Package ‘ufs’

June 9, 2023

Type Package

Title A Collection of Utilities

Version 0.5.10

Maintainer Gjalt-Jorn Peters <ufs@opens.science>

License GPL (>= 3)

Description This is a new version of the 'userfriendlyscience' package, which has grown a bit unwieldy. Therefore, distinct functionalities are being 'consciously uncoupled' into different packages. This package contains the general-purpose tools and utilities (see the 'behaviorchange' package, the 'rosetta' package, and the soon-to-be-released 'scd' package for other functionality), and is the most direct 'successor' of the original 'userfriendlyscience' package. For example, this package contains a number of basic functions to create higher level plots, such as diamond plots, to easily plot sampling distributions, to generate confidence intervals, to plan study sample sizes for confidence intervals, and to do some basic operations such as (dis)attenuate effect size estimates.

URL https://ufs.opens.science

BugReports https://gitlab.com/r-packages/ufs/-/issues

Encoding UTF-8

LazyData true

RoxygenNote 7.2.3

Depends R (>= 3.0.0)

Suggests bootES (>= 1.2), car (>= 3.0), careless (>= 1.1), GGally (>= 1.4.0), jmvcore (>= 1.2), lavaan (>= 0.6), MASS (>= 7.3), MBESS (>= 4.5.1), psych (>= 1.8), rio (>= 0.5), remotes (>= 0.2), rmarkdown (>= 2.5), rstudioapi (>= 0.11), viridis (>= 0.5.1)

Imports digest (>= 0.6.19), diptest (>= 0.75.7), dplyr (>= 0.7.6), GPArotation, ggplot2 (>= 2.2.1), ggrepel (>= 0.8), ggridges (>= 0.5.0), grDevices (>= 3.0.0), gridExtra (>= 2.3), gtable (>= 0.2.0), htmltools (>= 0.4.0), kableExtra (>= 1.1.0), knitr (>= 1.22), pander (>= 0.6.3), plyr (>= 1.8.4), pwr, rmdpartials (>= 0.5.8), scales (>= 1.0.0), SuppDists (>= 1.1.9)
R topics documented:

aipedjmv ......................................................... 4
aiperjmv ........................................................ 5
areColors .......................................................... 5
arr ................................................................. 6
associationMatrix .................................................. 7
associationsDiamondPlot ......................................... 10
attenuate.d ..................................................... 13
attenuate.r ..................................................... 14
A_VarghaDelaney .................................................. 14
BAC_plot .......................................................... 15
bfi-data ........................................................... 17
biAxisDiamondPlot .................................................. 17
biDimColors ...................................................... 20
carelessObject ................................................... 21
carelessReport ................................................... 22
cat0 ............................................................... 23
checkDataIntegrity ................................................. 24
checkPkgs .......................................................... 26
CIM ................................................................. 27
cohensdCI ........................................................ 29
computeStatistic_t .................................................. 32
confintdjmv ...................................................... 34
confIntOmegaSq ...................................................... 35
confIntProp ....................................................... 36
confIntR .......................................................... 37
confintrjmv ....................................................... 39
confIntSD ........................................................ 39
convert ............................................................. 40
convert.cer.to.d .................................................. 41
convert.d.to.U3 ................................................... 43
convertToNumeric .................................................. 44
cramersV .......................................................... 44
dataShape .......................................................... 45
descr ............................................................... 49
diamondCoordinates ................................................. 52
diamondPlot ....................................................... 55
disattenuate.d .................................................... 57
disattenuate.r .................................................... 58
duoComparisonDiamondPlot .................................... 58

NeedsCompilation no

Author  Gjalt-Jorn Peters [aut, cre] (<https://orcid.org/0000-0002-0336-9589>),
        Stefan Gruijters [ctb] (<https://orcid.org/0000-0003-0141-0071>)

Repository  CRAN

Date/Publication  2023-06-09 16:30:03 UTC
<table>
<thead>
<tr>
<th>R topics documented:</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>escapeRegex</td>
<td>62</td>
</tr>
<tr>
<td>exceptionalScore</td>
<td>63</td>
</tr>
<tr>
<td>exceptionalScores</td>
<td>64</td>
</tr>
<tr>
<td>exportToHTML</td>
<td>66</td>
</tr>
<tr>
<td>extractVarName</td>
<td>66</td>
</tr>
<tr>
<td>faConfInt</td>
<td>67</td>
</tr>
<tr>
<td>factorLoadingDiamondCIplot</td>
<td>68</td>
</tr>
<tr>
<td>factorLoadingHeatmap</td>
<td>70</td>
</tr>
<tr>
<td>fafailsafe</td>
<td>72</td>
</tr>
<tr>
<td>findShortestInterval</td>
<td>72</td>
</tr>
<tr>
<td>formatCI</td>
<td>73</td>
</tr>
<tr>
<td>formatPvalue</td>
<td>74</td>
</tr>
<tr>
<td>formatR</td>
<td>75</td>
</tr>
<tr>
<td>getData</td>
<td>76</td>
</tr>
<tr>
<td>ggBarChart</td>
<td>77</td>
</tr>
<tr>
<td>ggBoxplot</td>
<td>78</td>
</tr>
<tr>
<td>ggEasyBar</td>
<td>79</td>
</tr>
<tr>
<td>ggPie</td>
<td>81</td>
</tr>
<tr>
<td>ggProportionPlot</td>
<td>82</td>
</tr>
<tr>
<td>ggqq</td>
<td>85</td>
</tr>
<tr>
<td>ggSave</td>
<td>87</td>
</tr>
<tr>
<td>heading</td>
<td>88</td>
</tr>
<tr>
<td>ifelseObj</td>
<td>89</td>
</tr>
<tr>
<td>insertFigureCaption</td>
<td>89</td>
</tr>
<tr>
<td>invertItem</td>
<td>91</td>
</tr>
<tr>
<td>iqrOutlier</td>
<td>92</td>
</tr>
<tr>
<td>irpplot</td>
<td>93</td>
</tr>
<tr>
<td>is.nr</td>
<td>94</td>
</tr>
<tr>
<td>is.odd</td>
<td>94</td>
</tr>
<tr>
<td>isTrue</td>
<td>95</td>
</tr>
<tr>
<td>kblXtra</td>
<td>95</td>
</tr>
<tr>
<td>knitAndSave</td>
<td>96</td>
</tr>
<tr>
<td>knitFig</td>
<td>97</td>
</tr>
<tr>
<td>makeScales</td>
<td>99</td>
</tr>
<tr>
<td>massConvertToNumeric</td>
<td>99</td>
</tr>
<tr>
<td>meanConfInt</td>
<td>100</td>
</tr>
<tr>
<td>meansDiamondPlot</td>
<td>101</td>
</tr>
<tr>
<td>meansDiamondPlotjmv</td>
<td>104</td>
</tr>
<tr>
<td>meanSDtoDiamondPlot</td>
<td>104</td>
</tr>
<tr>
<td>multiResponse</td>
<td>106</td>
</tr>
<tr>
<td>multiResponsejmv</td>
<td>108</td>
</tr>
<tr>
<td>multiVarFreq</td>
<td>108</td>
</tr>
<tr>
<td>normalHist</td>
<td>109</td>
</tr>
<tr>
<td>noZero</td>
<td>111</td>
</tr>
<tr>
<td>opts</td>
<td>112</td>
</tr>
<tr>
<td>parallelSubscales</td>
<td>113</td>
</tr>
<tr>
<td>pomegaSq</td>
<td>113</td>
</tr>
<tr>
<td>pwr.bootES</td>
<td>115</td>
</tr>
</tbody>
</table>
aipedjmv

Sample size for accuracy: \( d \)

Description

Sample size for accuracy: \( d \)

Usage

\[
aipedjmv(d = 0.5, w = 0.1, \text{conf.level} = 95)
\]
areColors

Arguments

d
w
conf.level

Value

A results object containing:

| results$text | a html |
| results$aipePlot | an image |

aiperjmv

Sample size for accuracy: r

Description

Sample size for accuracy: r

Usage

aiperjmv(r = 0.3, w = 0.1, conf.level = 95)

Arguments

r
w
conf.level

Value

A results object containing:

| results$text | a html |
| results$aipePlot | an image |

areColors

Check whether elements of a vector are valid colors
**Description**

This function by Josh O’Brien checks whether elements of a vector are valid colors. It has been copied from a Stack Exchange answer (see https://stackoverflow.com/questions/13289009/check-if-character-string-is-a-valid-color-representation).

**Usage**

\[ \text{areColors}(x) \]

**Arguments**

\[ x \] The vector.

**Value**

A logical vector.

**Author(s)**

Josh O’Brien
Maintainer: Gjalt-Jorn Peters gjalt-jorn@userfriendlyscience.com

**Examples**

\[ \text{ufs::areColors(c(NA, "black", "blackk", "1", "#00", "#000000"));} \]

---

**arr**

**Absolute Relative Risk and confidence interval**

**Description**

This is a function to conveniently and quickly compute the absolute relative risk (ARR) and its confidence interval.

**Usage**

\[ \text{arr(} \]
\[ \quad \text{expPos,} \]
\[ \quad \text{expN,} \]
\[ \quad \text{conPos,} \]
\[ \quad \text{conN,} \]
\[ \quad \text{conf.level = 0.95,} \]
\[ \quad \text{digits = 2,} \]
\[ \quad \text{printAsPercentage = TRUE} \]
\[ \quad \text{)} \]

---
## S3 method for class 'ufsARR'
print(x, digits = x$digits, printAsPercentage = x$printAsPercentage, ...)

### Arguments

- `expPos` Number of positive events in the experimental condition.
- `expN` Total number of cases in the experimental condition.
- `conPos` Number of positive events in the control condition.
- `conN` Total number of cases in the control condition.
- `conf.level` The confidence level for the confidence interval.
- `digits` The number of digits to round to when printing the results.
- `printAsPercentage` Whether to multiply with 100 when printing the results.
- `x` The result of the call to `arr`.
- `...` Any additional arguments are neglected.

### Value

An object with in `estimate`, the ARR, and in `conf.int`, the confidence interval.

### Examples

```r
ufs::arr(10, 60, 20, 60);
```

### Description

`associationMatrix` produces a matrix with confidence intervals for effect sizes, point estimates for those effect sizes, and the p-values for the test of the hypothesis that the effect size is zero, corrected for multiple testing.

### Usage

```r
associationMatrix(
  dat = NULL,
  x = NULL,
  y = NULL,
  conf.level = 0.95,
  correction = "fdr",
  bootstrapV = FALSE,
  info = c("full", "ci", "es"),
  includeSampleSize = "depends",
```
bootstrapV.samples = 5000,
digits = 2,
pValueDigits = digits + 1,
colNames = FALSE,
type = c("R", "html", "latex"),
file = "",
statistic = associationMatrixStatDefaults,
effectSize = associationMatrixESDefaults,
var.equal = TRUE
)

## S3 method for class 'associationMatrix'
print(x, type = x$input$type, info = x$input$info, file = x$input$file, ...)

## S3 method for class 'associationMatrix'
pander(x, info = x$input/info, file = x$input/file, ...)

Arguments

- **dat**
  A dataframe with the variables of interest. All variables in this dataframe will be used if both x and y are NULL. If dat is NULL, the user will be presented with a dialog to select a datafile.

- **x**
  If not NULL, this should be a character vector with the names of the variables to include in the rows of the association table. If x is NULL, all variables in the dataframe will be used.

- **y**
  If not NULL, this should be a character vector with the names of the variables to include in the columns of the association table. If y is NULL, the variables in x will be used for the columns as well (which produces a symmetric matrix, similar to most correlation matrices).

- **conf.level**
  Level of confidence of the confidence intervals.

- **correction**
  Correction for multiple testing: an element out of the vector c("holm", "hochberg", "hommel", "bonferroni", "BH", "BY", "fdr", "none"). NOTE: the p-values are corrected for multiple testing; The confidence intervals are not!

- **bootstrapV**
  Whether to use bootstrapping to compute the confidence interval for Cramer’s V or whether to use the Fisher’s Z conversion.

- **info**
  Information to print: either both the confidence interval and the point estimate for the effect size (and the p-value, corrected for multiple testing), or only the confidence intervals, or only the point estimate (and the corrected p-value). Must be an element of the vector c("full", "ci", "es").

- **includeSampleSize**
  Whether to include the sample size when the effect size point estimate and p-value are shown. If this is "depends", it will depend on whether all associations have the same sample size (and the sample size will only be printed when they don’t). If "always", the sample size will always be added. If anything else, it will never be printed.

- **bootstrapV.samples**
  If using bootstrapping for Cramer’s V, the number of samples to generate.
digits
pValueDigits
colNames
type
file
statistic
effectSize
var.equal
...

Value

An object with the input and several output variables, one of which is a dataframe with the association matrix in it. When this object is printed, the association matrix is printed to the screen. If the 'file' parameter is specified, a file with this matrix will also be written to disk.

Note

The 'statistic' and 'effectSize' parameter make it possible to use different functions to conduct null hypothesis significance testing and compute effect sizes. In both cases, the parameter needs to be a list containing four lists, named 'dichotomous', 'nominal', 'ordinal', and 'interval'. Each of these lists has to contain four elements, character vectors of length one (i.e. just one string value), again named 'dichotomous', 'nominal', 'ordinal', and 'interval'.

The combination of each of these names (e.g. 'dichotomous' and 'nominal', or 'ordinal' and 'interval', etc) determine which test should be done when computing the p-value to test the association between two variables of those types, or which effect sizes to compute. When called, associationMatrix determines the measurement levels of the relevant variables. It then uses these two levels (their string representation, e.g. 'dichotomous' etc) to find a string in the 'statistic' and 'effectSize' objects. Two functions with these names are then called from two lists, 'computeStatistic' and computeEffectSize. These lists list contain functions that have the same names as the strings in the 'statistic' list.

For example, when the default settings are used, the string (function name) found for two dichotomous variables when searching in associationMatrixStatDefaults is 'chisq', and the string found in associationMatrixESDefaults is 'v'. associationMatrix then calls computeStatistic[['chisq']] and computeEffectSize[['v']], providing the two variables as arguments, as well as passing the 'conf.level' argument. These two functions then each return an object that associationMatrix
extracts the information from. Inspect the source code of these functions (by typing their names without parentheses in the R prompt) to learn how this object should look, if you want to write your own functions.

Author(s)

Gjalt-Jorn Peters
Maintainer: Gjalt-Jorn Peters gjalt-jorn@userfriendlyscience.com

Examples

```r
### Generate a simple association matrix using all three variables in the
### Orange tree dataframe
associationMatrix(Orange);

### Or four variables from infert:
associationMatrix(infert, c("education", "parity",
                           "induced", "case"), colNames=TRUE);

### Use variable names in the columns and generate html
associationMatrix(Orange, colNames=TRUE, type='html');
```

---

associationsDiamondPlot

*A diamondplot with confidence intervals for associations*

Description

This function produces is a diamondplot that plots the confidence intervals for associations between a number of covariates and a criterion. It currently only supports the Pearson’s r effect size metric; other effect sizes are converted to Pearson’s r.

Usage

```r
associationsDiamondPlot(
    dat,
    covariates,
    criteria,
    labels = NULL,
    criteriaLabels = NULL,
    decreasing = NULL,
    sortBy = NULL,
    conf.level = 0.95,
    criteriaColors = viridisPalette(length(criteria)),
)```
associationsDiamondPlot

```
criterionColor = "black",
returnLayerOnly = FALSE,
esMetric = "r",
multiAlpha = 0.33,
singleAlpha = 1,
showLegend = TRUE,
xlab = "Effect size estimates",
ylab = "",
theme = ggplot2::theme_bw(),
lineSize = 1,
outputFile = NULL,
outputWidth = 10,
outputHeight = 10,
ggsaveParams = ufs::opts$get("ggsaveParams"),
...
)

associationsToDiamondPlotDf(
  dat,
covariates,
criterion,
labels = NULL,
decreasing = NULL,
conf.level = 0.95,
esMetric = "r"
)
```

**Arguments**

- **dat**
  The dataframe containing the relevant variables.

- **covariates**
  The covariates: the list of variables to associate to the criterion or criteria, usually the predictors.

- **criterion, criterion**
  The criteria, usually the dependent variables; one criterion (one dependent variable) can also be specified of course. The helper function associationsToDiamondPlotDf always accepts only one criterion.

- **labels**
  The labels for the covariates, for example the questions that were used (as a character vector).

- **criteriaLabels**
  The labels for the criteria (in the legend).

- **decreasing**
  Whether to sort the covariates by the point estimate of the effect size of their association with the criterion. Use NULL to not sort at all, TRUE to sort in descending order, and FALSE to sort in ascending order.

- **sortBy**
  When specifying multiple criteria, this can be used to indicate by which criterion the items should be sorted (if they should be sorted).

- **conf.level**
  The confidence of the confidence intervals.
criteriaColors, criterionColor

The colors to use for the different associations can be specified in criteriaColors. This should be a vector of valid colors with at least as many elements as criteria are specified in criteria. If only one criterion is specified, the color in criterionColor is used.

returnLayerOnly

Whether to return the entire object that is generated, or just the resulting ggplot2 layer.

esMetric

The effect size metric to plot - currently, only 'r' is supported, and other values will return an error.

multiAlpha, singleAlpha

The transparency (alpha channel) value of the diamonds for each association can be specified in multiAlpha, and if only one criterion is specified, the alpha level of the diamonds can be specified in singleAlpha.

showLegend

Whether to show the legend.

xlab, ylab

The label to use for the x and y axes (for duoComparisonDiamondPlot, must be vectors of two elements). Use NULL to not use a label.

theme

The ggplot() theme to use.

lineSize

The thickness of the lines (the diamonds’ strokes).

outputFile

A file to which to save the plot.

outputWidth, outputHeight

Width and height of saved plot (specified in centimeters by default, see ggsaveParams).

ggsaveParams

Parameters to pass to ggsave when saving the plot.

... Any additional arguments are passed to diamondPlot() and eventually to ggDiamondLayer().

Details

associationsToDiamondPlotDf is a helper function that produces the required dataframe. This function can be used to quickly plot multiple confidence intervals.

Value

A plot.

Author(s)

Gjalt-Jorn Peters

Maintainer: Gjalt-Jorn Peters gjalt-jorn@userfriendlyscience.com

See Also

diamondPlot(), ggDiamondLayer()
Examples

```r
### Simple diamond plot with correlations
### and their confidence intervals
associationsDiamondPlot(mtcars, 
  covariates=c('cyl', 'hp', 'drat', 'wt',
               'am', 'gear', 'vs', 'carb', 'qsec'),
  criteria='mpg');

### Same diamond plot, but now with two criteria,
### and colouring the diamonds based on the
### correlation point estimates: a gradient
### is created where red is used for -1,
### green for 1 and blue for 0.
associationsDiamondPlot(mtcars, 
  covariates=c('cyl', 'hp', 'drat', 'wt',
               'am', 'gear', 'vs', 'carb', 'qsec'),
  criteria=c('mpg', 'disp'),
  generateColors=c('red', 'blue', 'green'),
  fullColorRange=c(-1, 1));
```

### Description

Measurement error (i.e. the complement of reliability) results in a downward bias of observed effect sizes. This attenuation can be emulated by this function.

### Usage

```r
attenuate.d(d, reliability)
```

### Arguments

- **d**: The value of Cohen’s d (that would be obtained with perfect measurements)
- **reliability**: The reliability of the measurements of the continuous variable

### Value

The attenuated value of Cohen’s d

### Author(s)

Gjalt-Jorn Peters & Stefan Gruijters
References


Examples

attenuate.d(.5, .8);

`attenuate.r`  
*Attenuate a Pearson’s r estimate for unreliability in the measurements*

Description

Attenuate a Pearson’s r estimate for unreliability in the measurements

Usage

`attenuate.r(r, reliability1, reliability2)`

Arguments

- `r`: The (disattenuated) value of Pearson’s r
- `reliability1, reliability2`: The reliabilities of the two variables

Value

The attenuated value of Pearson’s r

Examples

`attenuate.r(.5, .8, .9);`

---

A_VarghaDelaney  
*Vargha & Delaney’s A*

Description

Vargha & Delaney’s A
Usage

`A_VarghaDelaney(
  control,
  experimental,
  bootstrap = NULL,
  conf.level = 0.95,
  warn = FALSE
)`

Arguments

- control: A vector with the data for the control condition.
- experimental: A vector with the data from the experimental condition.
- bootstrap: The number of bootstrap samples to use to compute confidence intervals, or NULL to not compute confidence intervals.
- conf.level: The confidence level of the confidence intervals.
- warn: Whether to allow the `stats::wilcox.test()` function to emit warnings, for example if ties are encountered.

Value

A numeric vector of length 1 with the A value, named 'A'.

Examples

```r
ufs::A_VarghaDelaney(1:8, 3:12);
```

---

**BAC_plot**

*Bland-Altman Change plot*

Description

Bland-Altman Change plot

Usage

```r
BAC_plot(
  data,
  cols = names(data),
  reliability = NULL,
  pointSize = 2,
  deterioratedColor = "#482576E6",
  unchangedColor = "#25848E80",
  improvedColor = "#7AD151E6",
  zeroLineColor = "black",
  zeroLineType = "dashed",
)```
ciLineColor = "red",
ciLineType = "solid",
conf.level = 0.95,
theme = ggplot2::theme_minimal(),
ignoreBias = FALSE,
iccFromPsych = FALSE,
iccFromPsychArgs = NULL
)

Arguments

- **data**: The data frame; if it only has two columns, the first of which is the pre-change column, `cols` can be left empty.
- **cols**: The names of the columns with the data; the first is the column with the pre-change data, the second the column after the change.
- **reliability**: The reliability estimate, for example as obtained with the `ICC()` function in the `psych()` package; can be omitted, in which case the intraclass correlation is computed.
- **pointSize**: The size of the points in the plot.
- **deterioratedColor**, **unchangedColor**, **improvedColor**: The colors to use for cases who deteriorate, stay the same, and improve, respectively.
- **zeroLineColor**, **ciLineColor**: The colors for the line at 0 (no change) and at the confidence interval bounds (i.e. the point at which a difference becomes indicative of change given the reliability), respectively.
- **zeroLineType**, **ciLineType**: The line types for the line at 0 (no change) and at the confidence interval bounds (i.e. the point at which a difference becomes indicative of change given the reliability), respectively.
- **conf.level**: The confidence level of the confidence interval.
- **theme**: The ggplot2 theme to use.
- **ignoreBias**: Whether to ignore bias (i.e. allow the measurements at the second time to shift upwards or downwards). If FALSE, the variance associated with such a shift is considered error variance (i.e. 'unreliability').
- **iccFromPsych**: Whether to compute ICC using the `psych::ICC()` function or not.
- **iccFromPsychArgs**: If using the `psych::ICC()` function, the arguments to pass.

Value

A ggplot2 plot.
bfi-data

Examples

```r
### Create smaller dataset for example
dat <-
ufs::testRetestSimData[
  1:25,
  c('t0_item1', 't1_item1')
];

ufs::BAC_plot(dat, reliability = .5);
ufs::BAC_plot(dat, reliability = .8);
ufs::BAC_plot(dat, reliability = .9);
```

---

bfi-data 25 Personality items representing 5 factors

---

Description

This is a dataset lifted from the psychTools package (which was originally in the psych package). For details, please check that help page (using "psychTools::bfi").

Usage

data(bfi)

Format

A data.frame with 2800 rows and 28 columns.

