totally unrealistic)
alpha <- runif(20,0.5,2.5)
beta <- runif(20,-2,2)
gamma <- runif(20,0,0.3)
delta <- runif(20,0.8,1)
error_variance_4PL(
  lower = -Inf, upper = Inf,
  alpha, beta, gamma, delta)

error_variance_4PL(
  lower= -Inf, upper= Inf,
  alpha, beta, gamma, delta,
  density_cutoff = 1e-9)

error_variance_4PL(
  lower= -Inf, upper= Inf,
  alpha, beta, gamma, delta,
  density_cutoff = 1e-8)

error_variance_4PL(
  lower = -Inf, upper= Inf,
  alpha, beta, gamma, delta,
  density_cutoff = 1e-7)

reliability_4PL(alpha, beta, gamma, delta)

theta <- seq(-4, 4, length.out = 101)

plot(theta, test_info_4PL(theta, alpha, beta, gamma, delta))
```

`timesPerInterval` *Frequency Per Duration.*

Description

Estimate frequency of a behavior for a particular duration.

Usage

```
timesPerInterval(
  num_occurrences = NULL,
  interval = NULL,
  duration = "month",
  not_occurred_past_year = NULL
)

timesPerLifetime(num_occurrences = NULL, never_occurred = NULL)
```

```
computeItemFrequencies(
  item_names,
  data,
  duration = "month",
  frequency_vars,
  interval_vars,
  not_in_past_year_vars
)

computeLifetimeFrequencies(
  item_names,
  data,
  frequency_vars,
  never_occurred_vars
)
```

Arguments

`num_occurrences`	The number of times the behavior occurred during the specified interval, `interval`.
`interval`	The specified interval corresponding to the number of times the behavior occurred, `num_occurrences`. One of: • `1` = average number of times per day • `2` = average number of times per week • `3` = number of times in the past month • `4` = number of times in the past year
`duration`	The desired duration during which to estimate how many times the behavior occurred: • "day" = average number of times per day • "week" = average number of times per week • "month" = number of times in the past month • "year" = number of times in the past year
`not_occurred_past_year`	Whether or not the behavior did NOT occur in the past year. If `0`, the behavior did occur in the past year. If `1`, the behavior did not occur in the past year.
`never_occurred`	Whether or not the behavior has NEVER occurred in the person's lifetime. If `0`, the behavior has occurred in the person's lifetime. If `1`, the behavior has never occurred in the person's lifetime.
`item_names`	The names of the questionnaire items.
`data`	The data object.
`frequency_vars`	The name(s) of the variables corresponding to the number of occurrences (`num_occurrences`).
`interval_vars`	The name(s) of the variables corresponding to the intervals (`interval`).
`not_in_past_year_vars`	The name(s) of the variables corresponding to whether the behavior did not occur in the past year (`not_occurred_past_year`).

`never_occurred_vars`
: The name(s) of the variables corresponding to whether the behavior has never occurred during the person's lifetime (`never_occurred`).

Details

Estimates the frequency of a given behavior for a particular duration, given a specified number of times it occurred during a specified interval.

Value

The frequency of the behavior for the specified duration.

Examples

```
timesPerInterval(
  num_occurrences = 2,
  interval = 3,
  duration = "month",
  not_occurred_past_year = 0
)

timesPerInterval(
  duration = "month",
  not_occurred_past_year = 1
)

timesPerLifetime(
  num_occurrences = 2,
  never_occurred = 0
)

timesPerLifetime(
  never_occurred = 1
)
```

`update_nested` *Update Nested Models in Hierarchical Regression.*

Description

Wrapper function to ensure the same observations are used for each updated model as were used in the first model.

Usage

```
update_nested(object, formula., ..., evaluate = TRUE)
```

Arguments

`object`	model object to update
`formula.`	updated model formula
`...`	further parameters passed to the fitting function
`evaluate`	whether to evaluate the model. One of: `TRUE` or `FALSE`

Details

Convenience wrapper function to ensure the same observations are used for each updated model as were used in the first model, to ensure comparability of models.

Value

`lm` model

See Also

https://stackoverflow.com/a/37341927

https://stackoverflow.com/a/37416336

https://stackoverflow.com/a/47195348

Other multipleRegression: `lmCombine()`, `plot2WayInteraction()`, `ppPlot()`, `semPlotInteraction()`

Examples