Examples

data(bfi);

---

biAxisDiamondPlot Diamondplot with two Y axes

---

Description

This is basically a meansDiamondPlot(), but extended to allow specifying subquestions and anchors at the left and right side. This is convenient for psychological questionnaires when the anchors or dimensions were different from item to item. This function is used to function the left panel of the CIBER plot in the behaviorchange package.
Usage

biAxisDiamondPlot(
  dat,
  items = NULL,
  leftAnchors = NULL,
  rightAnchors = NULL,
  subQuestions = NULL,
  decreasing = NULL,
  conf.level = 0.95,
  showData = TRUE,
  dataAlpha = 0.1,
  dataColor = "#444444",
  diamondColors = NULL,
  jitterWidth = 0.45,
  jitterHeight = 0.45,
  xbreaks = NULL,
  xLabels = NA,
  xAxisLab = paste0("Scores and ", round(100 * conf.level, 2), ", % CIs"),
  drawPlot = TRUE,
  returnPlotOnly = TRUE,
  baseSize = 1,
  dotSize = baseSize,
  baseFontSize = 10 * baseSize,
  theme = ggplot2::theme_bw(base_size = baseFontSize),
  outputFile = NULL,
  outputWidth = 10,
  outputHeight = 10,
  ggsaveParams = ufs::opts$get("ggsaveParams"),
  ...
)

Arguments

dat The dataframe containing the variables.
items The variables to include.
leftAnchors The anchors to display on the left side of the left hand panel. If the items were measured with one variable each, this can be used to show the anchors that were used for the respective scales. Must have the same length as items.
rightAnchors The anchors to display on the left side of the left hand panel. If the items were measured with one variable each, this can be used to show the anchors that were used for the respective scales. Must have the same length as items.
subQuestions The subquestions used to measure each item. This can also be used to provide pretty names for the variables if the items were not measured by one question each. Must have the same length as items.
decreasing Whether to sort the items. Specify NULL to not sort at all, TRUE to sort in descending order, and FALSE to sort in ascending order.
conf.level The confidence levels for the confidence intervals.
showData  Whether to show the individual datapoints.
dataAlpha  The alpha level (transparency) of the individual datapoints. Value between 0 and 1, where 0 signifies complete transparency (i.e. invisibility) and 1 signifies complete 'opaqueness'.
dataColor  The color to use for the individual datapoints.
diamondColors  The colours to use for the diamonds. If NULL, the generateColors argument can be used which will then be passed to diamondPlot().
jitterWidth  How much to jitter the individual datapoints horizontally.
jitterHeight  How much to jitter the individual datapoints vertically.
xbreaks  Which breaks to use on the X axis (can be useful to override ggplot()'s defaults).
xLabels  Which labels to use for those breaks (can be useful to override ggplot()'s defaults; especially useful in combination with xBreaks of course).
xAxisLab  Axis label for the X axis.
drawPlot  Whether to draw the plot, or only return it.
returnPlotOnly  Whether to return the entire object that is generated (including all intermediate objects) or only the plot.
baseSize  This can be used to efficiently change the size of most plot elements.
dotSize  This is the size of the points used to show the individual data points in the left hand plot.
baseFontSize  This can be used to set the font size separately from the baseSize.
theme  This is the theme that is used for the plots.
outputFile  A file to which to save the plot.
outputWidth, outputHeight  Width and height of saved plot (specified in centimeters by default, see ggsaveParams).
ggsaveParams  Parameters to pass to ggsave when saving the plot.
...  These arguments are passed on to diamondPlot).

Details

This is a diamondplot that can be used for items/questions where the anchors of the response scales could be different for every item. For the rest, it is very similar to meansDiamondPlot().

Value

Either just a plot (a gtable::gtable() object) or an object with all produced objects and that plot.

Author(s)

Gjalt-Jorn Peters
Maintainer: Gjalt-Jorn Peters gjalt-jorn@userfriendlyscience.com

See Also

CIBER() in the behaviorchange package, associationsDiamondPlot()
Examples

```r
biAxisDiamondPlot(dat=mtcars,
  items=c('cyl', 'wt'),
  subQuestions=c('cylinders', 'weight'),
  leftAnchors=c('few', 'light'),
  rightAnchors=c('many', 'heavy'),
  xbreaks=0:8);
```

---

**biDimColors**

*Create colours for a response scale for an item*

**Description**

Create colours for a response scale for an item

**Usage**

```r
biDimColors(start, mid, end, length, show = TRUE)
uniDimColors(start, end, length, show = TRUE)
```

**Arguments**

- `start`: Color to start with
- `mid`: Color in the middle, for bidimensional scales
- `end`: Color to end with
- `length`: The number of response options
- `show`: Whether to show the colours

**Value**

The colours as hex codes.

**Examples**

```r
uniDimColors("#000000", "#00BB00", length=5, show=FALSE);
```
Description

This function is a wrapper for the functions from the careless package. Normally, you’d probably call carelessReport which calls this function to generate a report of suspect participants.

Usage

carelessObject(
  data,
  items = names(data),
  flagUnivar = 0.99,
  flagMultivar = 0.95,
  irvSplit = 4,
  responseTime = NULL
)

Arguments

data The dataframe.
items The items to look at.
flagUnivar How extreme a score has to be for it to be flagged as suspicious univariately.
flagMultivar This has not been implemented yet.
irvSplit Whether to split for the IRV, and if so, in how many parts.
responseTime If not NULL, the name of a column containing the participants’ response times.

Value

An object of class carelessObject.

Examples

carelessObject(mtcars);
carelessReport  

A report to help diagnosing careless responders

**Description**

This function wraps functions from the careless package to help inspect and diagnose careless participants. It is optimized for using in R Markdown files.

**Usage**

```r

carelessReport(
  data,
  items = names(data),
  nFlags = 1,
  flagUnivar = 0.99,
  flagMultivar = 0.95,
  irvSplit = 4,
  headingLevel = 3,
  datasetName = NULL,
  responseTime = NULL,
  headingSuffix = " {.tabset}",
  digits = 2,
  missingSymbol = "Missing"
)
```

**Arguments**

- **data**
  The dataframe.
- **items**
  The items to look at.
- **nFlags**
  How many indicators need to be flagged for a participant to be considered suspect.
- **flagUnivar**
  How extreme a score has to be for it to be flagged as suspicious univariately.
- **flagMultivar**
  This has not been implemented yet.
- **irvSplit**
  Whether to split for the IRV, and if so, in how many parts.
- **headingLevel**
  The level of the heading in Markdown (the number of #s to include before the heading).
- **datasetName**
  The name of the dataset to display (to override, if desired).
- **responseTime**
  If not NULL, the name of a column containing the participants’ response times.
- **headingSuffix**
  The suffix to include; by default, set such that the individual participants IRP plots are placed in separate tabs.
- **digits**
  The number of digits to round to.
- **missingSymbol**
  How to represent missing values.
Value

NULL, invisibly; and prints the report.

Examples

```r
### Get the BFI data taken from the 'psych' package
dat <- ufs::bfi;

### Get the variable names for the regular items
bfiVars <- setdiff(names(dat), c("gender", "education", "age"));

### Inspect suspect participants, very conservatively to
### limit the output (these are 2800 participants).
carelessReport(data = dat,
items = bfiVars,
nFlags = 5);
```

---

**cat0**  
*Concatenate to screen without spaces*

Description

The `cat0` function is to `cat` what `paste0` is to `paste`; it simply makes concatenating many strings without a separator easier.

Usage

```r
cat0(..., sep = "")
```

Arguments

- `...` The character vector(s) to print; passed to `cat`.
- `sep` The separator to pass to `cat`, of course, "" by default.

Value

Nothing (invisible NULL, like `cat`).

Examples

```r
cat0("The first variable is ", names(mtcars)[1], ",");
```
checkDataIntegrity  Conveniently checking data integrity

Description
This function is designed to make it easy to perform some data integrity checks, specifically checking for values that are impossible or unrealistic. These values can then be replaced by another value, or the offending cases can be deleted from the dataframe.

Usage
checkDataIntegrity(
  x,
  dat,
  newValue = NA,
  removeCases = FALSE,
  validValueSuffix = "_validValue",
  newValueSuffix = "_newValue",
  totalVarName = "numberOfInvalidValues",
  append = TRUE,
  replace = TRUE,
  silent = FALSE,
  rmarkdownOutput = FALSE,
  callingSelf = FALSE
)

Arguments

x  This can be either a vector or a list. If it is a vector, it should have two elements, the first one being a regular expression matching one or more variables in the dataframe specified in dat, and second one being the condition the matching variables have to satisfy. If it is a list, it should be a list of such vectors. The conditions should start with a Comparison operator followed by a value (e.g. "<30" or ">=0).

dat  The dataframe containing the variables of which we should check the integrity.

newValue  The new value to be assigned to cases not satisfying the specified conditions.

removeCases  Whether to delete cases that do not satisfy the criterion from the dataframe (if FALSE, they’re not deleted, but the offending value is replaced by newValue).

validValueSuffix  Suffix to append to variable names when creating variable names for new variables that contain TRUE and FALSE to specify for each original variable whether its value satisfied the specified criterion.

newValueSuffix  If replace is FALSE, original values are not replaced, but instead new variables are created where the offending values have been replaced. This suffix is appended to each original variable name to create the new variable name.
checkDataIntegrity

totalVarName This is the name of a variable that contains, for each case, the total number of invalid values among all variables checked.

append Whether to append the columns to the dataframe, or only return the new columns.

replace Whether to replace the offending values with the value specified in newValue or whether to create new columns (see newValueSuffix).

silent Whether to display the log, or only set it as attribute of the returned dataframe.

rmarkdownOutput Whether to format the log so that it’s ready to be included in RMarkdown reports.

callingSelf For internal use; whether the function calls itself.

Value

The dataframe with the corrections, and the log stored in attribute checkDataIntegrity_log.

Author(s)

Gjalt-Jorn Peters

Maintainer: Gjalt-Jorn Peters gjalt-jorn@userfriendlyscience.com

Examples

### Default behavior: return dataframe with offending values replaced by NA

checkDataIntegrity(c('mpg', '<30'), mtcars);

### Check two conditions, and instead of returning the dataframe with the results appended, only return the columns indicating which cases 'pass', what the new values would be, and how many invalid values were found for each case (to easily remove cases that provided many invalid values)

checkDataIntegrity(list(c('mpg', '<30'),
                         c('gear', '<5')),
                     mtcars,
                     append=FALSE);
checkPkgs

Check for presence of a package

Description

This function efficiently checks for the presence of a package without loading it (unlike `library()` or `require()`). This is useful to force yourself to use the package::function syntax for addressing functions; you can make sure required packages are installed, but their namespace won’t attach to the search path.

Usage

```r
checkPkgs(
  ..., 
  install = FALSE,
  load = FALSE,
  repos = "https://cran.rstudio.com"
)
```

Arguments

- `...`: A series of packages. If the packages are named, the names are the package names, and the values are the minimum required package versions (see the second example).
- `install`: Whether to install missing packages from `repos`.
- `load`: Whether to load packages (which is exactly not the point of this package, but hey, YMMV).
- `repos`: The repository to use if installing packages; default is the RStudio repository.

Value

Invisibly, a vector of the available packages.

Examples

```r
ufs::checkPkgs('base');

### Require a specific version
ufs::checkPkgs(ufs = "0.3.1");

### This will show the error message
tryCatch(
  ufs::checkPkgs(
    base = "99",
    stats = "42.5",
    ufs = 20
  )
)"
## Conceptual Independence Matrix

### Description

Conceptual Independence Matrix

### Usage

```r
CIM(
  data,
  scales,
  conf.level = 0.95,
  colors = c("#440154FF", "#7AD151FF"),
  outputFile = NULL,
  outputWidth = 100,
  outputHeight = 100,
  outputUnits = "cm",
  faMethod = "minres",
  n.iter = 100,
  n.repeatOnWarning = 50,
  warningTolerance = 2,
  showWarnings = FALSE,
  skipRegex = NULL,
  headingLevel = 2,
  printAbbreviations = TRUE,
  drawPlot = TRUE,
  returnPlotOnly = TRUE
)
```

```r
CIM_partial(
  x,
  headingLevel = x$input$headingLevel,
  quiet = TRUE,
  echoPartial = FALSE,
  partialFile = NULL,
  ...
)
```

```r
## S3 method for class 'CIM'
knit_print(
```
Arguments

data
scales
conf.level
colors
outputFile
outputWidth, outputHeight, outputUnits
faMethod
n.iter
n.repeatOnWarning
warningTolerance
silentRepeatOnWarning
showWarnings
skipRegex
headingLevel
printAbbreviations
drawPlot
returnPlotOnly
x
quiet
echoPartial
partialFile
...
### Load dataset `bfi`, originally from psychTools package

```r
data(bfi, package='ufs');
```

### Specify scales

```r
bfiScales <-
  list(Agreeableness = paste0("Agreeableness_item_", 1:5),
       Conscientiousness = paste0("Conscientiousness_item_", 1:5),
       Extraversion = paste0("Extraversion_item_", 1:5),
       Neuroticism = paste0("Neuroticism_item_", 1:5),
       Openness = paste0("Openness_item_", 1:5));
```

```r
names(bfi) <- c(unlist(bfiScales),
               c('gender', 'education', 'age'));
```

### Only select first two and the first three items to keep it quick; just pass the full 'bfiScales'

```r
CIM(bfi, scales=clapply(bfiScales, head, 3)[1:2],
    n.iter=10);
```

---

**cohensdCI**

The distribution of Cohen's $d$

---

**Description**

These functions use some conversion to and from the $t$ distribution to provide the Cohen's $d$ distribution. There are four versions that act similar to the standard distribution functions (the `d`, `p`, `q`, and `r` functions, and their longer aliases `.Cohensd`), three convenience functions (`pdExtreme`, `pdMild`, and `pdInterval`), a function to compute the confidence interval for a Cohen's $d$ estimate `cohensdCI`, and a function to compute the sample size required to obtain a confidence interval around a Cohen's $d$ estimate with a specified accuracy (`pwr.cohensdCI` and its alias `pwr.confIntd`).

**Usage**

```r
cohensdCI(d, n, conf.level = 0.95, plot = FALSE, silent = TRUE)
```

```r
dCohensd(
  x,
  df = NULL,
)```
cohensdCI

populationD = 0,
n = NULL,
n1 = NULL,
n2 = NULL,
silent = FALSE
)

pCohensd(q, df, populationD = 0, lower.tail = TRUE)

qCohensd(p, df, populationD = 0, lower.tail = TRUE)

rCohensd(n, df, populationD = 0)

pdInterval(ds, n, populationD = 0)

pdExtreme(d, n, populationD = 0)

pdMild(d, n, populationD = 0)

pwr.cohensdCI(d, w = 0.1, conf.level = 0.95, extensive = FALSE, silent = TRUE)

Arguments

n, n1, n2 Desired number of Cohen’s d values for rCohensd and rd (n), and the number of participants/datapoints in total (n) or in each group (n1 and n2) for dd, dCohensd, pdExtreme, pdMild, pdInterval, and cohensdCI.

conf.level The level of confidence of the confidence interval.

plot Whether to show a plot of the sampling distribution of Cohen’s d and the confidence interval. This can only be used if specifying one value for d, n, and conf.level.

silent Whether to provide FALSE or suppress (TRUE) warnings. This is useful because function 'qt', which is used under the hood (see qt() for more information), warns that ‘full precision may not have been achieved’ when the density of the distribution is very close to zero. This is normally no cause for concern, because with sample sizes this big, small deviations have little impact.

x, q, d Vector of quantiles, or, in other words, the value(s) of Cohen’s d.

df Degrees of freedom.

populationD The value of Cohen’s d in the population; this determines the center of the Cohen’s d distribution. I suppose this is the noncentrality parameter.

lower.tail logical; if TRUE (default), probabilities are the likelihood of finding a Cohen’s d smaller than the specified value; otherwise, the likelihood of finding a Cohen’s d larger than the specified value.

p Vector of probabilities (p-values).

ds A vector with two Cohen’s d values.

w The desired maximum ’half-width’ or margin of error of the confidence interval.

extensive Whether to only return the required sample size, or more extensive results.
The functions use `convert.d.to.t()` and `convert.t.to.d()` to provide the Cohen’s $d$ distribution.

The confidence interval functions, `cohensdCI` and `pwr.cohensdCI`, now use the same method as MBESS (a slightly adapted version of the MBESS function `conf.limits.nct` is used).

More details about `cohensdCI` and `pwr.cohensdCI` are provided in Peters & Crutzen (2017).

### Value

dCohensd (or dd) gives the density, pCohensd (or pd) gives the distribution function, qCohensd (or qd) gives the quantile function, and rCohensd (or rd) generates random deviates.

pdExtreme returns the probability (or probabilities) of finding a Cohen’s $d$ equal to or more extreme than the specified value(s).

pdMild returns the probability (or probabilities) of finding a Cohen’s $d$ equal to or less extreme than the specified value(s).

pdInterval returns the probability of finding a Cohen’s $d$ that lies in between the two specified values of Cohen’s $d$.

cohensdCI provides the confidence interval(s) for a given Cohen’s $d$ value.

pwr.cohensdCI provides the sample size required to obtain a confidence interval for Cohen’s $d$ with a desired width.

### Author(s)

Gjalt-Jorn Peters (Open University of the Netherlands), with the exported MBESS function `conf.limits.nct` written by Ken Kelley (University of Notre Dame), and with an error noticed by Guy Prochilo (University of Melbourne).

Maintainer: Gjalt-Jorn Peters gjalt-jorn@userfriendlyscience.com

### References

Peters, G. J. Y. & Crutzen, R. (2017) Knowing exactly how effective an intervention, treatment, or manipulation is and ensuring that a study replicates: accuracy in parameter estimation as a partial solution to the replication crisis. https://dx.doi.org/


### See Also

`convert.d.to.t()`, `convert.t.to.d()`, `dt()`, `pt()`, `qt()`, `rt()`
Examples

```r
### Confidence interval for Cohen's d of .5
### from a sample of 200 participants, also
### showing this visually: this clearly shows
### how wildly our Cohen's d value can vary
### from sample to sample.
cohensdCI(.5, n=200, plot=TRUE);

### How many participants would we need if we
### would want a more accurate estimate, say
### with a maximum confidence interval width
### of .2?
pwr.cohensdCI(.5, w=.1);

### Show that 'sampling distribution':
cohensdCI(.5,
  n=pwr.cohensdCI(.5, w=.1),
  plot=TRUE);

### Generate 10 random Cohen's d values
rCohensd(10, 20, populationD = .5);

### Probability of findings a Cohen's d smaller than
### .5 if it's 0 in the population (i.e. under the
### null hypothesis)
pCohensd(.5, 64);

### Probability of findings a Cohen's d larger than
### .5 if it's 0 in the population (i.e. under the
### null hypothesis)
1 - pCohensd(.5, 64);

### Probability of findings a Cohen's d more extreme
### than .5 if it's 0 in the population (i.e. under
### the null hypothesis)
pdExtreme(.5, 64);

### Probability of findings a Cohen's d more extreme
### than .5 if it's 0.2 in the population.
pdExtreme(.5, 64, populationD = .2);
```

computeStatistic_t  associationMatrix Helper Functions

Description

These objects contain a number of settings and functions for associationMatrix.
Usage

computeStatistic_t(var1, var2, conf.level = 0.95, var.equal = TRUE, ...)
computeStatistic_r(var1, var2, conf.level = 0.95, ...)
computeStatistic_f(var1, var2, conf.level = 0.95, ...)
computeStatistic_chisq(var1, var2, conf.level = 0.95, ...)
computeEffectSize_d(var1, var2, conf.level = 0.95, var.equal = TRUE, ...)
computeEffectSize_r(var1, var2, conf.level = 0.95, ...)
computeEffectSize_etasq(var1, var2, conf.level = 0.95, ...)
computeEffectSize_omegasq(var1, var2, conf.level = 0.95, ...)
computeEffectSize_v(
  var1,
  var2,
  conf.level = 0.95,
  bootstrap = FALSE,
  samples = 5000,
  ...
)

Arguments

- **var1**: One of the two variables for which to compute a statistic or effect size
- **var2**: The other variable for which to compute the statistic or effect size
- **conf.level**: The confidence for the confidence interval for the effect size
- **var.equal**: Whether to test for equal variances (test), assume equality (yes), or assume inequality (no).
- **...**: Any additional arguments are sometimes used to specify exactly how statistics and effect sizes should be computed.
- **bootstrap**: Whether to bootstrap to estimate the confidence interval for Cramer’s V. If FALSE, the Fisher’s Z conversion is used.
- **samples**: If bootstrapping, the number of samples to generate (of course, more samples means more accuracy and longer processing time).

Value

associationMatrixStatDefaults and associationMatrixESDefaults contain the default functions from computeStatistic and computeEffectSize that are called (see the help file for associationMatrix for more details).

The other functions return an object with the relevant statistic or effect size, with a confidence interval for the effect size.
For computeStatistic, this object always contains:

statistic       The relevant statistic
statistic.type  The type of statistic
parameter       The degrees of freedom for this statistic
p.raw           The p-value of this statistic for NHST

And in addition, it often contains (among other things, sometimes):

object          The object from which the statistics are extracted

For computeEffectSize, this object always contains:

es               The point estimate for the effect size
esc.type         The type of effect size
.ci              The confidence interval for the effect size

And in addition, it often contains (among other things, sometimes):

object          The object from which the effect size is extracted

Author(s)

Gjalt-Jorn Peters
Maintainer: Gjalt-Jorn Peters gjalt-jorn@userfriendlyscience.com

Examples

computeStatistic_f(Orange$Tree, Orange$circumference)
computeEffectSize_etasq(Orange$Tree, Orange$circumference)

confintdjmv()   Effect Size Confidence Interval: Cohens's d

Description

Effect Size Confidence Interval: Cohens's d

Usage

confintdjmv(d = 0.5, n = 128, conf.level = 95)

Arguments

  d
  n
  conf.level
Value

A results object containing:

- `results$text` a html
- `results$ciPlot` an image

Description

This function uses the MBESS functions `conf.limits.ncf()` (which has been copied into this package to avoid the dependency on MBESS) and `convert.ncf.to.omegasq()` to compute the point estimate and confidence interval for Omega Squared (which have been lifted out of MBESS to avoid importing the whole package).

Usage

```r
confIntOmegaSq(var1, var2, conf.level = 0.95)
```

```r
# S3 method for class 'confIntOmegaSq'
print(x, ..., digits = 2)
```

Arguments

- `var1`, `var2` The two variables: one should be a factor (or will be made a factor), the other should have at least interval level of measurement. If none of the variables is a factor, the function will look for the variable with the least unique values and change it into a factor.
- `conf.level` Level of confidence for the confidence interval.
- `x`, `digits`, ... Respectively the object to print, the number of digits to round to, and any additional arguments to pass on to the `print` function.

Value

A `confIntOmegaSq` object is returned, with as elements:

- `input` The input arguments
- `intermediate` Objects generated while computing the output
- `output` The output of the function, consisting of:
  - `output$es` The point estimate
  - `output$ci` The confidence interval
Note

Formula 16 in Steiger (2004) is used for the conversion in `convert.ncf.to.omegasq()`.

Author(s)

Gjalt-Jorn Peters
Maintainer: Gjalt-Jorn Peters gjalt-jorn@userfriendlyscience.com

References


Examples

```r
confIntOmegaSq(mtcars$mpg, mtcars$cyl);
```

---

**confIntProp**

Confidence intervals for proportions, vectorized over all arguments

Description

This function simply computes confidence intervals for proportions.