```
# Prepare Data
data("mtcars")

dat <- mtcars

# Create some missing values in mtcars
dat[1, "wt"] <- NA
dat[5, "cyl"] <- NA
dat[7, "hp"] <- NA

m1 <- lm(mpg ~ wt + cyl + hp, data = dat)
m2 <- update_nested(m1, . ~ . - wt)  # Remove wt
m3 <- update_nested(m1, . ~ . - cyl) # Remove cyl
m4 <- update_nested(m1, . ~ . - wt - cyl) # Remove wt and cyl
m5 <- update_nested(m1, . ~ . - wt - cyl - hp) # Remove all three variables
# (i.e., model with intercept only)

anova(m1, m2, m3, m4, m5)
```

varsDifferentTypes *Identify Variables of Different Types.*

Description

Identifies the variables in common across two dataframes that have different types.

Usage

```
varsDifferentTypes(df1, df2)
```

Arguments

df1	dataframe 1 (object)
df2	dataframe 2 (object)

Details

Identifies the variables that have the same name across two dataframes that have different types, which can pose challenges for merging two dataframes.

Value

Dataframe with columns for the variable name, the variable type in `df1` and the variable type in `df2`.

See Also

Other dataManipulation: `columnBindFill()`, `convert.magic()`, `dropColsWithAllNA()`, `dropRowsWithAllNA()`

Examples

```
# Prepare Data
df1 <- data.frame(
  A = 1:3,
  B = 2:4,
  C = 3:5
)

df2 <- data.frame(
  A = as.character(1:3),
  B = 2:4,
  C = as.factor(3:5)
)

# Check if any rows are not NA
varsDifferentTypes(df1, df2)
```

vwReg *Visually Weighted Regression.*

Description

Create watercolor plot to visualize weighted regression.

Usage

```
vwReg(
  formula,
  data,
  title = "",
  B = 1000,
  shade = TRUE,
  shade.alpha = 0.1,
  spag = FALSE,
  spag.color = "darkblue",
  mweight = TRUE,
  show.lm = FALSE,
  show.median = TRUE,
  median.col = "white",
  shape = 21,
  show.CI = FALSE,
  method = loess,
  bw = FALSE,
  slices = 200,
  palette = colorRampPalette(c("#FFEDA0", "#DD0000"), bias = 2)(20),
  ylim = NULL,
  quantize = "continuous",
  add = FALSE,
  ...
)
```

Arguments

`formula`	regression model.
`data`	dataset.
`title`	plot title.
`B`	number of bootstrapped smoothers.
`shade`	whether to plot the shaded confidence region.
`shade.alpha`	whether to fade out the confidence interval shading at the edges (by reducing alpha; 0 = no alpha decrease, 0.1 = medium alpha decrease, 0.5 = strong alpha decrease).
`spag`	whether to plot spaghetti lines.

`spag.color`	the fitting function for the spaghettis; default: `loess`.
`mweight`	logical indicating whether to make the median smoother visually weighted.
`show.lm`	logical indicating whether to plot the linear regression line.
`show.median`	logical indicating whether to plot the median smoother.
`median.col`	color of the median smoother.
`shape`	shape of points.
`show.CI`	logical indicating whether to plot the 95% confidence interval limits.
`method`	color of spaghetti lines.
`bw`	logical indicating whether to use a b&w palette; default: `TRUE`.
`slices`	number of slices in x and y direction for the shaded region. Higher numbers make a smoother plot, but takes longer to draw. I would not set `slices` to more than 500.
`palette`	provide a custom color palette for the watercolors.
`ylim`	restrict range of the watercoloring.
`quantize`	either `continuous`, or `SD`. In the latter case, we get three color regions for 1, 2, and 3 SD (an idea of John Mashey).
`add`	if `add == FALSE`, a new ggplot is returned. If `add == TRUE`, only the elements are returned, which can be added to an existing ggplot (with the + operator).
`...`	further parameters passed to the fitting function, in the case of loess, for example, `span = .9`, or `family = "symmetric"`.

Details

Creates a watercolor plot to visualize weighted regression.

Value

plot

See Also

`https://www.nicebread.de/visually-weighted-regression-in-r-a-la-solomon-hsiang/`

`https://www.nicebread.de/visually-weighted-watercolor-plots-new-variants-please-vote/`

`http://www.fight-entropy.com/2012/07/visually-weighted-regression.html`

`http://www.fight-entropy.com/2012/08/visually-weighted-confidence-intervals.html`

`http://www.fight-entropy.com/2012/08/watercolor-regression.html`

`https://papers.ssrn.com/sol3/papers.cfm?abstract_id=2265501`

Other plot: `addText()`, `plot2WayInteraction()`, `ppPlot()`, `semPlotInteraction()`

Other correlations: `addText()`, `cor.table()`, `crossTimeCorrelation()`, `crossTimeCorrelationDF()`, `partialcor.table()`

Examples

```
# Prepare Data
data("mtcars")
df <- data.frame(x = mtcars$hp, y = mtcars$mpg)

## Visually Weighted Regression

# Default
vwReg(y ~ x, df)

# Shade
vwReg(y ~ x, df, shade = TRUE, show.lm = TRUE, show.CI = TRUE,
quantize = "continuous")
vwReg(y ~ x, df, shade = TRUE, show.lm = TRUE, show.CI = TRUE,
quantize = "SD")

# Spaghetti
vwReg(y ~ x, df, shade = FALSE, spag = TRUE, show.lm = TRUE, show.CI = TRUE)
vwReg(y ~ x, df, shade = FALSE, spag = TRUE)

# Black/white
vwReg(y ~ x, df, shade = TRUE, spag = FALSE, show.lm = TRUE, show.CI = TRUE,
bw = TRUE, quantize = "continuous")
vwReg(y ~ x, df, shade = TRUE, spag = FALSE, show.lm = TRUE, show.CI = TRUE,
bw = TRUE, quantize = "SD")
vwReg(y ~ x, df, shade = FALSE, spag = TRUE, show.lm = TRUE, show.CI = TRUE,
bw = TRUE, quantize = "SD")

# Change the bootstrap smoothing
vwReg(y ~ x, df, family = "symmetric") # use an M-estimator for
# bootstrap smoothers. Usually yields wider confidence intervals
vwReg(y ~ x, df, span = 1.7) # increase the span of the smoothers
vwReg(y ~ x, df, span = 0.5) # decrease the span of the smoothers

# Change the color scheme
vwReg(y ~ x, df, palette = viridisLite::viridis(4)) # viridis
vwReg(y ~ x, df, palette = viridisLite::magma(4)) # magma
vwReg(y ~ x, df, palette = RColorBrewer::brewer.pal(9, "YlGnBu")) # change the
# color scheme, using a predefined ColorBrewer palette. You can see all
# available palettes by using this command:
# `library(RColorBrewer); display.brewer.all()`
vwReg(y ~ x, df, palette = grDevices::colorRampPalette(c("white","yellow",
"green","red"))(20)) # use a custom-made palette
vwReg(y ~ x, df, palette = grDevices::colorRampPalette(c("white","yellow",
"green","red"), bias = 3)(20)) # use a custom-made palette, with the
# parameter bias you can shift the color ramp to the “higher” colors
vwReg(y ~ x, df, bw = TRUE) # black and white version
vwReg(y ~ x, df, shade.alpha = 0, palette = grDevices::colorRampPalette(
c("black","grey30","white"), bias = 4)(20)) # Milky-Way Plot
vwReg(y ~ x, df, shade.alpha = 0, slices = 400, palette =
grDevices::colorRampPalette(c("black","green","yellow","red"),
bias = 5)(20), family = "symmetric") # Northern Light Plot/ fMRI plot
```