Usage

```r
confIntProp(x, n, conf.level = 0.95, plot = FALSE)
```

Arguments

- `x` The number of 'successes', i.e. the number of events, observations, or cases that one is interested in.
- `n` The total number of cases or observations.
- `conf.level` The confidence level.
- `plot` Whether to plot the confidence interval in the binomial distribution.

Details

This function is the adapted source code of `binom.test()`. It uses `pbeta()`, with some lines of code taken from the `binom.test()` source. Specifically, the count for the low category is specified as first 'shape argument' to `pbeta()`, and the total count (either the sum of the count for the low category and the count for the high category, or the total number of cases if `compareHiToLo` is `FALSE`) minus the count for the low category as the second 'shape argument'.
confIntR

Value

The confidence interval bounds in a twodimensional matrix, with the first column containing the lower bound and the second column containing the upper bound.

Author(s)

Unknown (see `binom.test()`; adapted by Gjalt-Jorn Peters)

Maintainer: Gjalt-Jorn Peters gjalt-jorn@userfriendlyscience.com

See Also

`binom.test()` and `ggProportionPlot`, the function for which this was written.

Examples

### Simple case

```r
confIntProp(84, 200);
```

### Using vectors

```r
confIntProp(c(2,3), c(10, 20), conf.level=c(.90, .95, .99));
```

---

confIntR

*A function to compute a correlation’s confidence interval*

Description

This function computes the confidence interval for a given correlation and its sample size. This is useful to obtain confidence intervals for correlations reported in papers when informing power analyses.

Usage

```r
confIntR(r, N, conf.level = 0.95, plot = FALSE)
```

Arguments

- **r**: The observed correlation coefficient.
- **N**: The sample size of the sample where the correlation was computed.
- **conf.level**: The desired confidence level of the confidence interval.
- **plot**: Whether to show a plot.
The confidence interval(s) in a matrix with two columns. The left column contains the lower bound, the right column the upper bound. The `rownames()` are the observed correlations, and the `colnames()` are 'lo' and 'hi'. The confidence level and sample size are stored as attributes. The results are returned like this to make it easy to access single correlation coefficients from the resulting object (see the examples).

Author(s)

Douglas Bonett (UC Santa Cruz, United States), with minor edits by Murray Moinester (Tel Aviv University, Israel) and Gjalt-Jorn Peters (Open University of the Netherlands, the Netherlands).

Maintainer: Gjalt-Jorn Peters gjalt-jorn@userfriendlyscience.com

References


Peters, G. J. Y. & Crutzen, R. (forthcoming) An easy and foolproof method for establishing how effective an intervention or behavior change method is: required sample size for accurate parameter estimation in health psychology.

See Also

`confIntR()`

Examples

```r
### To request confidence intervals for one correlation
confIntR(.3, 100);

### The lower bound of a single correlation
confIntR(.3, 100)[1];

### To request confidence intervals for multiple correlations:
confIntR(c(.1, .3, .5), 250);

### The upper bound of the correlation of .5:
confIntR(c(.1, .3, .5), 250)['0.5', 'hi'];
```
### confIntSD

**Description**

Confidence interval for standard deviation

**Usage**

```r
confIntSD(x, n = NULL, conf.level = 0.95)
```

**Arguments**

- `x` Either a standard deviation, in which case `n` must also be provided, or a vector, in which case `n` must be NULL.
- `n` The sample size is `x` is a standard deviation.
- `conf.level` The confidence level

---

### confintrjmv

**Description**

Effect Size Confidence Interval: Pearson’s r

**Usage**

```r
cointrjmv(r = 0.3, N = 400, conf.level = 95)
```

**Arguments**

- `r` .
- `N` .
- `conf.level` .

**Value**

A results object containing:

- `results$text` a html
- `results$ciPlot` an image

---
Value

A vector or matrix.

Examples

ufs::confIntSD(mtcars$mpg);
ufs::confIntSD(c(6, 7), c(32, 32));

convert

Description

These are a number of functions to convert statistics and effect size measures from/to each other.

Arguments

- chisq, cohensf, cohensfsq, d, etasq, f, logodds, means, omegasq, or, p, r, t, z
  The value of the relevant statistic or effect size.
- ncf
  The value of a noncentrality parameter of the F distribution.
- n, n1, n2, N, ns
  The number of observations that the r or t value is based on, or the number of observations in each of the two groups for an anova, or the total number of participants when specifying a noncentrality parameter.
- df, df1, df2
  The degrees of freedom for that statistic (for F, the first one is the numerator (i.e. the effect), and the second one the denominator (i.e. the error term)).
- proportion
  The proportion of participants in each of the two groups in a t-test or anova. This is used to compute the sample size in each group if the group sizes are unknown. Thus, if you only provide df1 and df2 when converting an F value to a Cohen's d value, equal group sizes are assumed.
- b
  The value of a regression coefficient.
- se, sds
  The standard error of standard errors of the relevant statistic (e.g. of a regression coefficient) or variables.
- minDim
  The smallest of the number of columns and the number of rows of the crosstable for which the chisquare is translated to a Cramer's V value.
- lower.tail
  For the F and chisquare distributions, whether to get the probability of the lower or upper tail.
- akfEq8
  When converting Cohen's d to r, for small sample sizes, bias is introduced when the commonly suggested formula is used (Aaron, Kromrey & Ferron, 1998). Therefore, by default, this function uses different equations depending on the sample size (for n < 50 and for n > 50). When akfEq8 is set to TRUE or FALSE, the corresponding action is taken; when akfEq8 is not logical (i.e. TRUE or FALSE), the function depends on the sample size.
- var.equal
  Whether to compute the value of t or Cohen's d assuming equal variances ('yes'), unequal variances ('no'), or whether to test for the difference ('test').
convert.cer.to.d

Details

Note that by default, the behavior of convert.d.to.r depends on the sample size (see Bruce, Kromrey & Ferron, 1998).

Value

The converted value as a numeric value.

Author(s)

Gjalt-Jorn Peters and Peter Verboon

Maintainer: Gjalt-Jorn Peters gjalt-jorn@userfriendlyscience.com

References


Examples

convert.t.to.r(t=-6.46, n=200);
convert.r.to.t(r=-.41, n=200);

### Compute some p-values
convert.t.to.p(4.2, 197);
convert.chisq.to.p(5.2, 3);
convert.f.to.p(8.93, 3, 644);

### Convert d to r using both equations
convert.d.to.r(d=.2, n1=5, n2=5, akfEq8 = FALSE);
convert.d.to.r(d=.2, n1=5, n2=5, akfEq8 = TRUE);

convert.cer.to.d

Helper functions for Numbers Needed for Change

Description

These functions are used by nnc() in the behaviorchange package to compute the Numbers Needed for Change, but are also available for manual use.
Usage

convert.cer.to.d(
    cer,
    eer,
    eventDesirable = TRUE,
    eventIfHigher = TRUE,
    dist = "norm",
    distArgs = NULL,
    distNS = "stats"
)

convert.d.to.eer(
    d,
    cer,
    eventDesirable = TRUE,
    eventIfHigher = TRUE,
    dist = "norm",
    distArgs = list(),
    distNS = "stats"
)

convert.d.to.nnc(d, cer, r = 1, eventDesirable = TRUE, eventIfHigher = TRUE)

convert.eer.to.d(
    eer,
    cer,
    eventDesirable = TRUE,
    eventIfHigher = TRUE,
    dist = "norm",
    distArgs = NULL,
    distNS = "stats"
)

Arguments

cer The Control Event Rate.
eer The Experimental Event Rate.
eventDesirable Whether an event is desirable or undesirable.
eventIfHigher Whether scores above or below the threshold are considered 'an event'.
dist, distArgs, distNS Used to specify the distribution to use to convert between Cohen’s d and the CER and EER. distArgs can be used to specify additional arguments to the corresponding q and p functions, and distNS to specify the namespace (i.e. package) from where to get the distribution functions.
d The value of Cohen’s d.
r The correlation between the determinant and behavior (for mediated Numbers Needed for Change).
convert.d.to.U3

Value

The converted value.

Author(s)

Gjalt-Jorn Peters & Stefan Gruijters
Maintainer: Gjalt-Jorn Peters gjalt-jorn@userfriendlyscience.com

References


See Also

nnc() in the behaviorchange package.

Examples

convert.d.to.eer(d=.5, cer=.25);
convert.d.to.nnc(d=.5, cer=.25);

convert.d.to.U3 (Convert Cohen’s d to U3)

Description

This function simply returns the result of pnorm() for Cohen’s d.

Usage

convert.d.to.U3(d)

Arguments

d        Cohen’s d.

Value

An unnames numeric vector with the U3 values.

Examples

convert.d.to.U3(.5);
convertToNumeric  Conveniently convert vectors to numeric

Description
Tries to `smartly` convert factor and character vectors to numeric.

Usage
convertToNumeric(vector, byFactorLabel = FALSE)

Arguments
- vector: The vector to convert.
- byFactorLabel: When converting factors, whether to do this by their label value (TRUE) or their level value (FALSE).

Value
The converted vector.

Examples
ufs::convertToNumeric(as.character(1:8));

cramersV  Cramer's V and its confidence interval

Description
These functions compute the point estimate and confidence interval for Cramer's V.

Usage
cramersV(x, y = NULL, digits = 2)

## S3 method for class 'CramersV'
print(x, digits = x$input$digits, ...)

confIntV(
  x,
  y = NULL,
  conf.level = 0.95,
  samples = 500,
  digits = 2,
  method = c("bootstrap", "fisher"),...
storeBootstrappingData = FALSE

## S3 method for class 'confIntV'
print(x, digits = x$input$digits, ...)

**Arguments**

- **x**: Either a crosstable to analyse, or one of two vectors to use to generate that crosstable. The vector should be a factor, i.e. a categorical variable identified as such by the 'factor' class.
- **y**: If `x` is a crosstable, `y` can (and should) be empty. If `x` is a vector, `y` must also be a vector.
- **digits**: Minimum number of digits after the decimal point to show in the result.
- **...**: Any additional arguments are passed on to the `print` function.
- **conf.level**: Level of confidence for the confidence interval.
- **samples**: Number of samples to generate when bootstrapping.
- **method**: Whether to use Fisher’s Z or bootstrapping to compute the confidence interval.
- **storeBootstrappingData**: Whether to store (or discard) the data generating during the bootstrapping procedure.

**Value**

A point estimate or a confidence interval for Cramer’s V, an effect size to describe the association between two categorical variables.

**Examples**

```r
### Get confidence interval for Cramer’s V
### Note that by using 'table', and so removing the raw data, inhibits
### bootstrapping, which could otherwise take a while.
confIntV(table(infert$education, infert$induced));
```

---

**Description**

`normalityAssessment` can be used to assess whether a variable and the sampling distribution of its mean have an approximately normal distribution.
Usage

dataShape(
  sampleVector,
  na.rm = TRUE,
  type = 2,
  digits = 2,
  conf.level = 0.95,
  plots = TRUE,
  xLabs = NA,
  yLabs = NA,
  qqCI = TRUE,
  labelOutliers = TRUE,
  sampleSizeOverride = NULL
)

## S3 method for class 'dataShape'
print(x, digits = x$input$digits, extraNotification = TRUE, ...)

## S3 method for class 'dataShape'
pander(x, digits = x$input$digits, extraNotification = TRUE, ...)

normalityAssessment(
  sampleVector,
  samples = 10000,
  digits = 2,
  samplingDistColor = "#2222CC",
  normalColor = "#00CC00",
  samplingDistLineSize = 2,
  normalLineSize = 1,
  xLabel.sampleDist = NULL,
  yLabel.sampleDist = NULL,
  xLabel.samplingDist = NULL,
  yLabel.samplingDist = NULL,
  sampleSizeOverride = TRUE
)

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

## S3 method for class 'normalityAssessment'
pander(x, headerPrefix = "#####", suppressPlot = FALSE, ...)

samplingDistribution(
  popValues = c(0, 1),
  popFrequencies = c(50, 50),
  sampleSize = NULL,
  sampleFromPop = FALSE,
  ...
Arguments

- **sampleVector**: Numeric vector containing the sample data.
- **na.rm**: Whether to remove missing data first.
- **type**: Type of skewness and kurtosis to compute; either 1 (g1 and g2), 2 (G1 and G2), or 3 (b1 and b2). See Joanes & Gill (1998) for more information.
- **digits**: Number of digits to use when printing results.
- **conf.level**: Confidence of confidence intervals.
- **plots**: Whether to display plots.
- **xLabs, yLabs**: The axis labels for the three plots (should be vectors of three elements; the first specifies the X or Y axis label for the rightmost plot (the histogram), the second for the middle plot (the QQ plot), and the third for the rightmost plot (the box plot).
- **qqCI**: Whether to show the confidence interval for the QQ plot.
- **labelOutliers**: Whether to label outliers with their row number in the box plot.
- **sampleSizeOverride**: Whether to use the sample size of the sample as sample size for the sampling distribution, instead of the sampling distribution size. This makes sense, because otherwise, the sample size and thus sensitivity of the null hypothesis significance tests is a function of the number of samples used to generate the sampling distribution.
- **x**: The object to print/pander.
- **extraNotification**: Whether to be particularly informative.
- **samples**: Number of samples to use when constructing sampling distribution.
- **samplingDistColor**: Color to use when drawing the sampling distribution.
- **normalColor**: Color to use when drawing the standard normal curve.
- **samplingDistLineSize**: Size of the line used to draw the sampling distribution.
- **normalLineSize**: Size of the line used to draw the standard normal distribution.
- **xLabel.sampleDist**: Label of x axis of the distribution of the sample.
- **yLabel.sampleDist**: Label of y axis of the distribution of the sample.
- **xLabel.samplingDist**: Label of x axis of the sampling distribution.
- **yLabel.samplingDist**: Label of y axis of the sampling distribution.
- **headerPrefix**: A prefix to insert before the heading (e.g. to use Markdown headings).
suppressPlotWhether to suppress (TRUE) or print (FALSE) the plot.

popValuesThe possible values (levels) of the relevant variable. For example, for a dichotomous variable, this can be "c(1:2)" (or "c(1, 2)"). Note that samplingDistribution is for manually specifying the frequency distribution (or proportions); if you have a vector with 'raw' data, just call normalityAssessment directly.

popFrequenciesThe frequencies corresponding to each value in popValues; must be in the same order! See the examples.

sampleSizeSize of the sample; the sum of the frequencies if not specified.

sampleFromPopIf true, the sample vector is created by sampling from the population information specified; if false, rep() is used to generate the sample vector. Note that is proportions are supplied in popFrequencies, sampling from the population is necessary!

Details

samplingDistribution is a convenient wrapper for normalityAssessment that makes it easy to quickly generate a sample and sampling distribution from frequencies (or proportions).

dataShape computes the skewness and kurtosis.

normalityAssessment provides a number of normality tests and draws histograms of the sample data and the sampling distribution of the mean (most statistical tests assume the latter is normal, rather than the first; normality of the sample data guarantees normality of the sampling distribution of the mean, but if the sample size is sufficiently large, the sampling distribution of the mean is approximately normal even when the sample data are not normally distributed). Note that for the sampling distribution, the degrees of freedom are usually so huge that the normality tests, negligible deviations from normality will already result in very small p-values.

samplingDistribution makes it easy to quickly assess the distribution of a variables based on frequencies or proportions, and dataShape computes skewness and kurtosis.

Value

An object with several results, the most notably of which are:

plot.sampleDistHistogram of sample distribution

sw.sampleDistShapiro-Wilk normality test of sample distribution

ad.sampleDistAnderson-Darling normality test of sample distribution

ks.sampleDistKolmogorov-Smirnov normality test of sample distribution

kurtosis.sampleDistKurtosis for sample distribution

skewness.sampleDistSkewness for sample distribution

plot.samplingDistHistogram of sampling distribution

sw.samplingDistShapiro-Wilk normality test of sampling distribution
ad.samplingDist
Anderson-Darling normality test of sampling distribution

ks.samplingDist
Kolmogorov-Smirnov normality test of sampling distribution
dataShape.samplingDist
Skewness and kurtosis for sampling distribution

Examples

### Note: the 'not run' is simply because running takes a lot of time,
### but these examples are all safe to run!
## Not run:

normalityAssessment(rnorm(35));

### Create a distribution of three possible values and
### show the sampling distribution for the mean
popValues <- c(1, 2, 3);
popFrequencies <- c(20, 50, 30);
sampleSize <- 100;
samplingDistribution(popValues = popValues,
    popFrequencies = popFrequencies,
    sampleSize = sampleSize);

### Create a very skewed distribution of ten possible values
popValues <- c(1, 2, 3, 4, 5, 6, 7, 8, 9, 10);
popFrequencies <- c(2, 4, 8, 6, 10, 15, 12, 200, 350, 400);
samplingDistribution(popValues = popValues,
    popFrequencies = popFrequencies,
    sampleSize = sampleSize, digits=5);

## End(Not run)

descr

### descr (or descriptives)

Description

This function provides a number of descriptives about your data, similar to what SPSS’s DESCRIPTIVES (often called with DESC) does.

Usage

descr(
  x,
  digits = 4,
  errorOnFactor = FALSE,
include = c("central tendency", "spread", "range", "distribution shape", "sample size"),
maxModes = 1,
t = FALSE,
conf.level = 0.95,
quantileType = 2
)

## Default S3 method:
descr(
  x,
  digits = 4,
  errorOnFactor = FALSE,
  include = c("central tendency", "spread", "range", "distribution shape", "sample size"),
  maxModes = 1,
  t = FALSE,
  conf.level = 0.95,
  quantileType = 2
)

## S3 method for class 'descr'
print(
  x,
  digits = attr(x, "digits"),
  t = attr(x, "transpose"),
  row.names = FALSE,
  ...
)

## S3 method for class 'descr'
pander(x, headerPrefix = "", headerStyle = "**", ...)

## S3 method for class 'descr'
as.data.frame(x, row.names = NULL, optional = FALSE, ...)

## S3 method for class 'data.frame'
descr(x, ...)

Arguments

x  The vector for which to return descriptives.
digits The number of digits to round the results to when showing them.
errorOnFactor Whether to show an error when the vector is a factor, or just show the frequencies instead.
include Which elements to include when showing the results.
maxModes Maximum number of modes to display: displays "multi" if more than this number of modes if found.
t Whether to transpose the dataframes when printing them to the screen (this is easier for users relying on screen readers).
conf.level  Confidence of confidence interval around the mean in the central tendency measures.
quantileType  The type of quantiles to be used to compute the interquartile range (IQR). See `quantile` for more information.
row.names  Whether to show row names (TRUE) or not (FALSE).
...  Additional arguments are passed to the default print and pander methods.
headerPrefix  The prefix for the heading; can be used to insert hashes (#) to create Markdown headings.
headerStyle  A string to insert before and after the heading (to make stuff bold or italic in Markdown).
optional  Provided for compatibility with the default `as.data.frame()` method - see that help page for details.

Details

Note that R (of course) has many similar functions, such as `summary`, `psych::describe()` in the excellent `psych::psych` package.

The Hartigans' Dip Test may be unfamiliar to users; it is a measure of uni- vs. multidimensionality, computed by `diptest::dip.test()` from the `dip.test` package. Depending on the sample size, values over .025 can be seen as mildly indicative of multimodality, while values over .05 probably warrant closer inspection (the p-value can be obtained using `diptest::dip.test()`; also see Table 1 of Hartigan & Hartigan (1985) for an indication as to critical values).

Value

A list of dataframes with the requested values.

Author(s)

Gjalt-Jorn Peters
Maintainer: Gjalt-Jorn Peters gjalt-jorn@userfriendlyscience.com

References


See Also

`summary`, `psych::describe()`

Examples

descr(mtcars$mpg);
Description

These functions are used by `diamondPlot()` to construct a diamond plot. It’s normally not necessary to call this function directly: instead, use `meansDiamondPlot()`, `meanSDtoDiamondPlot()`, and `factorLoadingDiamondCIplot()`.

Usage

```r
diamondCoordinates(
  values,
  otherAxisValue = 1,
  direction = "horizontal",
  autoSize = NULL,
  fixedSize = 0.15
)

ggDiamondLayer(
  data,
  ciCols = 1:3,
  colorCol = NULL,
  generateColors = NULL,
  fullColorRange = NULL,
  color = "black",
  lineColor = NA,
  otherAxisCol = 1:nrow(data),
  autoSize = NULL,
  fixedSize = 0.15,
  direction = "horizontal",
  ...
)

rawDataDiamondLayer(
  dat,
  items = NULL,
  itemOrder = 1:length(items),
  dataAlpha = 0.1,
  dataColor = "#444444",
  jitterWidth = 0.5,
  jitterHeight = 0.4,
  size = 3,
  ...
)

varsToDiamondPlotDf(
```
diamondCoordinates

dat,
items = NULL,
labels = NULL,
decreasing = NULL,
conf.level = 0.95
)

Arguments

values A vector of 2 or more values that are used to construct the diamond coordinates. If three values are provided, the middle one becomes the diamond’s center. If two, four, or more values are provided, the median becomes the diamond’s center.

otherAxisValue The value on the other axis to use to compute the coordinates; this will be the Y axis value of the points of the diamond (if direction is ‘horizontal’) or the X axis value (if direction is ‘vertical’).

direction Whether the diamonds should be constructed horizontally or vertically.

autoSize Whether to make the height of each diamond conditional upon its length (the width of the confidence interval).

data, dat A dataframe (or matrix) containing lower bounds, centers (e.g. means), and upper bounds of intervals (e.g. confidence intervals) for ggDiamondLayer or items and raw data for varsToDiamondPlotDf and rawDataDiamondLayer.

ciCols The columns in the dataframe with the lower bounds, centers (e.g. means), and upper bounds (in that order).

colorCol The column in the dataframe containing the colors for each diamond, or a vector with colors (with as many elements as the dataframe has rows).

generateColors A vector with colors to use to generate a gradient. These colors must be valid arguments to colorRamp() (and therefore, to col2rgb()).

fullColorRange When specifying a gradient using generateColors, it is usually desirable to specify the minimum and maximum possible value corresponding to the outer anchors of that gradient. For example, when plotting numbers from 0 to 100 using a gradient from ‘red’ through ‘orange’ to ‘green’, none of the means may actually be 0 or 100; the lowest mean may be, for example, 50. If no fullColorRange is specified, the diamond representing that lowest mean of 50 will be red, not orange. When specifying the fullColorRange, the lowest and highest ‘colors’ in generateColors are anchored to the minimum and maximum values of fullColorRange.

color When no colors are automatically generated, all diamonds will have this color.

lineColor If NA, lines will have the same colors as the diamonds’ fill. If not NA, must be a valid color, which is then used as line color. Note that e.g. linetype and color can be used as well, which will be passed on to geom_polygon().

otherAxisCol A vector of values, or the index of the column in the dataframe, that specifies the values for the Y axis of the diamonds. This should normally just be a vector of consecutive integers.
... Any additional arguments are passed to `geom_polygon()`. This can be used to set, for example, the alpha value of the diamonds. Additional arguments for `rawDataDiamondLayer` are passed on to `geom_jitter()`.

- `items` The items from the dataframe to include in the diamondplot or dataframe.
- `itemOrder` Order of the items to use (if not sorting).
- `dataAlpha` This determines the alpha (transparency) of the data points.
- `dataColor` The color of the data points.
- `jitterWidth` How much to jitter the individual datapoints horizontally.
- `jitterHeight` How much to jitter the individual datapoints vertically.
- `size` The size of the data points.
- `labels` The item labels to add to the dataframe.
- `decreasing` Whether to sort the items (rows) in the dataframe decreasing (TRUE), increasing (FALSE), or not at all (NULL).
- `conf.level` The confidence of the confidence intervals.

**Value**

- `ggDiamondLayer` returns a `ggplot()` `geom_polygon()` object, which can then be used in `ggplot()` plots (as `diamondPlot()` does).
- `diamondCoordinates` returns a set of four coordinates that together specify a diamond.
- `varsToDiamondPlotDf` returns a dataframe of `diamondCoordinates`.
- `rawDataDiamondLayer` returns a `geom_jitter()` object.

**Author(s)**

Gjalt-Jorn Peters

Maintainer: Gjalt-Jorn Peters gjalt-jorn@userfriendlyscience.com

**See Also**

- `meansDiamondPlot()`, `meanSDtoDiamondPlot()`,
- `factorLoadingDiamondCIplot()`,
- `diamondPlot()`

**Examples**

```r
## Not run:
### (Don't run this example as a test, because we
### need the ggplot function which isn't part of
### this package.)

### The coordinates for a simple diamond
diamondCoordinates(values = c(1,2,3));

### Plot this diamond
ggplot() + ggDiamondLayer(data.frame(1,2,3));

## End(Not run)
```
diamondPlot

Basic diamond plot construction function

diamondPlot

Description

This function constructs a diamond plot using ggDiamondLayer(). It’s normally not necessary to call this function directly: instead, use meansDiamondPlot() meanSDtoDiamondPlot(), and factorLoadingDiamondCIplot().

Usage

diamondPlot(
  data,
  ciCols = 1:3,
  colorCol = NULL,
  otherAxisCol = NULL,
  yValues = NULL,
  yLabels = NULL,
  ylab = NULL,
  autoSize = NULL,
  fixedSize = 0.15,
  xlab = "Effect Size Estimate",
  theme = ggplot2::theme_bw(),
  color = "black",
  returnLayerOnly = FALSE,
  outputFile = NULL,
  outputWidth = 10,
  outputHeight = 10,
  ggsaveParams = ufs::opts$get("ggsaveParams"),
  ...
)

Arguments

data A dataframe (or matrix) containing lower bounds, centers (e.g. means), and upper bounds of intervals (e.g. confidence intervals).
ciCols The columns in the dataframe with the lower bounds, centers (e.g. means), and upper bounds (in that order).
colorCol The column in the dataframe containing the colors for each diamond, or a vector with colors (with as many elements as the dataframe has rows).
otherAxisCol The column in the dataframe containing the values that determine where on the Y axis the diamond should be placed. If this is not available in the dataframe, specify it manually using yValues.
yValues The values that determine where on the Y axis the diamond should be placed (can also be a column in the dataframe; in that case, use otherAxisCol).
The labels to use for for each diamond (placed on the Y axis).
Whether to make the height of each diamond conditional upon its length (the width of the confidence interval).
If not using relative heights, fixedSize determines the height to use.
The labels of the X and Y axes.
The theme to use.
Color to use if colors are specified for each diamond.
Set this to TRUE to only return the `ggplot()` layer of the diamondplot, which can be useful to include it in other plots.
A file to which to save the plot.
Width and height of saved plot (specified in centimeters by default, see `ggsaveParams`).
Parameters to pass to `ggsave` when saving the plot.
Additional arguments will be passed to `ggDiamondLayer()`.
A `ggplot2::ggplot()` plot with a `ggDiamondLayer()` is returned.
Gjalt-Jorn Peters
Maintainer: Gjalt-Jorn Peters gjalt-jorn@userfriendlyscience.com
`meansDiamondPlot()`, `meanSDtoDiamondPlot()`, `ggDiamondLayer()`, `factorLoadingDiamondCIplot()`
```r
tmpDf <- data.frame(lo = c(1, 2, 3),
mean = c(1.5, 3, 5),
hi = c(2, 4, 10),
color = c('green', 'red', 'blue'));

### A simple diamond plot
diamondPlot(tmpDf);

### A diamond plot using the specified colours
diamondPlot(tmpDf, colorCol = 4);

### A diamond plot using automatically generated colours
### using a gradient
diamondPlot(tmpDf, generateColors=c('green', 'red'));

### A diamond plot using automatically generated colours
```
### Description

Measurement error (i.e. the complement of reliability) results in a downward bias of observed effect sizes. This attenuation can be reversed by disattenuation.

### Usage

```r
disattenuate.d(d, reliability)
```

### Arguments

- **d**: The (attenuated) value of Cohen’s d (i.e. the value as observed in the sample, and therefore attenuated (decreased) by measurement error in the continuous variable).
- **reliability**: The reliability of the measurements of the continuous variable

### Value

The disattenuated value of Cohen’s d

### Author(s)

Gjalt-Jorn Peters & Stefan Gruijters

### References


### Examples

```r
disattenuate.d(.5, .8);
```
disattenuate.r  Disattenuate a Pearson’s r estimate for unreliability

Description

Disattenuate a Pearson’s r estimate for unreliability

Usage

\texttt{disattenuate.r(r, reliability1, reliability2)}

Arguments

- \textit{r}  
  The (attenuated) value of Pearson’s r

- \textit{reliability1, reliability2}  
  The reliabilities of the two variables

Value

The disattenuated value of Pearson’s r

Examples

\texttt{disattenuate.r(.5, .8, .9);}

duoComparisonDiamondPlot  meansComparisonDiamondPlot and duoComparisonDiamondPlot

Description

These are two diamond plot functions to conveniently make diamond plots to compare subgroups or different samples. They are both based on a univariate diamond plot where colors are used to distinguish the data points and diamonds of each subgroup or sample. The means comparison diamond plot produces only this plot, while the duo comparison diamond plot combines it with a diamond plot visualising the effect sizes of the associations. The latter currently only works for two subgroups or samples, while the simple meansComparisonDiamondPlot also works when comparing more than two sets of datapoints. These functions are explained more in detail in Peters (2017).
Usage

duoComparisonDiamondPlot(
    dat,
    items = NULL,
    compareBy = NULL,
    labels = NULL,
    compareByLabels = NULL,
    decreasing = NULL,
    conf.level = c(0.95, 0.95),
    showData = TRUE,
    dataAlpha = 0.1,
    dataSize = 3,
    comparisonColors = viridisPalette(length(unique(dat[, compareBy]))),
    associationsColor = "grey",
    alpha = 0.33,
    jitterWidth = 0.5,
    jitterHeight = 0.4,
    xlab = c("Scores and means", "Effect size estimates"),
    ylab = c(NULL, NULL),
    plotTitle = NULL,
    theme = ggplot2::theme_bw(),
    showLegend = TRUE,
    legend.position = "top",
    lineSize = 1,
    drawPlot = TRUE,
    xbreaks = "auto",
    outputFile = NULL,
    outputWidth = 10,
    outputHeight = 10,
    ggsaveParams = ufs::opts$get("ggsaveParams"),
    ...
)

meansComparisonDiamondPlot(
    dat,
    items = NULL,
    compareBy = NULL,
    labels = NULL,
    compareByLabels = NULL,
    decreasing = NULL,
    sortBy = NULL,
    conf.level = 0.95,
    showData = TRUE,
    dataAlpha = 0.1,
    dataSize = 3,
    comparisonColors = viridisPalette(length(unique(dat[, compareBy]))),
    alpha = 0.33,
    jitterWidth = 0.5,
jitterHeight = 0.4,
xlab = "Scores and means",
ylab = NULL,
plotTitle = NULL,
theme = ggplot2::theme_bw(),
showLegend = TRUE,
legend.position = "top",
lineSize = 1,
xbreaks = "auto",
outputFile = NULL,
outputWidth = 10,
outputHeight = 10,
ggsaveParams = ufs::opts$get("ggsaveParams"),
...)

Arguments

dat
The dataframe containing the relevant variables.

items
The variables to plot (on the y axis).

compareBy
The variable by which to compare (i.e. the variable indicating to which subgroup
or sample a row in the dataframe belongs).

labels
The labels to use on the y axis; these values will replace the variable names in
the dataframe (specified in items).

compareByLabels
The labels to use to replace the value labels of the compareBy variable.

decreasing
Whether to sort the variables by their mean values (NULL to not sort, TRUE to sort in descending order (i.e. items with lower means are plotted more to the bottom), and FALSE to sort in ascending order (i.e. items with lower means are plotted more to the top).

conf.level
The confidence level of the confidence intervals specified by the diamonds for the means (for meansComparisonDiamondPlot) and for both the means and effect sizes (for duoComparisonDiamondPlot).

showData
Whether to plot the data points.

dataAlpha
The transparency (alpha channel) value for the data points: a value between 0
and 1, where 0 denotes complete transparency and 1 denotes complete opacity.

dataSize
The size of the data points.

comparisonColors
The colors to use for the different subgroups or samples. This should be a vector
of valid colors with at least as many elements as sets of data points that should be plotted.

associationsColor
For duoComparisonDiamondPlot, the color to use to plot the effect sizes in the
right-hand plot.

alpha
The alpha channel (transparency) value for the diamonds: a value between 0 and
1, where 0 denotes complete transparency and 1 denotes complete opacity.
**duoComparisonDiamondPlot**

**jitterWidth, jitterHeight**
How much noise to add to the data points (to prevent overplotting) in the horizontal (x axis) and vertical (y axis) directions.

**xlab, ylab**
The label to use for the x and y axes (for duoComparisonDiamondPlot, must be vectors of two elements). Use NULL to not use a label.

**plotTitle**
Optionally, for meansComparisonDiamondPlot, a title for the plot (can also be specified for duoComparisonDiamondPlot, in which case it’s passed on to meansComparisonDiamondPlot for the left panel - but note that this messes up the alignment of the two panels).

**theme**
The theme to use for the plots.

**showLegend**
Whether to show the legend (which color represents which subgroup/sample).

**legend.position**
Where to place the legend in meansComparisonDiamondPlot (can also be specified for duoComparisonDiamondPlot, in which case it’s passed on to meansComparisonDiamondPlot for the left panel - but note that this messes up the alignment of the two panels).

**lineSize**
The thickness of the lines (the diamonds’ strokes).

**drawPlot**
Whether to draw the plot, or only (invisibly) return it.

**xbreaks**
Where the breaks (major grid lines, ticks, and labels) on the x axis should be.

**outputFile**
A file to which to save the plot.

**outputWidth, outputHeight**
Width and height of saved plot (specified in centimeters by default, see ggsaveParams).

**ggsaveParams**
Parameters to pass to ggsave when saving the plot.

**...**
Any additional arguments are passed to diamondPlot() by meansComparisonDiamondPlot and to both meansComparisonDiamondPlot and associationsDiamondPlot() by duoComparisonDiamondPlot.

**sortBy**
If the variables should be sorted (see decreasing), this variable specified which subgroup should be sorted by. Therefore, the value specified here must be a value label (‘level label’) of the compareBy variable.

**Details**
These functions are explained in Peters (2017).

**Value**
A Diamond plots: a ggplot2::ggplot() plot meansComparisonDiamondPlot, and a gtable() by duoComparisonDiamondPlot.

**Author(s)**
Gjalt-Jorn Peters
Maintainer: Gjalt-Jorn Peters gjalt-jorn@userfriendlyscience.com
References

See Also
diamondPlot(), meansDiamondPlot(), the CIBER() function in the behaviorchange package

Examples
meansComparisonDiamondPlot(mtcars, items=c('disp', 'hp'), compareBy='vs', xbreaks=c(100,200, 300, 400));
meansComparisonDiamondPlot(chickwts, items='weight', compareBy='feed', xbreaks=c(100,200,300,400), showData=FALSE);
duoComparisonDiamondPlot(mtcars, items=c('disp', 'hp'), compareBy='vs', xbreaks=c(100,200,300,400));

---

escapeRegex
Escapes any characters that would have special meaning in a regular expression.

Description
Escapes any characters that would have special meaning in a regular expression.

Usage
escapeRegex(string)

Arguments
string string being operated on.

Details
escapeRegex will escape any characters that would have special meaning in a regular expression. For any string grep(regexpEscape(string), string) will always be true.
exceptionalScore

Value

The value of the string with any characters that would have special meaning in a regular expression escaped.

Note

Note that this function was copied literally from the Hmisc package (to prevent importing the entire package for one line of code).

Author(s)

Charles Dupont
Department of Biostatistics
Vanderbilt University
Maintainer: Gjalt-Jorn Peters gjalt-jorn@userfriendlyscience.com

See Also


Examples

```r
string <- "this\(system\) \{is\} [full]."
escapeRegex(string)
```

```
exceptionalScore      Find exceptional scores

Description

This function can be used to detect exceptionally high or low scores in a vector.

Usage

```r
exceptionalScore(
  x,
  prob = 0.025,
  both = TRUE,
  silent = FALSE,
  quantileCorrection = 1e-04,
  quantileType = 8
)
```
exceptionalScores

Arguments

- **x**: Vector in which to detect exceptional scores.
- **prob**: Probability that a score is exceptionally positive or negative; i.e. scores with a quartile lower than `prob` or higher than `1-prob` are considered exceptional (if both is TRUE, at least). So, note that a prob of .025 means that if both=TRUE, the most exceptional 5% of the values is marked as such.
- **both**: Whether to consider values exceptional if they’re below `prob` as well as above `1-prob`, or whether to only consider values exceptional if they’re below `prob` is `prob` is < .5, or above `prob` if `prob` > .5.
- **silent**: Can be used to suppress messages.
- **quantileCorrection**: By how much to correct the computed quantiles; this is used because when a distribution is very right-skewed, the lowest quantile is the lowest value, which is then also the mode; without subtracting a correction, almost all values would be marked as ‘exceptional’.
- **quantileType**: The algorithm used to compute the quantiles; see `stats::quantile()`.

Details

Note that of course, by definition, `prob` or `2 * prob` percent of the values is exceptional, so it is usually not a wise idea to remove scores based on their 'exceptionalness'. Instead, use `exceptionalScores()`, which calls this function, to see how often participants answered exceptionally, and remove them based on that.

Value

A logical vector, indicating for each value in the supplied vector whether it is exceptional.

Examples

```r
exceptionalScore(
  c(1,1,2,2,2,3,3,3,4,4,4,5,5,5,5,6,6,7,8,20),
  prob=.05
);
```

---

exceptionalScores  
*Find exceptional scores*

Description

A function to detect participants that consistently respond exceptionally.
exceptionalScores 65

Usage

exceptionalScores(
  dat,
  items = NULL,
  exception = 0.025,
  totalOnly = TRUE,
  append = TRUE,
  both = TRUE,
  silent = FALSE,
  suffix = "_isExceptional",
  totalVarName = "exceptionalScores"
)

Arguments

dat  The dataframe containing the variables to inspect, or the vector to inspect (but for vectors, exceptionalScore() might be more useful).

items  The names of the variables to inspect.

exception  When an item will be considered exceptional, passed on as prob to exceptionalScore().

totalOnly  Whether to return only the number of exceptional scores for each row in the dataframe, or for each inspected item, which values are exceptional.

append  Whether to return the supplied dataframe with the new variable(s) appended (if TRUE), or whether to only return the new variable(s) (if FALSE).

both  Whether to look for both low and high exceptional scores (TRUE) or not (FALSE; see exceptionalScore()).

silent  Can be used to suppress messages.

suffix  If not returning the total number of exceptional values, for each inspected variable, a new variable is returned indicating which values are exceptional. The text string is appended to each original variable name to create the new variable names.

totalVarName  If returning only the total number of exceptional values, and appending these to the provided dataset, this text string is used as variable name.

Value

Either a vector containing the number of exceptional values, a dataset containing, for each inspected variable, which values are exceptional, or the provided dataset where either the total or the exceptional values for each variable are appended.

Examples

exceptionalScores(mtcars);
exportToHTML  

Exporting tables to HTML

Description

This function exports data frames or matrices to HTML, sending output to one or more of the console, viewer, and one or more files.

Usage

```r
exportToHTML(
  input,
  output = ufs::opts$get("tableOutput"),
  tableOutputCSS = ufs::opts$get("tableOutputCSS")
)
```

Arguments

- **input**: Either a `data.frame`, `table`, or `matrix`, or a list with three elements: `pre`, `input`, and `post`. The `pre` and `post` are simply prepended and postpended to the HTML generated based on the `input$input` element.
- **output**: The output: a character vector with one or more of "console" (the raw concatenated input, without conversion to HTML), "viewer", which uses the RStudio viewer if available, and one or more filenames in existing directories.
- **tableOutputCSS**: The CSS to use for the HTML table.

Value

Invisibly, the (potentially concatenated) `input` as character vector.

Examples

```r
exportToHTML(mtcars[1:5, 1:5]);
```

extractVarName  

Extract variable names

Description

Functions often get passed variables from within dataframes or other lists. However, printing these names with all their dollar signs isn’t very userfriendly. This function simply uses a regular expression to extract the actual name.

Usage

```r
extractVarName(x)
```
Arguments
  x       A character vector of one or more variable names.

Value
  The actual variables name, with all containing objects stripped off.

Author(s)
  Gjalt-Jorn Peters
  Maintainer: Gjalt-Jorn Peters gjalt-jorn@userfriendlyscience.com

Examples

extractVarName('mtcars$mpg');

faConfInt  Extract confidence bounds from psych’s factor analysis object

Description
  This function contains some code from a function in psych::psych-package that’s not exported print.psych.fa.ci but useful nonetheless. It basically takes the outcomes of a factor analysis and extracted the confidence intervals.

Usage
  faConfInt(fa)

Arguments
  fa       The object produced by the psych::fa() function from the psych::psych-package package. It is important that the n.iter argument of psych::fa() was set to a realistic number, because otherwise, no confidence intervals will be available.

Details
  This function extract confidence interval bounds and combines them with factor loadings using the code from the print.psych.fa.ci in psych::psych-package.

Value
  A list of dataframes, one for each extracted factor, with in each dataframe three variables:
  lo  lower bound of the confidence interval
  est point estimate of the factor loading
  hi  upper bound of the confidence interval
factorLoadingDiamondCIplot

Two-dimensional visualisation of factor analyses

Description

This function uses the `diamondPlot()` to visualise the results of a factor analyses. Because the factor loadings computed in factor analysis are point estimates, they may vary from sample to sample. The factor loadings for any given sample are usually not relevant; samples are but means to study populations, and so, researchers are usually interested in population values for the factor loadings. However, tables with lots of loadings can quickly become confusing and intimidating. This function aims to facilitate working with and interpreting factor analysis based on confidence intervals by visualising the factor loadings and their confidence intervals.

Usage

```r
factorLoadingDiamondCIplot(
  fa,
  xlab = "Factor Loading",
  colors = viridisPalette(max(2, fa$factors)),
  labels = NULL,
  theme = ggplot2::theme_bw(),
  sortAlphabetically = FALSE,
  ...
)
```

Arguments

- `fa` The object produced by the `psych::fa()` function from the `psych::psych` package. It is important that the `n.iter` argument of `psych::fa()` was set to a realistic number, because otherwise, no confidence intervals will be available.
xlab

The label for the x axis.

colors

The colors used for the factors. The default uses the discrete viridis palette, which is optimized for perceptual uniformity, maintaining its properties when printed in grayscale, and designed for colourblind readers.

labels

The labels to use for the items (on the Y axis).

theme

The ggplot2 theme to use.

sortAlphabetically

Whether to sort the items alphabetically.

... Additional arguments will be passed to ggDiamondLayer(). This can be used to set, for example, the transparency (alpha value) of the diamonds to a lower value using e.g. alpha=.5.

Value

A ggplot2::ggplot() plot with several ggDiamondLayer()s is returned.

Author(s)

Gjalt-Jorn Peters
Maintainer: Gjalt-Jorn Peters gjalt-jorn@userfriendlyscience.com

See Also

psych::fa()s, meansDiamondPlot(), meanSDtoDiamondPlot(), diamondPlot(), ggDiamondLayer()

Examples

## Not run:
### (Not run during testing because it takes too long and
### may generate warnings because of the bootstrapping of
### the confidence intervals)

factorLoadingDiamondCIplot(psych::fa(psych::Bechtoldt,
  nFactors=2,
  n.iter=50,
  n.obs=200));

### And using a lower alpha value for the diamonds to
### make them more transparent

factorLoadingDiamondCIplot(psych::fa(psych::Bechtoldt,
  nFactors=2,
  n.iter=50,
  n.obs=200),
  alpha=.5,
  size=1);

## End(Not run)
factorLoadingHeatmap Two-dimensional visualisation of factor analyses

Description

This function uses the diamondPlot() to visualise the results of a factor analyses. Because the factor loadings computed in factor analysis are point estimates, they may vary from sample to sample. The factor loadings for any given sample are usually not relevant; samples are but means to study populations, and so, researchers are usually interested in population values for the factor loadings. However, tables with lots of loadings can quickly become confusing and intimidating. This function aims to facilitate working with and interpreting factor analysis based on confidence intervals by visualising the factor loadings and their confidence intervals.

Usage

factorLoadingHeatmap(
  fa,
  xlab = "Factor Loading",
  colors = viridisPalette(max(2, fa$factors)),
  labels = NULL,
  showLoadings = FALSE,
  heatmap = FALSE,
  theme = ggplot2::theme_minimal(),
  sortAlphabetically = FALSE,
  digits = 2,
  labs = list(title = NULL, x = NULL, y = NULL),
  themeArgs = list(panel.grid = ggplot2::element_blank(), legend.position = "none",
                    axis.text.x = ggplot2::element_blank()),
  ...
)

Arguments

fa The object produced by the psych::fa() function from the psych::psych package. It is important that the n.iter argument of psych::fa() was set to a realistic number, because otherwise, no confidence intervals will be available.

xlab The label for the x axis.

colors The colors used for the factors. The default uses the discrete viridis palette, which is optimized for perceptual uniformity, maintaining its properties when printed in grayscale, and designed for colourblind readers.

labels The labels to use for the items (on the Y axis).

showLoadings Whether to show the factor loadings or not.

heatmap Whether to produce a heatmap or use diamond plots.

theme The ggplot2 theme to use.
sortAlphabetically
  Whether to sort the items alphabetically.

digits
  Number of digits to round to.
labs
  The labels to pass to ggplot2.

themeArgs
  Additional theme arguments to pass to ggplot2.

... Additional arguments will be passed to ggDiamondLayer(). This can be used to set, for example, the transparency (alpha value) of the diamonds to a lower value using e.g. alpha=.5.

Value

A ggplot2::ggplot() plot with several ggDiamondLayer()s is returned.

Author(s)

Gjalt-Jorn Peters
Maintainer: Gjalt-Jorn Peters gjalt-jorn@userfriendlyscience.com

See Also

psych::fa(), meansDiamondPlot(), meanSDtoDiamondPlot(), diamondPlot(), ggDiamondLayer()

Examples

```r
## Not run:
### (Not run during testing because it takes too long and
### may generate warnings because of the bootstrapping of
### the confidence intervals)

factorLoadingHeatmap(psych::fa(psych::Bechtoldt,
   nfactors=2,
   n.iter=50,
   n.obs=200));

### And using a lower alpha value for the diamonds to
### make them more transparent

factorLoadingHeatmap(psych::fa(psych::Bechtoldt,
   nfactors=2,
   n.iter=50,
   n.obs=200),
   alpha=.5,
   size=1);