```
vwReg(y ~ x, df, quantize = "SD") # 1-2-3-SD plot
```

---

write.aes *Write Encrypted Data.*

Description

Write data to encrypted file.

Usage

```
write.aes(df, filename, key)
```

Arguments

df	Data to encrypt.
filename	Location where to save encrypted data.
key	Encryption key.

Details

Writes data to an encrypted file. To read data from an encrypted file, see read.aes.

Value

A file with encrypted data.

See Also

`https://stackoverflow.com/questions/25318800/how-do-i-read-an-encrypted-file-from-disk-with-r/25321586#25321586`

Other encrypted: `read.aes()`

Examples

```
# Location Where to Save Encryption Key on Local Computer
  #(where only you should have access to it)
#encryptionKeyLocation <- file.path(getwd(), "/encryptionKey.RData",
#  fsep = "") #Can change to a different path, e.g.: "C:/Users/[USERNAME]/"

# Generate a Temporary File Path for Encryption Key
encryptionKeyLocation <- tempfile(fileext = ".RData")

# Generate Encryption Key
key <- as.raw(sample(1:16, 16))
```

```
# Save Encryption Key
save(key, file = encryptionKeyLocation)

# Specify Credentials
credentials <- "Insert My Credentials Here"

# Generate a Temporary File Path for Encrypted Credentials
encryptedCredentialsLocation <- tempfile(fileext = ".txt")

# Save Encrypted Credentials
#write.aes(
#  df = credentials,
#  filename = file.path(getwd(), "/encrypytedCredentials.txt", fsep = ""),
#  key = key) #Change the file location to save this on the lab drive

write.aes(
  df = credentials,
  filename = encryptedCredentialsLocation,
  key = key)

rm(credentials)
rm(key)
```

`%ni%` *NOTIN Operator.*

Description

NOTIN operator.

Usage

```
x %ni% table
```

Arguments

`x`	vector or `NULL`: the values to be matched. Long vectors are supported.
`table`	vector or `NULL`: the values to be matched against. Long vectors are supported.

Details

Determine whether values in one vector are not in another vector.

Value

Vector of `TRUE` and `FALSE`, indicating whether values in one vector are not in another vector.

See Also

https://www.r-bloggers.com/2018/07/the-notin-operator/ https://stackoverflow.com/questions/71309487/r-package-documentation-undocumented-arguments-in-documentation-object-for-a?noredirect=1

Examples

```
# Prepare Data
v1 <- c("Sally","Tom","Barry","Alice")
listToCheckAgainst <- c("Tom","Alice")