## End(Not run)
```
fa_failsafe  
*Do factor-analysis, logging warnings and errors*

**Description**

Do factor-analysis, logging warnings and errors

**Usage**

```r
fa_failsafe(
    ..., 
    n.repeatOnWarning = 50, 
    warningTolerance = 2, 
    silentRepeatOnWarning = FALSE, 
    showWarnings = TRUE 
)
```

**Arguments**

- `...`: The arguments for `fa` in `psych`.
- `n.repeatOnWarning`: How often to repeat on warnings (in the hopes of getting a run without warnings).
- `warningTolerance`: How many warnings are accepted.
- `silentRepeatOnWarning`: Whether to be chatty or silent when repeating after warnings.
- `showWarnings`: Whether to show the warnings.

**Value**

A list with the `fa` object and a `warnings` and an `errors` object.

---

findShortestInterval  
*Find the shortest interval*

**Description**

This function takes a numeric vector, sorts it, and then finds the shortest interval and returns its length.

**Usage**

```r
findShortestInterval(x)
```
Arguments

x The numeric vector.

Value

The length of the shortest interval.

Author(s)

Gjalt-Jorn Peters
Maintainer: Gjalt-Jorn Peters gjalt-jorn@userfriendlyscience.com

Examples

findShortestInterval(c(1, 2, 4, 7, 20, 10, 15));

formatCI  
Pretty formatting of confidence intervals

Description

Pretty formatting of confidence intervals

Usage

formatCI(
  ci,
  sep = "; ",
  prefix = "[",
  suffix = "]",
  digits = 2,
  noZero = FALSE
)

Arguments

ci A confidence interval (a vector of 2 elements; longer vectors work, but I guess that wouldn’t make sense).
sep The separator of the values, usually "; " or ", ".
prefix, suffix The prefix and suffix, usually a type of opening and closing parenthesis/bracket.
digits The number of digits to which to round the values.
noZero Whether to strip the leading zero (before the decimal point), as is typically done when following APA style and displaying correlations, p values, and other numbers that cannot reach 1 or more.
Value
A character vector of one element.

See Also

noZero(), formatR(), formatPvalue()

Examples

### With leading zero ...
formatCI(c(0.55, 0.021));

### ... and without
formatCI(c(0.55, 0.021), noZero=TRUE);

formatPvalue Pretty formatting of p values

Description

Pretty formatting of p values

Usage

formatPvalue(values, digits = 3, spaces = TRUE, includeP = TRUE)

Arguments

<table>
<thead>
<tr>
<th>values</th>
<th>The p-values to format.</th>
</tr>
</thead>
<tbody>
<tr>
<td>digits</td>
<td>The number of digits to round to. Numbers smaller than this number will be shown as &lt;.001 or &lt;.0001 etc.</td>
</tr>
<tr>
<td>spaces</td>
<td>Whether to include spaces between symbols, operators, and digits.</td>
</tr>
<tr>
<td>includeP</td>
<td>Whether to include the 'p' and '='-symbol in the results (the '&lt;' symbol is always included).</td>
</tr>
</tbody>
</table>

Value

A formatted P value, roughly according to APA style guidelines. This means that the noZero function is used to remove the zero preceding the decimal point, and p values that would round to zero given the requested number of digits are shown as e.g. p<.001.

See Also

formatCI(), formatR(), noZero()
formatR

Pretty formatting of correlation coefficients

Description

Pretty formatting of correlation coefficients

Usage

formatR(r, digits = 2)

Arguments

r
  The Pearson correlation to format.

digits
  The number of digits to round to.

Value

The formatted correlation.

See Also

noZero(), formatCI(), formatPvalue()

Examples

formatR(cor(mtcars$mpg, mtcars$disp));
getData

Use a dialog to load data from an SPSS file

description

gedata() and getDat() provide an easy way to load SPSS datafiles.

Usage

gedata(
  filename = NULL,
  file = NULL,
  errorMessage = "[defaultErrorMessage]",
  applyRioLabels = TRUE,
  use.value.labels = FALSE,
  to.data.frame = TRUE,
  stringsAsFactors = FALSE,
  silent = FALSE,
  ...
)

gedat(..., dfName = "dat", backup = TRUE)

Arguments

filename, file It is possible to specify a path and filename to load here. If not specified, the
default R file selection dialogue is shown. file is still available for backward
compatibility but will eventually be phased out.

errorMessage The error message that is shown if the file does not exist or does not have the
right extension; [defaultErrorMessage] is replaced with a default error mes-

applyRioLabels Whether to apply the labels supplied by Rio. This will make variables that has
value labels into factors.

use.value.labels Only useful when reading from SPSS files: whether to read variables with value
labels as factors (TRUE) or numeric vectors (FALSE).

to.data.frame Only useful when reading from SPSS files: whether to return a dataframe or not.

stringsAsFactors Whether to read strings as strings (FALSE) or factors (TRUE).

silent Whether to suppress potentially useful information.

... Additional options, passed on to the function used to import the data (which
depends on the extension of the file).

dfname The name of the dataframe to create in the parent environment.

backup Whether to backup an object with name dfName, if one already exists in the
parent environment.
Value

ggetData returns the imported dataframe, with the filename from which it was read stored in the 'filename' attribute.

ggetDat is a simple wrapper for getData() which creates a dataframe in the parent environment, by default with the name 'dat'. Therefore, calling getDat() in the console will allow the user to select a file, and the data from the file will then be read and be available as 'dat'. If an object with dfName (i.e. 'dat' by default) already exists, it will be backed up with a warning. getDat() also invisibly returns the data.frame.

Note

ggetData() currently can't read from LibreOffice or OpenOffice files. There doesn't seem to be a platform-independent package that allows this. Non-CRAN package ROpenOffice from Omega-Hat should be able to do the trick, but fails to install (manual download and installation using https://www.omegahat.org produces "ERROR: dependency 'Rcompression' is not available for package 'ROpenOffice'" - and manual download and installation of RCompression produces "Please define LIB_ZLIB; ERROR: configuration failed for package 'Rcompression'"). If you have any suggestions, please let me know!

Examples

## Not run:
### Open a dialogue to read an SPSS file
ggetData();
### End(Not run)

---

ggBarChart

**Bar chart using ggplot**

Description

This function provides a simple interface to create a *ggplot2::ggplot()* bar chart.

Usage

```r
ggBarChart(vector, plotTheme = ggplot2::theme_bw(), ...)
```

Arguments

- `vector` The vector to display in the bar chart.
- `plotTheme` The theme to apply.
- `...` And additional arguments are passed to *ggplot2::geom_bar()*.
ggBoxplot

Value

A `ggplot2::ggplot()` plot is returned.

Author(s)

Gjalt-Jorn Peters
Maintainer: Gjalt-Jorn Peters gjalt-jorn@userfriendlyscience.com

See Also

`ggplot2::geom_bar()`

Examples

```r
ggBarChart(mtcars$cyl);
```

Description

This function provides a simple interface to create a `ggplot` box plot, organising different boxplots by levels of a factor is desired, and showing row numbers of outliers.

Usage

```r
ggBoxplot(
  dat,
  y = NULL,
  x = NULL,
  labelOutliers = TRUE,
  outlierColor = "red",
  theme = ggplot2::theme_bw(),
  ...
)
```

Arguments

- `dat` Either a vector of values (to display in the box plot) or a dataframe containing variables to display in the box plot.
- `y` If `dat` is a dataframe, this is the name of the variable to make the box plot of.
- `x` If `dat` is a dataframe, this is the name of the variable (normally a factor) to place on the X axis. Separate box plots will be generate for each level of this variable.
- `labelOutliers` Whether or not to label outliers.
outlierColor If labeling outliers, this is the color to use.
theme The theme to use for the box plot.
... Any additional arguments will be passed to geom_boxplot.

Details
This function is based on JasonAizkalns’ answer to a question on Stack Exchange (Cross Validated; see https://stackoverflow.com/questions/33524669/labeling-outliers-of-boxplots-in-r).

Value
A ggplot plot is returned.

Author(s)
Jason Aizkalns; implemented in this package (and tweaked a bit) by Gjalt-Jorn Peters.
Maintainer: Gjalt-Jorn Peters gjalt-jorn@userfriendlyscience.com

See Also
ggboxplot

Examples

### A box plot for miles per gallon in the mtcars dataset:
```r
ggBoxplot(mtcars$mpg);
```

### And separate for each level of 'cyl' (number of cylinder):
```r
ggBoxplot(mtcars, y='mpg', x='cyl');
```

---

ggEasyBar

Convenience functions for ggplots based on multiple variables

Description
These are convenience functions to quickly generate plots for multiple variables, with the variables in the y axis.

Usage

```r
ggEasyBar(
  data,
  items = NULL,
  labels = NULL,
  sortByMean = TRUE,
  xlab = NULL,
)```
ggEasyBar

```r
ggEasyBar(
  ylab = NULL,
  scale_fill_function = NULL,
  fontColor = "white",
  fontSize = 2,
  labelMinPercentage = 1,
  showInLegend = "both",
  legendRows = 2,
  legendValueLabels = NULL,
  biAxisLabels = NULL
)
```

```r
ggEasyRidge(
  data,
  items = NULL,
  labels = NULL,
  sortByMean = TRUE,
  xlab = NULL,
  ylab = NULL
)
```

**Arguments**

- **data**
  
  The dataframe containing the variables.

- **items**
  
  The variable names (if not provided, all variables will be used).

- **labels**
  
  Labels can optionally be provided; if they are, these will be used instead of the variable names.

- **sortByMean**
  
  Whether to sort the variables by mean value.

- **xlab, ylab**
  
  The labels for the x and y axes.

- **scale_fill_function**
  
  The function to pass to `ggplot()` to provide the colors of the bars. If NULL, set to `ggplot2::scale_fill_viridis_d(labels = legendValueLabels, guide = ggplot2::guide_legend(title = NULL, nrow=legendRows, byrow=TRUE))`.

- **fontColor, fontSize**
  
  The color and size of the font used to display the labels

- **labelMinPercentage**
  
  The minimum percentage that a category must reach before the label is printed (in whole percentages, i.e., on a scale from 0 to 100).

- **showInLegend**
  
  What to show in the legend in addition to the values: nothing ("none"), the frequencies ("freq"), the percentages ("perc"), or both ("both"). This is only used if only one variable is shown in the plot; afterward, after all, the absolute frequencies and percentages differ for each variable.

- **legendRows**
  
  Number or rows in the legend.

- **legendValueLabels**
  
  Labels to use in the legend; must be a vector of the same length as the number of categories in the variables.
ggPie

biAxisLabels This can be used to specify labels to use if you want to use labels on both the left and right side. This is mostly useful when plotting single questions or semantic differentials. This must be a list with two character vectors, `leftAnchors` and `rightAnchors`, which must each have the same length as the number of items specified in `items`. See the examples for, well, examples.

Value

A `ggplot()` plot is returned.

Author(s)

Gjalt-Jorn Peters
Maintainer: Gjalt-Jorn Peters gjalt-jorn@userfriendlyscience.com

See Also

`geom_ridgeline()`, `geom_bar()`

Examples

```r
ggEasyBar(mtcars, c('gear', 'carb'));
ggEasyRidge(mtcars, c('disp', 'hp'));

### When plotting single questions, if you want to show the anchors:
ggEasyBar(mtcars, c('gear'),
   biAxisLabels=list(leftAnchors="Fewer",
                    rightAnchors="More"));

### Or for multiple questions (for e.g. semantic differentials):
ggEasyBar(mtcars, c('gear', 'carb'),
   biAxisLabels=list(leftAnchors=c("Fewer", "Lesser"),
                    rightAnchors=c("More", "Greater"));
```

Description

This function creates a pie chart. Note that these are generally quite strongly advised against, as people are not good at interpreting relative frequencies on the basis of pie charts.

Usage

```r
ggPie(vector, scale_fill = ggplot2::scale_fill_viridis_d())
```
Arguments

- `vector`: The vector (best to pass a factor).
- `scale_fill`: The `ggplot` scale fill function to use for the colors.

Value

A `ggplot` pie chart.

Note

This function is very strongly based on the Mathematical Coffee post at http://mathematicalcoffee.blogspot.com/2014/06/ggpie-pie-graphs-in-ggplot2.html.

Examples

```r
ggPie(mtcars$cyl);
```

---

### ggProportionPlot

**Sample distribution based plotting of proportions**

**Description**

This function visualises percentages, but avoids a clear cut for the sample point estimate, instead using the confidence (as in confidence interval) to create a gradient. This effectively hinders drawing conclusions on the basis of point estimates, thereby urging a level of caution that is consistent with what the data allows.

**Usage**

```r
ggProportionPlot(
  dat,
  items = NULL,
  loCategory = NULL,
  hiCategory = NULL,
  subQuestions = NULL,
  leftAnchors = NULL,
  rightAnchors = NULL,
  compareHiToLo = TRUE,
  showDiamonds = FALSE,
  diamonds.conf.level = 0.95,
  diamonds.alpha = 1,
  na.rm = TRUE,
  barHeight = 0.4,
  conf.steps = seq(from = 0.001, to = 0.999, by = 0.001),
  scale_color = c("#21908CFF", "#FDE725FF"),
  scale_fill = c("#21908CFF", "#FDE725FF"),
)```
ggProportionPlot

    rank.conf = FALSE,
    linetype = 1,
    theme = ggplot2::theme_bw(),
    returnPlotOnly = TRUE

)  

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

## S3 method for class 'ggProportionPlot'
grid.draw(x, ...)

Arguments

dat          The dataframe containing the items (variables), or a vector.
items        The names of the items (variables). If none are specified, all variables in the
dataframe are used.
loCategory   The value of the low category (usually 0). If not provided, the minimum value
             is used.
hiCategory   The value of the high category (usually 1). If not provided, the maximum value
             is used.
subQuestions The labels to use for the variables (for example, different questions). The vari-
             able names are used if these aren’t provided.
leftAnchors  The labels for the low categories. The values are used if these aren’t provided.
rightAnchors The labels for the high categories. The values are used if these aren’t provided.
compareHiToLo Whether to compare the percentage of low category values to the total of the
                   low category values and the high category values, or whether to ignore the high
                   category values and compute the percentage of low category values relative to
                   all cases. This can be useful when a variable has more than two values, and you
                   only want to know/plot the percentage relative to the total number of cases.
showDiamonds Whether to add diamonds to illustrate the confidence intervals.
diamonds.conf.level The confidence level of the diamonds’ confidence intervals.
diamonds.alpha The alpha channel (i.e. transparency, or rather ‘obliqueness’) of the diamonds.
na.rm        Whether to remove missing values.
barHeight    The height of the bars, or rather, half the height. Use .5 to completely fill the
             space.
conf.steps   The number of steps to use to generate the confidence levels for the proportion.
scale_color, scale_fill  A vector with two values (valid colors), that are used for the colors (stroke) and
                          fill for the gradient; both vectors should normally be the same, but if you feel
                          adventurous, you can play around with the number of conf.steps and this. If
                          you specify only one color, no gradient is used but a single color (i.e. specifying
                          the same single color for both scale_color and scale_fill simply draws bars
                          of that color).
ggProportionPlot

rank.conf
linetype
theme
returnPlotOnly
x
...

Details

This function used `confIntProp()` to compute confidence intervals for proportions at different levels of confidence. The confidence interval bounds at those levels of confidence are then used to draw rectangles with colors in a gradient that corresponds to the confidence level.

Note that percentually, the gradient may not look continuous because at the borders between lighter and darker rectangles, the shade of the lighter rectangle is perceived as even lighter than it is, and the shade of the darker rectangle is perceived as even darker. This makes it seem as if each rectangle is coloured with a gradient in the opposite direction.

Value

A `ggplot2()` object (if `returnPlotOnly` is TRUE), or an object containing that `ggplot2()` object and intermediate products.

Author(s)

Gjalt-Jorn Peters

Maintainer: Gjalt-Jorn Peters gjalt-jorn@userfriendlyscience.com

See Also

`confIntProp()` and `binom.test()`

Examples

```r
### V/S (no idea what this is: ?mtcars only mentions 'V/S' :-))
### and transmission (automatic vs manual)
ggProportionPlot(mtcars, items=c('vs', 'am'));

### Number of cylinders, by default comparing lowest value
### (4) to highest (8):
ggProportionPlot(mtcars, items=c('cyl'));

# Not run:
### Not running these to save time during package building/checking
```
### We can also compare 4 to 6:
```
ggProportionPlot(mtcars, items=c('cyl'),
                  hiCategory=6);
```
### Now compared to total records, instead of to
### highest value (hiCategory is ignored then)
```
ggProportionPlot(mtcars, items=c('cyl'),
                  compareHiToLo=FALSE);
```
### And for 6 cylinders:
```
ggProportionPlot(mtcars, items=c('cyl'),
                  loCategory=6, compareHiToLo=FALSE);
```
### And for 8 cylinders:
```
ggProportionPlot(mtcars, items=c('cyl'),
                  loCategory=8, compareHiToLo=FALSE);
```
### And for 8 cylinders with different labels
```
ggProportionPlot(mtcars, items=c('cyl'),
                  loCategory=8,
                  subQuestions="Cylinders",
                  leftAnchors="Eight",
                  rightAnchors="Four
or
six",
                  compareHiToLo=FALSE);
```
### ... And showing the diamonds for the confidence intervals
```
ggProportionPlot(mtcars, items=c('cyl'),
                  loCategory=8,
                  subQuestions="Cylinders",
                  leftAnchors="Eight",
                  rightAnchors="Four
or
six",
                  compareHiToLo=FALSE,
                  showDiamonds=TRUE);
```
### Using less steps for the confidence levels and changing
### the fill colours
```
ggProportionPlot(mtcars, items=c('vs', 'am'),
                  showDiamonds = TRUE,
                  scale_fill = c("#B63679FF", "#FCDBFFF"),
                  conf.steps=seq(from=0.0001, to=.9999, by=.2));
```

---

**ggqqq**

**Easy ggplot Q-Q plot**

**Description**

This function creates a qq-plot with a confidence interval.
Usage

```r
ggqq(
  x,
  distribution = "norm",
  ...,  
  ci = TRUE,
  line.estimate = NULL,
  conf.level = 0.95,
  sampleSizeOverride = NULL,
  observedOnX = TRUE,
  scaleExpected = TRUE,
  theoryLab = "Theoretical quantiles",
  observeLab = "Observed quantiles",
  theme = ggplot2::theme_bw()
)
```

Arguments

- **x**: A vector containing the values to plot.
- **distribution**: The distribution to (a ‘d’ and ‘q’ are prepended, and the resulting functions are used, e.g. `dnorm` and `qnorm` for the normal curve).
- **...**: Any additional arguments are passed to the quantile function (e.g. `qnorm`). Because of these dots, any following arguments must be named explicitly.
- **ci**: Whether to show the confidence interval.
- **line.estimate**: Whether to show the line showing the match with the specified distribution (e.g. the normal distribution).
- **conf.level**: The confidence of the confidence level around the estimate for the specified distribution.
- **sampleSizeOverride**: It can be desirable to get the confidence intervals for a different sample size (when the sample size is very large, for example, such as when this plot is generated by the function `normalityAssessment`). That different sample size can be specified here.
- **observedOnX**: Whether to plot the observed values (if TRUE) or the theoretically expected values (if FALSE) on the X axis. The other is plotted on the Y axis.
- **scaleExpected**: Whether the scale the expected values to match the scale of the variable. This option is provided to be able to mimic SPSS’ Q-Q plots.
- **theoryLab**: The label for the theoretically expected values (on the Y axis by default).
- **observeLab**: The label for the observed values (on the Y axis by default).
- **theme**: The theme to use.

Details

This is strongly based on the answer by user Floo0 to a Stack Overflow question at Stack Exchange (see [https://stackoverflow.com/questions/4357031/qqnorm-and-qline-in-ggplot2/27191036#27191036](https://stackoverflow.com/questions/4357031/qqnorm-and-qline-in-ggplot2/27191036)), also posted at GitHub (see [https://gist.github.com/rentrop/d39a8406ad8af2a1066c](https://gist.github.com/rentrop/d39a8406ad8af2a1066c)). That code is in turn based on the `qqPlot()` function from the `car` package.
Value

A \texttt{ggplot} plot is returned.

Author(s)

John Fox and Floo0; implemented in this package (and tweaked a bit) by Gjalt-Jorn Peters.
Maintainer: Gjalt-Jorn Peters gjalt-jorn@userfriendlyscience.com

Examples

\begin{verbatim}
ggqq(mtcars$mpg);
\end{verbatim}

\begin{verbatim}
\textbf{ggSave} \hspace{1cm} \textit{Save a ggplot with specific defaults}
\end{verbatim}

Description

This function is vectorized over all argument except 'plot': so if you want to save multiple versions, simply provide vectors. Vectors of length 1 will be recycled using \texttt{rep()}; otherwise vectors have to all be the same length as \texttt{file}.

Usage

\begin{verbatim}
ggSave(
  file = NULL,
  plot = ggplot2::last_plot(),
  width = ufs::opts$get("ggSaveFigWidth"),
  height = ufs::opts$get("ggSaveFigHeight"),
  units = ufs::opts$get("ggSaveUnits"),
  dpi = ufs::opts$get("ggSaveDPI"),
  device = NULL,
  type = NULL,
  bg = "transparent",
  preventType = ufs::opts$get("ggSavePreventType"),
  ...
)
\end{verbatim}

Arguments

\begin{verbatim}
file The file where to save to.
plot The plot to save; if omitted, the last drawn plot is saved.
height, width The dimensions of the plot, specified in units.
units The units, 'cm', 'mm', or 'in'.
dpi The resolution (dots per inch). This argument is vectorized.
\end{verbatim}
### heading

**device**  
The graphic device; is inferred from the file if not specified.

**type**  
An additional arguments for the graphic device.

**bg**  
The background (e.g. 'white').

**preventType**  
Whether to prevent passing a value for the type argument to `ggplot2::ggsave()`. This is prevented by default since ggplot switched to using the ragg device by default, resulting in throwing a warning ("Warning: Using ragg device as default. Ignoring type and antialias arguments") if something if passed for 'type'.

...  
Any additional arguments are passed on to `ggplot2::ggsave()`.

**Value**

The plot, invisibly.

**Examples**

```r
plot <- ufs::ggBoxplot(mtcars, "mpg");
ggsave(file=tempfile(fileext=".png"), plot=plot);
```

---

### heading

The heading text: pasted together with no separator.

**headingLevel**  
The level of the heading; the default can be set with e.g. `ufs::opts$set(defaultHeadingLevel=1)`.

**output**  
Whether to output to HTML ("html") or markdown (anything else).

**cat**  
Whether to cat (print) the heading or just invisibly return it.

**Value**

The heading, invisibly.

---

**Description**

This is just a convenience function to print a markdown or HTML heading at a given 'depth'.

**Usage**

```r
heading(
  ...,  
  headingLevel = ufs::opts$get("defaultHeadingLevel"),
  output = "markdown",
  cat = TRUE
)
```

**Arguments**

...  
The heading text: pasted together with no separator.

headingLevel  
The level of the heading; the default can be set with e.g. `ufs::opts$set(defaultHeadingLevel=1)`.

output  
Whether to output to HTML ("html") or markdown (anything else).

cat  
Whether to cat (print) the heading or just invisibly return it.
ifelseObj

### Conditional returning of an object

#### Description

The ifelseObj function just evaluates a condition, returning one object if it’s true, and another if it’s false.

#### Usage

```r
ifelseObj(condition, ifTrue, ifFalse)
```

#### Arguments

- **condition**: Condition to evaluate.
- **ifTrue**: Object to return if the condition is true.
- **ifFalse**: Object to return if the condition is false.

#### Value

One of the two objects

#### Examples

```r
dat <- ifelseObj(sample(c(TRUE, FALSE), 1), mtcars, Orange);
```

---

insertFigureCaption

### Insert numbered caption

#### Description

These functions can be used to manually insert a numbered caption. These functions have been designed to work well with `setFigCapNumbering()` and `setTabCapNumbering()`. This is useful when inserting figures or tables in an R Markdown document when you use automatic caption numbering for knitr chunks, but are inserting a table or figure that isn’t produced in a knitr chunk while still retaining the automatic numbering. `insertNumberedCaption()` is the general-purpose function; you will typically only use `insertFigureCaption()` and `insertTableCaption()`.