v1 %ni% listToCheckAgainst
v1[v1 %ni% listToCheckAgainst]
```

Index

∗ **IRT**
  deriv_d_negBinom, 24
  discriminationToFactorLoading, 28
  fourPL, 32
  itemInformation, 37
  reliabilityIRT, 73
  standardErrorIRT, 84
  test_info_4PL, 86
∗ **accuracy**
  accuracyAtCutoff, 4
  accuracyAtEachCutoff, 6
  accuracyOverall, 9
  nomogrammer, 48
  optimalCutoff, 52
  posttestOdds, 65
∗ **bayesian**
  deriv_d_negBinom, 24
  pA, 54
∗ **behaviorFrequency**
  timesPerInterval, 88
∗ **behaviorIntensity**
  recode_intensity, 71
∗ **computations**
  kish_ess, 38
  meanSum, 44
  Mode, 45
  mySum, 46
∗ **conversion**
  convert.magic, 16
  convertHoursAMPM, 17
  convertToHours, 18
  convertToMinutes, 19
  convertToSeconds, 20
  percentileToTScore, 59
  pom, 63
∗ **correlations**
  addText, 10
  cor.table, 21
  crossTimeCorrelation, 22
  crossTimeCorrelationDF, 23
  partialcor.table, 55
  vwReg, 93
∗ **correlation**
  attenuationCorrelation, 12
  disattenuationCorrelation, 27
∗ **dataEvaluations**
  dropColsWithAllNA, 29
  dropRowsWithAllNA, 30
  is.nan.data.frame, 36
  not_all_na, 50
  not_any_na, 51
∗ **dataManipulation**
  columnBindFill, 14
  convert.magic, 16
  dropColsWithAllNA, 29
  dropRowsWithAllNA, 30
  varsDifferentTypes, 92
∗ **encrypted**
  read.aes, 69
  write.aes, 96
∗ **factor analysis**
  my_loadings_sorter, 47
∗ **fantasyFootball**
  cleanUpNames, 13
∗ **formatting**
  apa, 11
  pValue, 68
  specify_decimal, 84
  suppressLeadingZero, 85
∗ **grants**
  percentEffort, 57
∗ **lab**
  setLabPath, 81
∗ **lavaan**
  make_esem_model, 42
∗ **mixedModel**
  lmeSummary, 40
∗ **multipleImputation**

imputationCombine, 34
imputationModelCompare, 35
imputationPRV, 35
lmCombine, 40
∗ **multipleRegression**
lmCombine, 40
plot2WayInteraction, 60
ppPlot, 66
semPlotInteraction, 78
update_nested, 90
∗ **operators**
%ni%, 97
∗ **packages**
getDependencies, 33
load_or_install, 41
∗ **plot**
addText, 10
plot2WayInteraction, 60
ppPlot, 66
semPlotInteraction, 78
vwReg, 93
∗ **reliability**
reliabilityOfDifferenceScore, 74
repeatability, 75
∗ **simulation**
complement, 15
simulateAUC, 81
simulateIndirectEffect, 82
∗ **structural equation modeling**
equiv_chi, 31
make_esem_model, 42
puc, 67
satorraBentlerScaledChiSquareDifferenceTestStatistic, 77
semPlotInteraction, 78
∗ **times**
convertHoursAMPM, 17
convertToHours, 18
convertToMinutes, 19
convertToSeconds, 20
%ni%, 97

accuracyAtCutoff, 4, *8*, *10*, *49*, *53*, *66*
accuracyAtEachCutoff, *4*, *6*, 6, *10*, *49*, *53*, *66*
accuracyOverall, *6*, *8*, 9, *49*, *53*, *66*
addText, 10, *22–24*, *56*, *61*, *67*, *79*, *94*
apa, 11, *69*, *84*, *86*
attenuationCorrelation, 12, *28*

cleanUpNames, 13
columnBindFill, 14, *16*, *30, 31*, *92*
complement, 15, *82, 83*
computeItemFrequencies (timesPerInterval), 88
computeLifetimeFrequencies (timesPerInterval), 88
convert.magic, *14*, 16, *17–20*, *30, 31*, *59*, *64*, *92*
convertHoursAMPM, *16*, 17, *18–20*, *59*, *64*
convertToHours, *16, 17*, 18, *19, 20*, *59*, *64*
convertToMinutes, *16–18*, 19, *20*, *59*, *64*
convertToSeconds, *16–19*, 20, *59*, *64*
cor.table, *11*, 21, *23, 24*, *56*, *94*
crossTimeCorrelation, *11*, *22*, 22, *24*, *56*, *94*