```r
# Examples
```

---
Usage

insertFigureCaption(
  captionText = "",
  captionName = "fig.cap",
  prefix = getOption(paste0(optionName, ".prefix"), "Figure %s: "),
  suffix = getOption(paste0(optionName, ".suffix"), ""),
  optionName = paste0("setCaptionNumbering_", captionName),
  resetCounterTo = NULL
)

insertNumberedCaption(
  captionText = "",
  captionName = "fig.cap",
  prefix = getOption(paste0(optionName, ".prefix"), "Figure %s: "),
  suffix = getOption(paste0(optionName, ".suffix"), ""),
  optionName = paste0("setCaptionNumbering_", captionName),
  resetCounterTo = NULL
)

insertTableCaption(
  captionText = "",
  captionName = "tab.cap",
  prefix = getOption(paste0(optionName, ".prefix"), "Table %s: "),
  suffix = getOption(paste0(optionName, ".suffix"), ""),
  optionName = paste0("setCaptionNumbering_", captionName),
  resetCounterTo = NULL
)

Arguments

captionText The text of the caption.
captionName The name of the caption; by default, for tables, "tab.cap".
prefix, suffix The prefix and suffix texts; base::sprintf() is used to insert the number in
the position taken up by \%s.
optionName The name of the option to use to save the number counter.
resetCounterTo If a numeric value, the counter is reset to that value.

Value

The caption in a character vector.

Examples

insertNumberedCaption("First caption");
insertNumberedCaption("Second caption");
sectionNumber <- 12;
insertNumberedCaption("Third caption",
  prefix = paste0("Table ",

insertFigureCaption("The figure caption");
insertTableCaption("The table caption");
invertItem

Description

Inverts items (as in, in a questionnaire), by calling *invertItem* on all relevant items.

Usage

```
invertItem(item, fullRange = NULL, ignorePreviousInversion = FALSE)
invertItems(dat, items = NULL, ...)
```

Arguments

- **item**: The vector to invert.
- **fullRange**: The full range; will otherwise be derived from the vector.
- **ignorePreviousInversion**: Whether to avoid inverting items that were already inverted.
- **dat**: The dataframe containing the variables to invert.
- **items**: The names or indices of the variables to invert. If not supplied (i.e. NULL), all variables in the dataframe will be inverted.
- **...**: Arguments (parameters) passed on to `data.frame` when recreating that after having used `lapply`.

Value

The dataframe with the specified items inverted.

Author(s)

Gjalt-Jorn Peters
Maintainer: Gjalt-Jorn Peters gjalt-jorn@userfriendlyscience.com

See Also

- *invertItem*

Examples

```
invertItems(mtcars, c('cyl'));
```
iqrOutlier

Identify outliers according to the IQR criterion

Description

The IQR criterion holds that any value lower than one-and-a-half times the interquartile range below the first quartile, or higher than one-and-a-half times the interquartile range above the third quartile, is an outlier. This function returns a logical vector that identifies those outliers.

Usage

iqrOutlier(x)

Arguments

x

The vector to scan for outliers.

Value

A logical vector where TRUE identifies outliers.

Author(s)

Gjalt-Jorn Peters

Maintainer: Gjalt-Jorn Peters gjalt-jorn@userfriendlyscience.com

See Also

IQR

Examples

### One outlier in the miles per gallon
iqrOutlier(mtcars$mpg);
irpplot

Visualising individual response patterns

Description

Visualising individual response patterns

Usage

irpplot(
  data,
  row,
  columns,
  dataName = NULL,
  title = paste("Row", row, " in dataset", dataName)
)

Arguments

data A dataframe with the dataset containing the responses.
row A vector with indices of the rows for which you want the individual response patterns. These can be either the relevant row numbers, or if character row names are set, the names of the relevant rows.
columns A vector with the names of the variables you want the individual response patterns for.
dataName, title Optionally, you can override the dataset name that is used in the title; or, the title (the dataset name is only used in the title).

Value

A ggplot2::ggplot().

Examples

### Get a dataset
dat <- ufs::bfi;

### Show the individual responses for
### the tenth participant
irpplot(dat, 10, 1:20);

### Set some missing values
dat[10, c(1, 5, 15)] <- NA;

### Show the individual responses again
irpplot(dat, 10, 1:20);
**is.nr**

NULL and NA 'proof' checking of whether something is a number

Description

Convenience function that returns TRUE if the argument is not null, not NA, and is.numeric.

Usage

\[
\text{is.nr}(x)
\]

Arguments

- **x**
  
  The value or vector to check.

Value

TRUE or FALSE.

Examples

\[
\begin{align*}
\text{is.nr}(8); & \quad \text{### Returns TRUE} \\
\text{is.nr}(\text{NULL}); & \quad \text{### Returns FALSE} \\
\text{is.nr}(\text{NA}); & \quad \text{### Returns FALSE}
\end{align*}
\]

**is.odd**

Checking whether numbers are odd or even

Description

Checking whether numbers are odd or even

Usage

\[
\text{is.odd}(\text{vector})
\]

\[
\text{is.even}(\text{vector})
\]

Arguments

- **vector**
  
  The vector to process

Value

A logical vector.
isTrue

Examples

```r
is.odd(4);
```

<table>
<thead>
<tr>
<th>isTrue</th>
<th>More flexible version of isTRUE</th>
</tr>
</thead>
</table>

Description

Returns TRUE for TRUE elements, FALSE for FALSE elements, and whatever is specified in `na` for NA items.

Usage

```r
isTrue(x, na = FALSE)
```

Arguments

- **x**: The vector to check for TRUE, FALSE, and NA values.
- **na**: What to return for NA values.

Value

A logical vector.

Examples

```r
isTrue(c(TRUE, FALSE, NA));
isTrue(c(TRUE, FALSE, NA), na=TRUE);
```

<table>
<thead>
<tr>
<th>kblXtra</th>
<th>Wrapper for kableExtra for consistent ufs table styling</th>
</tr>
</thead>
</table>

Description

Wrapper for kableExtra for consistent ufs table styling
Usage

```r
kblXtra(
  x,
  digits = 2,
  format = "html",
  escape = FALSE,
  print = TRUE,
  viewer = FALSE,
  kable_classic = FALSE,
  lightable_options = "striped",
  html_font = "\"Arial Narrow\", \"Source Sans Pro\", sans-serif",
  full_width = TRUE,
  table.attr = "style='border:0px solid black !important;'",
  ...
)
```

Arguments

- **x**: The dataframe to print
- **digits**, **format**, **escape**, **table.attr**, **lightable_options**, **html_font**, **full_width**: Defaults that are passed to `knitr::kable()`
- **print**: Whether to print the table
- **viewer**: Whether to show the table in the viewer
- **kable_classic**: Whether to call `kable_classic`; otherwise, `kable_styling` is called.
- **...**: Additional arguments are passed to `knitr::kable()`

Value

The table, invisibly.

Examples

```r
kblXtra(mtcars);
```
Usage

knitAndSave(
  plotToDraw,
  figCaption,
  file = NULL,
  path = NULL,
  figWidth = ufs::opts$get("ggSaveFigWidth"),
  figHeight = ufs::opts$get("ggSaveFigHeight"),
  units = ufs::opts$get("ggSaveUnits"),
  dpi = ufs::opts$get("ggSaveDPI"),
  catPlot = ufs::opts$get("knitAndSave.catPlot"),
  ...
)

Arguments

plotToDraw The plot to knit using knitFig() and save using ggSave().
figCaption The caption of the plot (used as filename if no filename is specified).
file, path The filename to use when saving the plot, or the path where to save the file if no
filename is provided (if path is also omitted, getWd() is used).
figWidth, figHeight The plot dimensions, by default specified in inches (but 'units' can be set which
is then passed on to ggSave()).
units, dpi The units and DPI of the image which are then passed on to ggSave().
catPlot Whether to use cat() to print the knitr fragment.
... Additional arguments are passed on to ggSave(). Note that file (and ...) are
vectorized (see the ggSave() manual page).

Value

The knitFig() result, visibly.

Examples

## Not run: plot <- ggBoxplot(mtcars, 'mpg');
knitAndSave(plot, figCaption="a boxplot", file=tempfile(fileext=".png"));
## End(Not run)

knitFig Easily knit a custom figure fragment

Description

This function was written to make it easy to knit figures with different, or dynamically generated,
widths and heights (and captions) in the same chunk when working with R Markdown.
Usage

knitFig(
plotToDraw,
  template = getOption("ufs.knitFig.template", NULL),
  figWidth = ufs::opts$get("ggSaveFigWidth"),
  figHeight = ufs::opts$get("ggSaveFigHeight"),
  figCaption = "A plot.",
  chunkName = NULL,
  returnRaw = FALSE,
  catPlot = ufs::opts$get("knitFig.catPlot"),
  ...)
)

Arguments

plotToDraw   The plot to draw, e.g. a ggplot plot.
template     A character value with the knit_expand template to use.
figWidth     The width to set for the figure (in inches).
figHeight    The height to set for the figure (in inches).
figCaption   The caption to set for the figure.
chunkName    Optionally, the name for the chunk. To avoid problems because multiple chunks
              have the name "unnamed-chunk-1", if no chunk name is provided, digest::digest() is used to
t              generate an MD5-hash from Sys.time.
returnRaw    Whether to cat() the result (TRUE) or whether to return it as knitr::asis_output()
t              object (FALSE).
catPlot      Whether to use the base::cat() function to print the code for the plot, and
t              return the result invisibly. If not, the result is returned visible, and so probably
              printed anyway.
...           Any additional arguments are passed on to knit_expand.

Value

This function returns nothing, but uses knit_expand and knit to cat the result.

Author(s)

Gjalt-Jorn Peters
Maintainer: Gjalt-Jorn Peters gjalt-jorn@userfriendlyscience.com

See Also

knit_expand and knit

Examples

## Not run: knitFig(ggBoxplot(mtcars, 'mpg'))
makeScales

<table>
<thead>
<tr>
<th>makeScales</th>
<th>Title</th>
</tr>
</thead>
</table>

**Description**

Title

**Usage**

makeScales(data, scales, append = TRUE)

**Arguments**

data: The dataframe containing the variables (the items).
scales: A list of character vectors with the items in each scale, where each vectors’ name is the name of the scale.
append: Whether to return the dataframe including the new variables (TRUE), or a dataframe with only those new variables (FALSE).

**Value**

Either a dataframe with the newly created variables, or the supplied dataframe with the newly created variables appended.

**Examples**

### First generate a list with the scales
scales <- list(scale1 = c('mpg', 'cyl'), scale2 = c('disp', 'hp'));

### Create the scales and add them to the dataframe
makeScales(mtcars, scales);

massConvertToNumeric

<table>
<thead>
<tr>
<th>massConvertToNumeric</th>
<th>Converting many dataframe columns to numeric</th>
</tr>
</thead>
</table>

**Description**

This function makes it easy to convert many dataframe columns to numeric.

**Usage**

massConvertToNumeric(
  dat,
  byFactorLabel = FALSE,
  ignoreCharacter = TRUE,
  stringsAsFactors = FALSE
)
Arguments

- **dat**: The dataframe with the columns.
- **byFactorLabel**: When converting factors, whether to do this by their label value (TRUE) or their level value (FALSE).
- **ignoreCharacter**: Whether to convert (FALSE) or ignore (TRUE) character vectors.
- **stringsAsFactors**: In the returned dataframe, whether to return string (character) vectors as factors or not.

Value

A data.frame.

Examples

```r
### Create a dataset
a <- data.frame(var1 = factor(1:4),
    var2 = as.character(5:6),
    stringsAsFactors=FALSE);

### Ignores var2
b <- ufs::massConvertToNumeric(a);

### Converts var2
c <- ufs::massConvertToNumeric(a,
    ignoreCharacter = FALSE);
```

```
meanConfInt A confidence interval for the mean

Description

A confidence interval for the mean

Usage

```r
meanConfInt(
    vector = NULL,
    mean = NULL,
    sd = NULL,
    n = NULL,
    se = NULL,
    conf.level = 0.95
)
```

```r
## S3 method for class 'meanConfInt'
print(x, digits = 2, ...)
```
Arguments

vector A vector with raw data points - either specify this or a mean and then either an sd and n or an se.
mean A mean.
sd, n A standard deviation and sample size; can be specified to compute the standard error.
se The standard error (can be specified instead of sd and n).
conf.level The confidence level of the interval.
x, digits, ... Respectively the object to print, the number of digits to round to, and any additional arguments to pass on to the print function.

Value

An object with elements input, intermediate, and output, where output holds the result in list ci.

Examples

meanConfInt(mean=5, sd=2, n=20);

meansDiamondPlot

Diamond plots

Description

This function generates a so-called diamond plot: a plot based on the forest plots that are commonplace in meta-analyses. The underlying idea is that point estimates are uninformative, and it would be better to focus on confidence intervals. The problem of the points with errorbars that are commonly employed is that the focus the audience’s attention on the upper and lower bounds, even though those are the least relevant values. Using diamonds remedies this.

Usage

meansDiamondPlot(
data, items = NULL, labels = NULL, decreasing = NULL, conf.level = 0.95, showData = TRUE, dataAlpha = 0.1, dataSize = 3, dataColor = "#444444", diamondColors = NULL, jitterWidth = 0.5,
jitterHeight = 0.4,
returnLayerOnly = FALSE,
xlab = "Scores and means",
ylab = NULL,
theme = ggplot2::theme_bw(),
xbreaks = "auto",
outputFile = NULL,
outputWidth = 10,
outputHeight = 10,
ggsaveParams = ufs::opts$get("ggsaveParams"),
dat = NULL,
...)

Arguments

data, dat  The dataframe containing the variables (items) to show in the diamond plot (the name dat for this argument is deprecated but still works for backward compatibility).

items  Optionally, the names (or numeric indices) of the variables (items) to show in the diamond plot. If NULL, all columns (variables, items) will be used.

labels  A character vector of labels to use instead of column names from the dataframe.

decreasing  Whether to sort the variables (rows) in the diamond plot decreasing (TRUE), increasing (FALSE), or not at all (NULL).

conf.level  The confidence of the confidence intervals.

showData  Whether to show the raw data or not.

dataAlpha  This determines the alpha (transparency) of the data points. Note that argument alpha can be used to set the alpha of the diamonds; this is eventually passed on to ggDiamondLayer().

dataSize  The size of the data points.

dataColor  The color of the data points.

diamondColors  A vector of the same length as there are rows in the dataframe, to manually specify colors for the diamonds.

jitterWidth  How much to jitter the individual datapoints horizontally.

jitterHeight  How much to jitter the individual datapoints vertically.

returnLayerOnly  Set this to TRUE to only return the ggplot() layer of the diamondplot, which can be useful to include it in other plots.

xlab, ylab  The labels of the X and Y axes.

theme  The theme to use.

xbreaks  Where the breaks (major grid lines, ticks, and labels) on the x axis should be.

outputFile  A file to which to save the plot.

outputWidth, outputHeight  Width and height of saved plot (specified in centimeters by default, see ggsaveParams).
meansDiamondPlot

**ggsaveParams**  Parameters to pass to ggsave when saving the plot.

Additional arguments are passed to `diamondPlot()` and eventually to `ggDiamondLayer()`. This can be used to, for example, specify two or more colors to use to generate a gradient (using `generateColors` and maybe `fullColorRange`).

**Value**

A `ggplot()` plot with a `ggDiamondLayer()` is returned.

**Author(s)**

Gjalt-Jorn Peters

Maintainer: Gjalt-Jorn Peters gjalt-jorn@userfriendlyscience.com

**See Also**

See `diamondPlot()`, `meanSDtoDiamondPlot()`, `ggDiamondLayer()`, `factorLoadingDiamondCIplot()`

**Examples**

```r
tmpDf <- data.frame(item1 = rnorm(50, 1.6, 1),
                   item2 = rnorm(50, 2.6, 2),
                   item3 = rnorm(50, 4.1, 3));

### A simple diamond plot
meansDiamondPlot(tmpDf);

### A diamond plot with manually specified labels and colors
meansDiamondPlot(tmpDf,
                 labels=c('First',
                          'Second',
                          'Third'),
                 diamondColors=c('blue', 'magenta', 'yellow'));

### Using a gradient for the colors
meansDiamondPlot(tmpDf,
                 labels=c('First',
                          'Second',
                          'Third'),
                 generateColors = c("magenta", "cyan"),
                 fullColorRange = c(1,5));
```
Diamond plot: means

Description

Diamond plot: means

Usage

meansDiamondPlotjmv(data, items, conf.level = 95, showData = TRUE)

Arguments

data .
items .
conf.level .
showData .

Value

A results object containing:

results$text a html

results$diamondPlot an image

A diamond plot based on means, standard deviations, and sample sizes

Description

This function generates a so-called diamond plot: a plot based on the forest plots that are commonplace in meta-analyses. The underlying idea is that point estimates are uninformative, and it would be better to focus on confidence intervals. The problem of the points with errorbars that are commonly employed is that the focus the audience's attention on the upper and lower bounds, even though those are the least relevant values. Using diamonds remedies this.

Usage

meanSDtoDiamondPlot(
  dat = NULL,
  means = 1,
  sds = 2,
  ns = 3,
meanSDtoDiamondPlot

labels = NULL,
colorCol = NULL,
conf.level = 0.95,
lab = "Means",
outputFile = NULL,
outputWidth = 10,
outputHeight = 10,
ggsaveParams = ufs::opts$get("ggsaveParams"),

Arguments

dat The dataset containing the means, standard deviations, sample sizes, and possible labels and manually specified colors.

means Either the column in the dataframe containing the means, as numeric or as character index, or a vector of means.

sds Either the column in the dataframe containing the standard deviations, as numeric or as character index, or a vector of standard deviations.

ns Either the column in the dataframe containing the sample sizes, as numeric or as character index, or a vector of sample sizes.

labels Optionally, either the column in the dataframe containing labels, as numeric or as character index, or a vector of labels.

colorCol Optionally, either the column in the dataframe containing manually specified colours, as numeric or as character index, or a vector of manually specified colours.

conf.level The confidence of the confidence intervals.

xlab The label for the x axis.

outputFile A file to which to save the plot.

outputWidth, outputHeight Width and height of saved plot (specified in centimeters by default, see ggsaveParams).

ggsaveParams Parameters to pass to ggsave when saving the plot.

Additional arguments are passed to diamondPlot() and eventually to ggDiamondLayer(). This can be used to, for example, specify two or more colors to use to generate a gradient (using generateColors and maybe fullColorRange).

Value

A ggplot() plot with a ggDiamondLayer() is returned.

Author(s)

Gjalt-Jorn Peters

Maintainer: Gjalt-Jorn Peters gjalt-jorn@userfriendlyscience.com
multiResponse

Generate a table for multiple response questions

Description

The multiResponse function mimics the behavior of the table produced by SPSS for multiple response questions.
Usage

```r
multiResponse(
  data,
  items = NULL,
  regex = NULL,
  perlRegex = TRUE,
  endorsedOption = 1
)
```

Arguments

data: Dataframe containing the variables to display.
items, regex: Arguments `items` and `regex` can be used to specify which variables to process. `items` should contain the variable (column) names (or indices), and `regex` should contain a regular expression used to match to the column names of the dataframe. If none is provided, all variables in the dataframe are processed.
perlRegex: Whether to use the perl engine to match the regex.
endorsedOption: Which value represents the endorsed option (note that producing this kind of table requires dichotomous items, where each variable is either endorsed or not endorsed, so this is also a way to treat other variables as dichotomous).

Value

A dataframe with columns `Option`, `Frequency`, `Percentage`, and `Percentage of (X) cases`, where `X` is the number of cases.

Author(s)

Ananda Mahto; implemented in this package (and tweaked a bit) by Gjalt-Jorn Peters.
Maintainer: Gjalt-Jorn Peters gjalt-jorn@userfriendlyscience.com

References

This function is based on the excellent and extensive Stack Exchange answer by Ananda Mahto at https://stackoverflow.com/questions/9265003/analysis-of-multiple-response.

Examples

```r
multiResponse(mtcars, c('vs', 'am'))
```
**multiResponsejmv**  
*Multi Response*

**Description**  
Multi Response

**Usage**  
multiResponsejmv(data, items, endorsedOption = 1)

**Arguments**
- data .
- items .
- endorsedOption .

**Value**  
A results object containing:

\[
\text{results$table} \quad \text{a table}
\]

Tables can be converted to data frames with asDF or \texttt{as.data.frame}. For example:

\[
\text{results$table$asDF}
\]
\[
\text{as.data.frame(results$table)}
\]

---

**multiVarFreq**  
*Generate a table collapsing frequencies of multiple variables*

**Description**  
This function can be used to efficiently combine the frequencies of variables with the same possible values. The frequencies are collapsed into a table with the variable names as row names and the possible values as column (variable) names.

**Usage**  
multiVarFreq(data, items = NULL, labels = NULL, sortByMean = TRUE)
normalHist

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>data</td>
<td>The dataframe containing the variables.</td>
</tr>
<tr>
<td>items</td>
<td>The variable names.</td>
</tr>
<tr>
<td>labels</td>
<td>Labels can be provided which will be set as row names when provided.</td>
</tr>
<tr>
<td>sortByMean</td>
<td>Whether to sort the rows by mean value for each variable (only sensible if the possible values are numeric).</td>
</tr>
</tbody>
</table>

Value

The resulting dataframe, but with class ‘multiVarFreq’ prepended to allow pretty printing.

Author(s)

Gjalt-Jorn Peters
Maintainer: Gjalt-Jorn Peters gjalt-jorn@userfriendlyscience.com

See Also

table()

Examples

```r
multiVarFreq(mtcars, c('gear', 'carb'));
```

Description

normalHist generates a histogram with a density curve and a normal density curve.

Usage

```r
normalHist(
    vector,
    histColor = "#0000CC",
    distributionColor = "#0000CC",
    normalColor = "#00CC00",
    distributionLineSize = 1,
    normalLineSize = 1,
    histAlpha = 0.25,
    xLabel = NULL,
    yLabel = NULL,
    normalCurve = TRUE,
```
distCurve = TRUE,
breaks = 30,
theme = ggplot2::theme_minimal(),
rug = NULL,
jitteredRug = TRUE,
rugSides = "b",
rugAlpha = 0.2,
returnPlotOnly = FALSE
)

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

Arguments

vector A numeric vector.
histColor The colour to use for the histogram.
distributionColor The colour to use for the density curve.
normalColor The colour to use for the normal curve.
distributionLineSize The line size to use for the distribution density curve.
normalLineSize The line size to use for the normal curve.
histAlpha Alpha value ('opaqueness', as in, versus transparency) of the histogram.
xLabel Label to use on x axis.
yLabel Label to use on y axis.
normalCurve Whether to display the normal curve.
distCurve Whether to display the curve showing the distribution of the observed data.
brakes The number of breaks to use (this is equal to the number of bins minus one, or in other words, to the number of bars minus one).
theme The theme to use.
rug Whether to add a rug (i.e. lines at the bottom that correspond to individual datapoints).
jitteredRug Whether to jitter the rug (useful for variables with several datapoints sharing the same value).
rugSides This is useful when the histogram will be rotated; for example, this can be set to 'r' if the histogram is rotated 270 degrees.
rugAlpha Alpha value to use for the rug. When there is a lot of overlap, this can help get an idea of the number of datapoints at 'popular' values.
returnPlotOnly Whether to return the usual normalHist object that also contains all settings and intermediate objects, or whether to only return the ggplot2::ggplot() plot.
x The object to print.
... Any additional arguments are passed to the default print method.
Value

An object, with the following elements:

- **input**: The input when the function was called.
- **intermediate**: The intermediate numbers and distributions.
- **dat**: The dataframe used to generate the plot.
- **plot**: The histogram.

Author(s)

Gjalt-Jorn Peters
Maintainer: Gjalt-Jorn Peters gjalt-jorn@userfriendlyscience.com

Examples

```r
normalHist(mtcars$mpg)
```

---

**noZero**

*Remove one or more zeroes before the decimal point*

Description

Remove one or more zeroes before the decimal point

Usage

```r
noZero(str)
```

Arguments

- **str**: The character string to process.

Value

The processed string.

See Also

`formatCI()`, `formatR()`, `formatPvalue()`

Examples

```r
noZero("0.3");
```
opts

Options for the ufs package

Description
The ufs::opts object contains three functions to set, get, and reset options used by the ufs package. Use ufs::opts$set to set options, ufs::opts$get to get options, or ufs::opts$reset to reset specific or all options to their default values.

Usage
opts

Format
An object of class list of length 5.

Details
It is normally not necessary to get or set ufs options. The following arguments can be passed:

... For ufs::opts$set, the dots can be used to specify the options to set, in the format option = value, for example, tableOutput = c("console", "viewer"). For ufs::opts$reset, a list of options to be reset can be passed.

option For ufs::opts$set, the name of the option to set.
default For ufs::opts$get, the default value to return if the option has not been manually specified.

The following options can be set:

tableOutput Where to show some tables.

Examples

### Get the default columns in the variable view
ufs::opts$get("tableOutput");

### Set it to a custom version
ufs::opts$set(tableOutput = c("values", "level"));

### Check that it worked
ufs::opts$get("tableOutput");

### Reset this option to its default value
ufs::opts$reset("tableOutput");

### Check that the reset worked, too
ufs::opts$get("tableOutput");
## parallelSubscales

**Description**

Split a dataset into two parallel halves

**Usage**

```r
parallelSubscales(dat, convertToNumeric = TRUE)
```

```
# S3 method for class 'parallelSubscales'
print(x, digits = 2, ...)
```

**Arguments**

- `dat`: The dataframe
- `convertToNumeric`: Whether to first convert all columns to numeric
- `x`: The object to print
- `digits`: The number of digits to round to
- `...`: Ignored.

**Value**

A `parallelSubscales` object that contains the new data frames, and when printed shows the descriptive statistics; or, for the print function, `x`, invisibly.

## pomegaSq

**Description**

These functions use some conversion to and from the $F$ distribution to provide the Omega Squared distribution.

**Usage**

```r
pomegaSq(q, df1, df2, populationOmegaSq = 0, lower.tail = TRUE)
qomegaSq(p, df1, df2, populationOmegaSq = 0, lower.tail = TRUE)
romegaSq(n, df1, df2, populationOmegaSq = 0)
domegaSq(x, df1, df2, populationOmegaSq = 0)
```
Arguments

- df1, df2: Degrees of freedom for the numerator and the denominator, respectively.
- populationOmegaSq: The value of Omega Squared in the population; this determines the center of the Omega Squared distribution. This has not been implemented yet in this version of ufs. If anybody has the inverse of `convert.ncf.to.omegasq()` for me, I'll happily integrate this.
- lower.tail: logical; if TRUE (default), probabilities are the likelihood of finding an Omega Squared smaller than the specified value; otherwise, the likelihood of finding an Omega Squared larger than the specified value.
- p: Vector of probabilities (p-values).
- n: Desired number of Omega Squared values.
- x, q: Vector of quantiles, or, in other words, the value(s) of Omega Squared.

Details

The functions use `convert.omegasq.to.f()` and `convert.f.to.omegasq()` to provide the Omega Squared distribution.

Value

domegaSq gives the density, pomegaSq gives the distribution function, qomegaSq gives the quantile function, and romegaSq generates random deviates.

Author(s)

Gjalt-Jorn Peters

Maintainer: Gjalt-Jorn Peters gjalt-jorn@userfriendlyscience.com

See Also

- `convert.omegasq.to.f()`, `convert.f.to.omegasq()`, `df()`, `pf()`, `qf()`, `rf()`

Examples

```r
### Generate 10 random Omega Squared values
romegaSq(10, 66, 3);

### Probability of finding an Omega Squared value smaller than .06 if it's 0 in the population
pomegaSq(.06, 66, 3);
```
pwr.bootES

Estimate required sample size for accuracy in parameter estimation using bootES

Description

This function uses bootES::bootES() to compute

Usage

pwr.bootES(data = data, ci.type = "bca", ..., w = 0.1, silent = TRUE)

Arguments

data The dataset, as you would normally supply to bootES::bootES(); you will probably have to simulate this.

ci.type The estimation method; by default, the default of bootES::bootES() is used ('bca'), but this is changed to 'basic' if it encounters problems.

... Other options for bootES::bootES() (see that help page).

w The desired 'halfwidth' of the confidence interval.

silent Whether to provide a lot of information about progress ('FALSE') or not ('TRUE').

Value

A single numeric value (the sample size).

References


Examples

### This requires the bootES package
if (requireNamespace("bootES", quietly = TRUE)) {

### To estimate a mean
x <- rnorm(500, mean=8, sd=3);
pwr.bootES(data.frame(x=x),
           R=500,
           w=.5);

### To estimate a correlation (the 'effect.type' parameter is redundant here; with two columns in the data frame, computing the confidence interval for the Pearson correlation is the default)
y <- x+rnorm(500, mean=0, sd=5);
cor(x, y);
}
```
requiredN <-
pwr.bootES(data.frame(x=x,
    y=y),
    effect.type='r',
    R=500,
    w=.2);
print(requiredN);
### Compare to parametric confidence interval
### based on the computed required sample size
confIntR(r = cor(x, y),
    N = requiredN);
### Width of obtained confidence interval
print(round(diff(as.numeric(confIntR(r = cor(x, y),
    N = requiredN))), 2));
```
pwr.confIntR

Determine required sample size for a given confidence interval width for Pearson’s r

Description

This function computes how many participants you need if you want to achieve a confidence interval of a given width. This is useful when you do a study and you are interested in how strongly two variables are associated.

Usage

pwr.confIntR(r, w = 0.1, conf.level = 0.95)

Arguments

- **r**: The correlation you expect to find (confidence intervals for a given level of confidence get narrower as the correlation coefficient increases).
- **w**: The required half-width (or margin of error) of the confidence interval.
- **conf.level**: The level of confidence.

Value

The required sample size, or a vector or matrix of sample sizes if multiple correlation coefficients or required (half-)widths were supplied. The row and column names specify the r and w values to which the sample size in each cell corresponds. The confidence level is set as attribute to the resulting vector or matrix.

Author(s)

Douglas Bonett (UC Santa Cruz, United States), with minor edits by Murray Moinester (Tel Aviv University, Israel) and Gjalt-Jorn Peters (Open University of the Netherlands, the Netherlands).

Maintainer: Gjalt-Jorn Peters gjalt-jorn@userfriendlyscience.com

References

- Peters, G. J. Y. & Crutzen, R. (forthcoming) An easy and foolproof method for establishing how effective an intervention or behavior change method is: required sample size for accurate parameter estimation in health psychology.
See Also

pwr.confIntR

Examples

pwr.confIntR(c(.4, .6, .8), w=c(.1, .2));

pwr.omegasq

Power calculations for Omega Squared.

Description

This function uses pwr.anova.test from the pwr package in combination with convert.cohensf.to.omegasq and convert.omegasq.to.cohensf to provide power analyses for Omega Squared.

Usage

pwr.omegasq(
  k = NULL,
  n = NULL,
  omegasq = NULL,
  sig.level = 0.05,
  power = NULL,
  digits = 4
)

## S3 method for class 'pwr.omegasq'
print(x, digits = x$digits, ...)

Arguments

k
  The number of groups.

n
  The sample size.

omegasq
  The Omega Squared value.

sig.level
  The significance level (alpha).

power
  The power.

digits
  The number of digits desired in the output (4, the default, is quite high; but omega squared value tend to be quite low).

x
  The object to print.

...
  Additional arguments are ignored.

Details

This function was written to work similarly to the power functions in the pwr package.
Value

An `power.htest.ufs` object that contains a number of input and output values, most notably:

- **power**: The (specified or computed) power
- **n**: The (specified or computed) sample size in each group
- **sig.level**: The (specified or computed) significance level (alpha)
- **sig.level**: The (specified or computed) Omega Squared value
- **cohensf**: The computed value for the Cohen’s $f$ effect size measure

Author(s)

Gjalt-Jorn Peters & Peter Verboon

Maintainer: Gjalt-Jorn Peters gjalt-jorn@userfriendlyscience.com

See Also

- `pwr.anova.test`
- `convert.cohensf.to.omegasq`
- `convert.omegasq.to.cohensf`

Examples

```r
pwr.omegasq(omegasq=.06, k=3, power=.8)
```

Description

Simple wrapper for `remotes` functions that fail gracefully (well, don’t fail at all, just don’t do what they’re supposed to do) when there’s no internet connection.

Usage

```r
quietRemotesInstall(x, func, unloadNamespace = TRUE, dependencies = FALSE, upgrade = FALSE, quiet = TRUE, errorInvisible = TRUE, ...)
```

```r
quietGitLabUpdate()
```
x,
unloadNamespace = TRUE,
dependencies = FALSE,
upgrade = FALSE,
quiet = TRUE,
errorInvisible = TRUE,
...
)

Arguments

x The repository name (e.g. "r-packages/ufs")
func The remotes function to use
unloadNamespace Whether to first unload the relevant namespace
dependencies, upgrade Whether to install dependencies or upgrade
quiet Whether to suppress messages and warnings
errorInvisible Whether to suppress errors
... Additional arguments are passed on to the remotes function

Value

The result of the call to the remotes function

qVec

Convenience function to quickly copy-paste a vector

Description

Convenience function to quickly copy-paste a vector

Usage

qVec(x, fn = NULL)
qVecSum(x)

Arguments

x A string with numbers, separated by arbitrary whitespace.
fn An optional function to apply to the vector before returning it.

Value

The numeric vector or result of calling the function
**rbind_dfs**

Simple alternative for `rbind.fill` or `bind_rows`

**Description**
Simple alternative for `rbind.fill` or `bind_rows`

**Usage**

```r
rbind_dfs(x, y, clearRowNames = TRUE)
```

**Arguments**

- `x`: One dataframe
- `y`: Another dataframe
- `clearRowNames`: Whether to clear row names (to avoid duplication)

**Value**

The merged dataframe

**Examples**

```r
rbind_dfs(Orange, mtcars);
```

---

**rbind_df_list**

Bind lots of dataframes together rowwise

**Description**

Bind lots of dataframes together rowwise

**Usage**

```r
rbind_df_list(x)
```

**Arguments**

- `x`: A list of dataframes

**Value**

A dataframe
regrInfluential

Detecting influential cases in regression analyses

Description
This function combines a number of criteria for determining whether a datapoint is an influential case in a regression analysis. It then sum the criteria to compute an index of influentiality. A list of cases with an index of influentiality of 1 or more is then displayed, after which the regression analysis is repeated without those influential cases. A scattermatrix is also displayed, showing the density curves of each variable, and in the scattermatrix, points that are colored depending on how influential each case is.

Usage

regrInfluential(formula, data, createPlot = TRUE)

## S3 method for class 'regrInfluential'
print(x, headingLevel = 3, ...)

Arguments

formula The formula of the regression analysis.
data The data to use for the analysis.
createPlot Whether to create the scattermatrix (requires the GGally package to be installed).
x Object to print.
headingLevel The number of hash symbols to prepend to the heading.
... Additional arguments are passed on to the regr print function.

Value

A regrInfluential object, which, if printed, shows the influential cases, the regression analyses repeated without those cases, and the scatter matrix.

Author(s)

Gjalt-Jorn Peters & Marwin Snippe

Maintainer: Gjalt-Jorn Peters gjalt-jorn@userfriendlyscience.com

Examples

regrInfluential(mpg ~ hp, mtcars);
**repeatStr**

Repeat a string a number of times

**Description**

Repeat a string a number of times

**Usage**

```r
repeatStr(n = 1, str = " ")
```

**Arguments**

- `n`, `str` Normally, respectively the frequency with which to repeat the string and the string to repeat; but the order of the inputs can be switched as well.

**Value**

A character vector of length 1.

**Examples**

```r
### 10 spaces:
repStr(10);

### Three euro symbols:
repStr("\u20ac", 3);
```

---

**report**

Output report from results

**Description**

This method can be used to format results in a way that can directly be included in a report or manuscript.

**Usage**

```r
report(x, headingLevel = 3, quiet = TRUE, ...)
```

## Default S3 method:
```r
report(x, headingLevel = 3, quiet = TRUE, ...)
```
Arguments

\texttt{x} \hspace{1cm} \text{The object to show.}
\texttt{headingLevel} \hspace{1cm} \text{The level of the Markdown heading to provide; basically the number of hashes ("#") to prepend to the headings.}
\texttt{quiet} \hspace{1cm} \text{Passed on to \texttt{knitr::knit()} whether it should be chatty (FALSE) or quiet (TRUE).}
\texttt{...} \hspace{1cm} \text{Passed to the specific method; for the default method, this is passed to the print method.}

\begin{verbatim}
safeRequire

Description
Load a package, install if not available

Usage
safeRequire(packageName, mirrorIndex = NULL)

Arguments

\texttt{packageName} \hspace{1cm} \text{The package}
\texttt{mirrorIndex} \hspace{1cm} \text{The index of the mirror (1 is used if not specified)}

\begin{verbatim}
scaleDiagnosis

Description
scaleDiagnosis provides a number of diagnostics for a scale (an aggregative measure consisting of several items).

Usage
scaleDiagnosis(
  data = NULL,
  items = NULL,
  plotSize = 180,
  sizeMultiplier = 1,
  axisLabels = "none",
  scaleReliability.ci = FALSE,
  conf.level = 0.95,
  normalHist = TRUE,
  poly = TRUE,
)
## Methods

### S3 method for class 'scaleDiagnosis'

```r
print(x, digits = x$digits, ...)
```

### S3 method for class 'scaleDiagnosis'

```r
scaleDiagnosis_partial(
  x,
  headingLevel = x$input$headingLevel,
  quiet = TRUE,
  echoPartial = FALSE,
  partialFile = NULL,
  ...
)
```

### S3 method for class 'scaleDiagnosis'

```r
knit_print(
  x,
  headingLevel = x$headingLevel,
  quiet = TRUE,
  echoPartial = FALSE,
  partialFile = NULL,
  ...
)
```

## Arguments

- **data**: A dataframe containing the items in the scale. All variables in this dataframe will be used if items is NULL.
- **items**: If not NULL, this should be a character vector with the names of the variables in the dataframe that represent items in the scale.
- **plotSize**: Size of the final plot in millimeters.
- **sizeMultiplier**: Allows more flexible control over the size of the plot elements.
- **axisLabels**: Passed to ggpairs function to set axisLabels.
- **scaleReliability.ci**: TRUE or FALSE: whether to compute confidence intervals for Cronbach’s Alpha and Omega (uses bootstrapping function in MBESS, takes a while).
- **conf.level**: Confidence of confidence intervals for reliability estimates (if requested with scaleReliability.ci).
- **normalHist**: Whether to use the default ggpairs histogram on the diagonal of the scattermatrix, or whether to use the `normalHist()` version.
- **poly**: Whether to also request the estimates based on the polychoric correlation matrix when calling `scaleStructure()`.
digits
headingLevel
scaleName
...
x
quiet
echoPartial
partialFile

Details

Function to generate an object with several useful statistics and a plot to assess how the elements (usually items) in a scale relate to each other, such as Cronbach’s Alpha, omega, the Greatest Lower Bound, a factor analysis, and a correlation matrix.

Value

An object with the input and several output variables. Most notably:

- scaleReliability
  - The results of scaleReliability.

- pca
  - A Principal Components Analysis

- fa
  - A Factor Analysis

- describe
  - Descriptive statistics about the items

- scatterMatrix
  - A scattermatrix with histograms on the diagonal and correlation coefficients in the upper right half.

Author(s)

Gjalt-Jorn Peters
Maintainer: Gjalt-Jorn Peters gjalt-jorn@userfriendlyscience.com

Examples

```r
### Note: the 'not run' is simply because running takes a lot of time,
### but these examples are all safe to run!
## Not run:
### This will prompt the user to select an SPSS file
scaleDiagnosis();

### Generate a datafile to use
exampleData <- data.frame(item1=rnorm(100));
exampleData$item2 <- exampleData$item1+rnorm(100);
```
scaleStructure

exampleData$item3 <- exampleData$item1+rnorm(100);
exampleData$item4 <- exampleData$item2+rnorm(100);
exampleData$item5 <- exampleData$item2+rnorm(100);

### Use a selection of two variables
scaleDiagnosis(data=exampleData, items=c('item2', 'item4'));

### Use all items
scaleDiagnosis(data=exampleData);

## End(Not run)

### scaleStructure

scaleStructure function (which was originally called scaleReliability) computes a number of measures to assess scale reliability and internal consistency. Note that to compute omega, the MBESS and/or the psych packages need to be installed, which are suggested packages and therefore should be installed separately (i.e. won’t be installed automatically).

#### Description

The scaleStructure function (which was originally called scaleReliability) computes a number of measures to assess scale reliability and internal consistency. Note that to compute omega, the MBESS and/or the psych packages need to be installed, which are suggested packages and therefore should be installed separately (i.e. won’t be installed automatically).

#### Usage

```r
scaleStructure(
  data = NULL,
  items = "all",
  digits = 2,
  ci = TRUE,
  interval.type = "normal-theory",
  conf.level = 0.95,
  silent = FALSE,
  samples = 1000,
  bootstrapSeed = NULL,
  omega.psych = TRUE,
  omega.psych_nfactors = 3,
  omega.psych_flip = TRUE,
  poly = TRUE,
  suppressSuggestedPkgsMsg = FALSE,
  headingLevel = 3
)
```

## S3 method for class 'scaleStructure'

```r
print(x, digits = x$input$digits, ...)
```

```r
scaleStructure_partial(
  x,
```
headingLevel = x$input$headingLevel,
quiet = TRUE,
echoPartial = FALSE,
partialFile = NULL,
...
)

## S3 method for class 'scaleStructure'
knit_print(
  x,
  headingLevel = x$input$headingLevel,
  quiet = TRUE,
echoPartial = FALSE,
partialFile = NULL,
...
)

Arguments

*data*  
A dataframe containing the items in the scale. All variables in this dataframe will be used if items = 'all'. If dat is NULL, a the getData function will be called to show the user a dialog to open a file.

*items*  
If not 'all', this should be a character vector with the names of the variables in the dataframe that represent items in the scale.

*digits*  
Number of digits to use in the presentation of the results.

*ci*  
Whether to compute confidence intervals as well. This requires the suggested MBESS package, which has to be installed separately. If true, the method specified in interval.type is used. When specifying a bootstrapping method, this can take quite a while!

*interval.type*  
Method to use when computing confidence intervals. The list of methods is explained in the help file for ci.reliability in MBESS. Note that when specifying a bootstrapping method, the method will be set to normal-theory for computing the confidence intervals for the ordinal estimates, because these are based on the polychoric correlation matrix, and raw data is required for bootstrapping.

*conf.level*  
The confidence of the confidence intervals.

*silent*  
If computing confidence intervals, the user is warned that it may take a while, unless silent=TRUE.

*samples*  
The number of samples to compute for the bootstrapping of the confidence intervals.

*bootstrapSeed*  
The seed to use for the bootstrapping - setting this seed makes it possible to replicate the exact same intervals, which is useful for publications.

*omega.psych*  
Whether to also compute the interval estimate for omega using the omega function in the psych package. The default point estimate and confidence interval for omega are based on the procedure suggested by Dunn, Baguley & Brunsden (2013) using the MBESS function ci.reliability (because it has more options for computing confidence intervals, not always requiring bootstrapping),
whereas the psych package point estimate was suggested in Revelle & Zinbarg (2008). The psych estimate usually (perhaps always) results in higher estimates for omega.

**omega.psych_nfactors**
The number of factor to use in the factor analysis when computing Omega. The default in `psych::omega()` is 3; to obtain the same results as in jamovi's "Reliability", set this to 1.

**omega.psych_flip**
Whether to let psych automatically flip items with negative correlations. The default in `psych::omega()` is TRUE; to obtain the same results as in jamovi's "Reliability", set this to FALSE.

**poly**
Whether to compute ordinal measures (if the items have sufficiently few categories).

**suppressSuggestedPkgsMsg**
Whether to suppress the message about the suggested MBESS and psych packages.

**headingLevel**
The level of the Markdown heading to provide; basically the number of hashes ("#") to prepend to the headings.

**x**
The object to print

... Any additional arguments for the default print function.

**quiet**
Passed on to `knitr::knit()` whether it should b chatty (FALSE) or quiet (TRUE).

**echoPartial**
Whether to show the executed code in the R Markdown partial (TRUE) or not (FALSE).

**partialFile**
This can be used to specify a custom partial file. The file will have object x available, which is the result of a call to `scaleStructure()`.

### Details
If you use this function in an academic paper, please cite Peters (2014), where the function is introduced, and/or Crutzen & Peters (2015), where the function is discussed from a broader perspective. This function is basically a wrapper for functions from the psych and MBESS packages that compute measures of reliability and internal consistency. For backwards compatibility, in addition to `scaleStructure`, `scaleReliability` can also be used to call this function.

### Value
An object with the input and several output variables. Most notably:

**input**
Input specified when calling the function

**intermediate**
Intermediate values and objects computed to get to the final results

**output**
Values of reliability / internal consistency measures, with as most notable elements:

**output$dat**
A dataframe with the most important outcomes

**output$omega**
Point estimate for omega

**output$glb**
Point estimate for the Greatest Lower Bound
output$alpha  Point estimate for Cronbach's alpha
output$coefficientH  Coefficient H
output$omega.ci  Confidence interval for omega
output$alpha.ci  Confidence interval for Cronbach's alpha

Author(s)
Gjalt-Jorn Peters and Daniel McNeish (University of North Carolina, Chapel Hill, US).
Maintainer: Gjalt-Jorn Peters gjalt-jorn@userfriendlyscience.com

References


See Also
psych::omega(), psych::alpha(), and MBESS::ci.reliability().

Examples

```r
## Not run:
### (These examples take a lot of time, so they are not run
```
## Description

`scatterMatrix` produces a matrix with jittered scatterplots, histograms, and correlation coefficients.