crossTimeCorrelationDF, *11*, *22, 23*, 23, *56*, *94*

d_negBinom (deriv_d_negBinom), 24
deriv_d_negBinom, 24, *29*, *33*, *38*, *55*, *73*, *85*, *87*
deriv_logd_negBinom (deriv_d_negBinom), 24
disattenuationCorrelation, *13*, 27
discriminationToFactorLoading, *25*, 28, *33*, *38*, *73*, *85*, *87*
dropColsWithAllNA, *14*, *16*, 29, *31*, *37*, *51*, *92*
dropRowsWithAllNA, *14*, *16*, *30*, 30, *37*, *51*, *92*

equiv_chi, 31, *43*, *68*, *78, 79*
error_variance_4PL (test_info_4PL), 86
error_variance_NB (deriv_d_negBinom), 24

fourPL, *25*, *29*, 32, *38*, *73*, *85*, *87*

getDependencies, 33, *42*

imputationCombine, 34, *35, 36*, *40*
imputationModelCompare, *34*, 35, *36*, *40*
imputationPRV, *34, 35*, 35, *40*
info_neg_binom_analytical (deriv_d_negBinom), 24
is.nan.data.frame, *30, 31*, 36, *51*
item_info_NB_analytical (deriv_d_negBinom), 24
item_info_NB_zero_analytical (deriv_d_negBinom), 24
itemInformation, *25*, *29*, *33*, 37, *73*, *85*, *87*

kish_ess, 38, *44, 45*, *47*

lmCombine, *34–36*, 40, *61*, *67*, *79*, *91*
lmeSummary, 40
load_or_install, *34*, 41
log_gen_binom *(*deriv_d_negBinom*)*, 24

make_esem_model, *32*, 42, *68*, *78*, *79*
mark_intensity_as_zero *(*recode_intensity*)*, 71
meanSum, *39*, 44, *45*, *47*
Mode, *39*, *44*, 45, *47*
mortgage, 46
my_loadings_sorter, 47
mySum, *39*, *44*, *45*, 46

nomogrammer, *6*, *8*, *10*, 48, *53*, *66*
not_all_na, *30*, *31*, *37*, 50, *51*
not_any_na, *30*, *31*, *37*, *51*, 51

optimalCutoff, *6*, *8*, *10*, *49*, 52, *66*

pA, *25*, 54
pAgivenB *(*pA*)*, 54
pAgivenNotB *(*pA*)*, 54
partialcor.table, *11*, *21–24*, 55, *94*
pB *(*pA*)*, 54
pBgivenA *(*pA*)*, 54
pBgivenNotA *(*pA*)*, 54
percentEffort, 57
percentileToTScore, *16–20*, 59, *64*
personMonths *(*percentEffort*)*, 57
plot2WayInteraction, *11*, *40*, 60, *67*, *79*, *91*, *94*
pom, *16–20*, *59*, 63
posttestOdds, *6*, *8*, *10*, *49*, *53*, 65
posttestProbability *(*posttestOdds*)*, 65
ppPlot, *11*, *40*, *61*, 66, *79*, *91*, *94*
puc, *32*, *43*, 67, *78*, *79*
pValue, *12*, 68, *84*, *86*

read.aes, 69, *96*
recode_intensity, 71
redcapProgressBar, 72
reliability_4PL *(*test_info_4PL*)*, 86
reliability_NB *(*deriv_d_negBinom*)*, 24
reliabilityIRT, *25*, *29*, *33*, *38*, 73, *85*, *87*
reliabilityOfDifferenceScore, 74, *75*
repeatability, *74*, 75
reverse_score, 76

satorraBentlerScaledChiSquareDifferenceTestStatistic, *32*, *43*, *68*, 77, *79*
semPlotInteraction, *11*, *32*, *40*, *43*, *61*, *67*, *68*, *78*, 78, *91*, *94*
setLabPath, 81
simulateAUC, *15*, 81, *83*
simulateIndirectEffect, *15*, *82*, 82
specify_decimal, *12*, *69*, 84, *86*
standardErrorIRT, *25*, *29*, *33*, *38*, *73*, 84, *87*
suppressLeadingZero, *12*, *69*, *84*, 85

test_info_4PL, *25*, *29*, *33*, *38*, *73*, *85*, 86
test_info_NB *(*deriv_d_negBinom*)*, 24
timesPerInterval, 88
timesPerLifetime *(*timesPerInterval*)*, 88

update_nested, *40*, *61*, *67*, *79*, 90

varsDifferentTypes, *14*, *16*, *30*, *31*, 92
vwReg, *11*, *22–24*, *56*, *61*, *67*, *79*, 93

whdquantile *(*kish_ess*)*, 38
wisdomOfCrowd *(*accuracyOverall*)*, 9
wquantile *(*kish_ess*)*, 38
wquantile_generic *(*kish_ess*)*, 38
write.aes, *69*, *70*, 96
wthdquantile *(*kish_ess*)*, 38