## Usage

```r
scatterMatrix(
  dat,            # data frame
  items = NULL,   # character vector of item names
  itemLabels = NULL, # character vector of item labels
  plotSize = 180,  # plot size
)```

### Example Usage

```r
## during testing.)
## This will prompt the user to select an SPSS file
scaleStructure();

## Load data from simulated dataset testRetestSimData (which
## satisfies essential tau-equivalence).
data(testRetestSimData);

## Select some items in the first measurement
exampleData <- testRetestSimData[2:6];

## Use all items (don't order confidence intervals to save time
## during automated testing of the example)
ufs::scaleStructure(dat=exampleData, ci=FALSE);

## Use a selection of three variables (without confidence
## intervals to save time
ufs::scaleStructure(  
  dat=exampleData,  
  items=c("t0_item2", "t0_item3", "t0_item4"),  
  ci=FALSE
);

## Make the items resemble an ordered categorical (ordinal) scale
ordinalExampleData <- data.frame(apply(exampleData, 2, cut,  
  breaks=5, ordered_result=TRUE,  
  labels=as.character(1:5)));

## Now we also get estimates assuming the ordinal measurement level
ufs::scaleStructure(ordinalExampleData, ci=FALSE);

## End(Not run)
```
sizeMultiplier = 1,
pointSize = 1,
axisLabels = "none",
normalHist = TRUE,
progress = NULL,
theme = ggplot2::theme_minimal(),
hideGrid = TRUE,
conf.level = 0.95,
...)

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

Arguments

dat          A dataframe containing the items in the scale. All variables in this dataframe
              will be used if items is NULL.
items        If not NULL, this should be a character vector with the names of the variables
              in the dataframe that represent items in the scale.
itemLabels   Optionally, labels to use for the items (optionally, named, with the names corre-
              sponding to the items; otherwise, the order of the labels has to match the order
              of the items)
plotSize     Size of the final plot in millimeters.
sizeMultiplier Allows more flexible control over the size of the plot elements
pointSize    Size of the points in the scatterplots
axisLabels   Passed to ggpairs function to set axisLabels.
normalHist   Whether to use the default ggpairs histogram on the diagonal of the scatterma-
              trix, or whether to use the normalHist() version.
progress     Whether to show a progress bar; set to FALSE to disable. See GGally::ggpairs() help
              for more information.
theme        The ggplot2 theme to use.
hideGrid     Whether to hide the gridlines in the plot.
conf.level   The confidence level of confidence intervals
...          Additional arguments for scatterMatrix() are passed on to normalHist(),
              and additional arguments for the print method are passed on to the default
              print method.
x            The object to print.

Value

An object with the input and several output variables. Most notably:

output$scatterMatrix

A scattermatrix with histograms on the diagonal and correlation coefficients in
the upper right half.
setCaptionNumberingKnitrHook

Author(s)
Gjalt-Jorn Peters
Maintainer: Gjalt-Jorn Peters gjalt-jorn@userfriendlyscience.com

Examples

```r
### Note: the 'not run' is simply because running takes a lot of time,
### but these examples are all safe to run!
## Not run:
### Generate a datafile to use
exampleData <- data.frame(item1=rnorm(100));
exampleData$item2 <- exampleData$item1+rnorm(100);
exampleData$item3 <- exampleData$item1+rnorm(100);
exampleData$item4 <- exampleData$item2+rnorm(100);
exampleData$item5 <- exampleData$item2+rnorm(100);
### Use all items
scatterMatrix(dat=exampleData);
## End(Not run)
```

Description
Set a knitr hook to automatically number captions for, e.g., figures and tables. `setCaptionNumberingKnitrHook()` is the general purpose function; you normally use `setFigCapNumbering()` or `setTabCapNumbering()`.

Usage

```r
setCaptionNumberingKnitrHook(
  captionName = "fig.cap",
  prefix = "Figure %s: ",
  suffix = "",
  optionName = paste0("setCaptionNumbering_", captionName),
  resetCounterTo = 1
)
```

```r
setFigCapNumbering(
  captionName = "fig.cap",
  prefix = "figure %s: ",
  suffix = "",
  optionName = paste0("setCaptionNumbering_", captionName),
)```
resetCounterTo = 1
)

setTabCapNumbering(
  captionName = "tab.cap",
  prefix = "Table \%s: ",
  suffix = "",
  optionName = paste0("setCaptionNumbering \_", captionName),
  resetCounterTo = 1
)

Arguments

  captionName   The name of the caption; for example, fig.cap or tab.cap.
  prefix, suffix The prefix and suffix; any occurrences of \%s will be replaced by the number.
  optionName   The name to use for the option that keeps track of the numbering.
  resetCounterTo Whether to reset the counter (as stored in the options), and if so, to what value
                   (set to FALSE to prevent resetting).

Value

  NULL, invisibly.

Examples

  ### To start automatically numbering figure captions
  setFigCapNumbering();

  ### To start automatically numbering table captions
  setTabCapNumbering();

sharedSubString   sharedSubString

Description

  A function to find the longest shared substring in a character vector.

Usage

  sharedSubString(x, y = NULL)

Arguments

  x   The character vector to process.
  y   Optionally, two single values can be specified. This is probably not useful to
       end users, but it’s used by the function when it calls itself.
**Value**

A vector of length one with either the longest substring that occurs in all values of the character vector, or NA if no overlap an be found.

**Author(s)**

Gjalt-Jorn Peters

Maintainer: Gjalt-Jorn Peters gjalt-jorn@userfriendlyscience.com

**Examples**

```r
sharedSubString(c("t0_responseTime", "t1_responseTime", "t2_responseTime"));
### Returns ".responseTime"
```

---

**simDataSet**

*Simulate a dataset*

**Description**

simDataSet can be used to conveniently and quickly simulate a dataset that satisfies certain constraints, such as a specific correlation structure, means, ranges of the items, and measurement levels of the variables. Note that the results are approximate; mvrnorm is used to generate the correlation matrix, but the factor are only created after that, so cutting the variable into factors may change the correlations a bit.

**Usage**

```r
simDataSet(
  n,
  varNames,
  correlations = c(0.1, 0.4),
  specifiedCorrelations = NULL,
  means = 0,
  sds = 1,
  ranges = c(1, 7),
  factors = NULL,
  cuts = NULL,
  labels = NULL,
  seed = 20160503,
  empirical = TRUE,
  silent = FALSE
)
```
Arguments

n  Number of requires cases (records, entries, participants, rows) in the final dataset.

varNames  Names of the variables in a vector; note that the length of this vector will determine the number of variables simulated.

correlations  The correlations between the variables are randomly sampled from this range using the uniform distribution; this way, it’s easy to have a relatively ‘messy’ correlation matrix without the need to specify every correlation manually.

specifiedCorrelations

The correlations that have to have a specific value can be specified here, as a list of vectors, where each vector’s first two elements specify variables names, and the last one the correlation between those two variables. Note that tweaking the correlations may take some time; the MASS::mvrnorm() function will complain that "'Sigma' is not positive definite", or in other words, you supplied a combination of correlations that can’t exist simultaneously, if you get it wrong.

means, sds  The means and standard deviations of the variables. Note that is you set ranges for one or more variables (see below), those ranges are used to rescale those variables, overriding any specified means and standard deviations. If only one mean or standard deviation is supplied, it’s recycled along the variables.

ranges  The desired ranges of the variables, supplied as a named list where the name of each element corresponds to a variable. The scales::rescale() function will be used to rescale those variables for which a desired scale is specified here. Note that for those variables, the means and standard deviations will be determined by these new ranges.

factors  A vector of variable names that should be converted into factors (using base::cut()). Make sure to specify lists for cuts and labels as well (of the same length).

cuts  A list of vectors that specify, for each factor, where to 'cut' the numeric vector into factor levels.

labels  A list of vectors that specify, for each factor, and for each level, the labels that should be assigned to the factor levels. Each vector in this list has to have one more element than each vector in the cuts list.

seed  The seed to use when generating the dataset (to make sure the exact same dataset can be generated repeatedly).

empirical  Whether to generate the data using the exact empirical = TRUE or approximate (empirical = FALSE) correlation matrix; this is passed on to MASS::mvrnorm().

silent  Whether to show intermediate and final descriptive information (correlation and covariance matrices as well as summaries).

Details

This function was intended to allow relatively quick generation of datasets that satisfy specific constraints, e.g. including a number of factors, variables with a specified minimum and maximum value or specified means and standard deviations, and of course specific correlations. Because all correlations except those specified are randomly generated from a uniform distribution, it’s quite convenient to generate messy kind of real looking datasets quickly. Note that it’s mostly a convenience function, and datasets will still require tweaking; for example, factors are simply numeric...
vectors that are `cut()` after `MASS::mvrnorm()` generated the data, so the associations will change slightly.

**Value**

The generated dataframe is returned invisibly.

**Examples**

```r
dat <- simDataSet(
  500,
  varNames=c('age',
    'sex',
    'educationLevel',
    'negativeLifeEventsInPast10Years',
    'problemCoping',
    'emotionCoping',
    'resilience',
    'depression'),
  means = c(40,
    0,
    0,
    5,
    3.5,
    3.5,
    3.5,
    3.5),
  sds = c(10,
    1,
    1,
    1.5,
    1.5,
    1.5,
    1.5,
    1.5),
  specifiedCorrelations =
    list(c('problemCoping', 'emotionCoping', -.5),
    c('problemCoping', 'resilience', .5),
    c('problemCoping', 'depression', -.4),
    c('depression', 'emotionCoping', .6),
    c('depression', 'resilience', -.3)),
  ranges = list(age = c(18, 54),
    negativeLifeEventsInPast10Years = c(0,8),
    problemCoping = c(1,7),
    emotionCoping = c(1,7)),
  factors=c("sex", "educationLevel"),
  cuts=list(c(0),
    c(-.5, .5)),
  labels=list(c('female', 'male'),
    c('lower', 'middle', 'higher')),
  silent=FALSE);
```
**spearmanBrown**  
*Spearmann-Brown formula*

**Description**
Spearmann-Brown formula

**Usage**
```r
spearmanBrown(nrOfItems, itemReliability)
spearmanBrown_reversed(nrOfItems, scaleReliability)
spearmanBrown_requiredLength(scaleReliability, itemReliability)
```

**Arguments**
- `nrOfItems` Number of items (or 'subtests') in the scale (or 'test').
- `itemReliability` The reliability of one item (or 'subtest').
- `scaleReliability` The reliability of the scale (or, desired reliability of the scale).

**Value**
For `spearmanBrown`, the predicted scale reliability; for `spearmanBrown_requiredLength`, the number of items required to achieve the desired scale reliability; and for `spearmanBrown_reversed`, the reliability of one item.

**Examples**
```r
spearmanBrown(10, .4);
spearmanBrown_reversed(10, .87);
spearmanBrown_requiredLength(.87, .4);
```

---

**strToFilename**  
*Convert a string to a safe filename*

**Description**
Convert a string to a safe filename

**Usage**
```r
strToFilename(str, ext = NULL)
```
suspectParticipants

Arguments

str The string to convert.

ext Optionally, an extension to append.

Value

The string, processed to remove potentially problematic characters.

Examples

strToFilename("this contains: illegal characters, spaces, et cetera.");

suspectParticipants Selects suspect participants from a carelessObject

Description

This function is a wrapper for the carelessObject() function, which wraps a number of functions from the careless package. Normally, you’d probably call carelessReport which calls this function to generate a report of suspect participants.

Usage

suspectParticipants(
  carelessObject,
  nFlags = 1,
  digits = 2,
  missingSymbol = "Missing"
)

Arguments

carelessObject The result of the call to carelessObject().
nFlags The number of flags required to be considered suspect.
digits The number of digits to round to.
missingSymbol How to represent missing values.

Value

A logical vector.

Examples

suspectParticipants(carelessObject(mtcars),
  nFlags = 2);
testRetestAlpha

Test-Retest Alpha Coefficient

Description

The testRetestAlpha function computes the test-retest alpha coefficient (Green, 2003).

Usage

testRetestAlpha(
  dat = NULL,
  moments = NULL,
  testDat = NULL,
  retestDat = NULL,
  sortItems = FALSE,
  convertToNumeric = TRUE
)

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

Arguments

dat A dataframe containing the items in the scale at both measurement moments. If no dataframe is specified, a dialogue will be launched to allow the user to select an SPSS datafile. If only one dataframe is specified, either the items have to be ordered chronologically (i.e. first all items for the first measurement, then all items for the second measurement), or the vector 'moments' has to be used to indicate, for each item, to which measurement moment it belongs.

moments Used to indicate to which measurement moment each item in 'dat' belongs; should be a vector with the same length as dat has columns, and with two possible values (e.g. 1 and 2).

testDat, retestDat Dataframes with the items for each measurement moment: note that the items have to be in the same order (unless sortItems is TRUE).

sortItems If true, the columns (items) in each dataframe are ordered alphabetically before starting. This can be convenient to ensure that the order of the items at each measurement moment is the same.

convertToNumeric When TRUE, the function will attempt to convert all vectors in the dataframes to numeric.

x The object to print

... Ignored.
Value

An object with the input and several output variables. Most notably:

input
Input specified when calling the function

intermediate
Intermediate values and objects computed to get to the final results

output$testRetestAlpha
The value of the test-retest alpha coefficient.

References


Examples

```r
## Not run:
### This will prompt the user to select an SPSS file
testRetestAlpha();

## End(Not run)

### Load data from simulated dataset testRetestSimData (which
### satisfies essential tau-equivalence).
data(testRetestSimData);

### The first column is the true score, so it's excluded in this example.
exampleData <- testRetestSimData[, 2:ncol(testRetestSimData)];

### Compute test-retest alpha coefficient
testRetestAlpha(exampleData);
```

testRetestCES

Test-Retest Coefficient of Equivalence & Stability

Description

The testRetestCES function computes the test-retest Coefficient of Equivalence and Stability (Schmidt, Le & Ilies, 2003).

Usage

```r
testRetestCES(
  dat = NULL,
  moments = NULL,
  testDat = NULL,
  retestDat = NULL,
)```
## S3 method for class 'testRetestCES'
print(x, digits = x$input$digits, ...)

Arguments

dat A dataframe. For testRetestCES, this dataframe must contain the items in the scale at both measurement moments. If no dataframe is specified, a dialogue will be launched to allow the user to select an SPSS datafile. If only one dataframe is specified, either the items have to be ordered chronologically (i.e. first all items for the first measurement, then all items for the second measurement), or the vector 'moments' has to be used to indicate, for each item, to which measurement moment it belongs. The number of columns in this dataframe MUST be even! Note that instead of providing this dataframe, the items of each measurement moment can be provided separately in testDat and retestDat as well.

moments Used to indicate to which measurement moment each item in 'dat' belongs; should be a vector with the same length as dat has columns, and with two possible values (e.g. 1 and 2).

testDat, retestDat Dataframes with the items for each measurement moment: note that the items have to be in the same order (unless sortItems is TRUE).

parallelTests A vector indicating which items belong to which parallel test; like the moments vector, this should have two possible values (e.g. 1 and 2). Alternatively, it can be character value with 'means' or 'variances'; in this case, parallelSubscales will be used to create roughly parallel halves.

sortItems If true, the columns (items) in each dataframe are ordered alphabetically before starting. This can be convenient to ensure that the order of the items at each measurement moment is the same.

convertToNumeric When TRUE, the function will attempt to convert all vectors in the dataframes to numeric.

digits Number of digits to print.

x The object to print

... Ignored.

Details

This function computes the test-retest Coefficient of Equivalence and Stability (CES) as described in Schmidt, Le & Ilies (2003). Note that this function only computes the test-retest CES for a scale that is administered twice and split into two parallel halves post-hoc (this procedure is explained on page 210, and the equations that are used, 16 and 17a are explained on page 212).
Value

An object with the input and several output variables. Most notably:

input
Input specified when calling the function

intermediate
Intermediate values and objects computed to get to the final results

output$testRetestCES
The value of the test-retest Coefficient of Equivalence and Stability.

Note

This function uses equations 16 and 17 on page 212 of Schmidt, Le & Ilies (2003): in other words, this function assumes that one scale is administered twice. If you’d like the computation for two different but parallel scales/measures to be implemented, please contact me.

References


Examples

## Not run:
### This will prompt the user to select an SPSS file
testRetestCES();

## End(Not run)

### Load data from simulated dataset testRetestSimData (which ### satisfies essential tau-equivalence).
data(testRetestSimData);

### The first column is the true score, so it's excluded in this example.
exampleData <- testRetestSimData[, 2:ncol(testRetestSimData)];

### Compute test-retest alpha coefficient
testRetestCES(exampleData);

Description

The testRetestReliability function is a convenient interface to testRetestAlpha and testRetestCES.
testRetestReliability

Usage

testRetestReliability(
  dat = NULL,
  moments = NULL,
  testDat = NULL,
  retestDat = NULL,
  parallelTests = "means",
  sortItems = FALSE,
  convertToNumeric = TRUE,
  digits = 2
)

## S3 method for class 'testRetestReliability'
print(x, digits = x$in$ digits, ...)

Arguments

dat A dataframe. This dataframe must contain the items in the scale at both measurement moments. If no dataframe is specified, a dialogue will be launched to allow the user to select an SPSS datafile. If only one dataframe is specified, either the items have to be ordered chronologically (i.e. first all items for the first measurement, then all items for the second measurement), or the vector 'moments' has to be used to indicate, for each item, to which measurement moment it belongs. The number of columns in this dataframe MUST be even! Note that instead of providing this dataframe, the items of each measurement moment can be provided separately in testDat and retestDat as well.
moments Used to indicate to which measurement moment each item in 'dat' belongs; should be a vector with the same length as dat has columns, and with two possible values (e.g. 1 and 2).
testDat, retestDat Dataframes with the items for each measurement moment: note that the items have to be in the same order (unless sortItems is TRUE).
parallelTests A vector indicating which items belong to which parallel test; like the moments vector, this should have two possible values (e.g. 1 and 2). Alternatively, it can be character value with 'means' or 'variances'; in this case, parallelSubscales will be used to create roughly parallel halves.
sortItems If true, the columns (items) in each dataframe are ordered alphabetically before starting. This can be convenient to ensure that the order of the items at each measurement moment is the same.
convertToNumeric When TRUE, the function will attempt to convert all vectors in the dataframes to numeric.
digits Number of digits to show when printing the output
x The object to print
... Passed on to the print function
Details

This function calls both testRetestAlpha and testRetestCES to compute and print measures of the test-retest reliability.

Value

An object with the input and several output variables. Most notably:

- **input**: Input specified when calling the function
- **intermediate**: Intermediate values and objects computed to get to the final results
- **output$testRetestAlpha**: The value of the test-retest alpha coefficient.
- **output$testRetestCES**: The value of the test-retest Coefficient of Equivalence and Stability.

Examples

```r
## Not run:
### This will prompt the user to select an SPSS file
testRetestReliability();

## End(Not run)

### Load data from simulated dataset testRetestSimData (which
### satisfies essential tau-equivalence).
data(testRetestSimData);

### The first column is the true score, so it's excluded in this example.
exampleData <- testRetestSimData[, 2:ncol(testRetestSimData)];

### Compute test-retest alpha coefficient
ufs::testRetestReliability(exampleData);
```

---

**testRetestSimData**

*testRetestSimData* is a simulated dataframe used to demonstrate the *testRetestAlpha* coefficient function.

Description

This dataset contains the true scores of 250 participants on some variable, and 10 items of a scale administered twice (at t0 and at t1).
Format

A data frame with 250 observations on the following 21 variables.

- **trueScore** The true scores
- **t0_item1** Score on item 1 at test
- **t0_item2** Score on item 2 at test
- **t0_item3** Score on item 3 at test
- **t0_item4** Score on item 4 at test
- **t0_item5** Score on item 5 at test
- **t0_item6** Score on item 6 at test
- **t0_item7** Score on item 7 at test
- **t0_item8** Score on item 8 at test
- **t0_item9** Score on item 9 at test
- **t0_item10** Score on item 10 at test
- **t1_item1** Score on item 1 at retest
- **t1_item2** Score on item 2 at retest
- **t1_item3** Score on item 3 at retest
- **t1_item4** Score on item 4 at retest
- **t1_item5** Score on item 5 at retest
- **t1_item6** Score on item 6 at retest
- **t1_item7** Score on item 7 at retest
- **t1_item8** Score on item 8 at retest
- **t1_item9** Score on item 9 at retest
- **t1_item10** Score on item 10 at retest

Details

This dataset was generated with the code in the reliabilityTest.r test script.

Author(s)

Gjalt-Jorn Peters
Maintainer: Gjalt-Jorn Peters gjalt-jorn@userfriendlyscience.com

Examples

data(testRetestSimData);
head(testRetestSimData);
hist(testRetestSimData$t0_item1);
cor(testRetestSimData);
vecTxt

Easily parse a vector into a character value

Description

vecTxtQ, vecTxtB, and vecTxtM and are convenience functions with default quotes that can be useful when working in R Markdown documents.

Usage

vecTxt(
  vector,  
  delimiter = "", 
  useQuote = "", 
  firstDelimiter = NULL, 
  lastDelimiter = " & ", 
  firstElements = 0, 
  lastElements = 1, 
  lastHasPrecedence = TRUE 
)

vecTxtQ(vector, useQuote = "'", ...)  
vecTxtB(vector, useQuote = "\", ...)  
vecTxtM(vector, useQuote = "$", ...)

Arguments

vector
  The vector to process.
delimiter, firstDelimiter, lastDelimiter
  The delimiters to use for respectively the middle, first firstElements, and last lastElements elements.
useQuote
  This character string is pre- and appended to all elements; so use this to quote all elements (useQuote="'"), doublequote all elements (useQuote="\"), or anything else (e.g. useQuote='|'). The only difference between vecTxt and vecTxtQ is that the latter by default quotes the elements.
firstElements, lastElements
  The number of elements for which to use the first respective last delimiters
lastHasPrecedence
  If the vector is very short, it's possible that the sum of firstElements and lastElements is larger than the vector length. In that case, downwardly adjust the number of elements to separate with the first delimiter (TRUE) or the number of elements to separate with the last delimiter (FALSE)?
...
  Any addition arguments to vecTxtQ are passed on to vecTxt.
Value
A character vector of length 1.

Examples
vecTxtQ(names(mtcars));

---

viridisPalette
Convenience function to get 2-7 color viridis palettes

Description
This function only exists to avoid importing the viridis package.

Usage
viridisPalette(x)

Arguments
x The number of colors you want (seven at most).

Value
A vector of colours.

---

wrapVector
Wrap all elements in a vector

Description
Wrap all elements in a vector

Usage
wrapVector(x, width = 0.9 * getOption("width"), sep = "\n", ...)

Arguments
x The character vector
width The number of
sep The glue with which to combine the new lines
... Other arguments are passed to strwrap().
Value
A character vector

Examples
```
res <- wrapVector(
  c(
    "This is a sentence ready for wrapping",
    "So is this one, although it's a bit longer"
  ),
  width = 10
);

print(res);
cat(res, sep="\n");
```

---

**zotero_construct_export_call**

*Construct the URL for a Zotero export call*

**Description**
This function is just a convenience function to create a simple URL to download references from a public Zotero group. See https://www.zotero.org/support/dev/web_api/v3/start for details.

**Usage**
```
zotero_construct_export_call(
  group,
  sort = "dateAdded",
  direction = "asc",
  format = "bibtex",
  start = 0,
  limit = 100
)
```

**Arguments**
- `group` The group ID
- `sort` On which field to sort
- `direction` The direction to sort in
- `format` The format to export
- `start` The index of the first record to return
- `limit` The number of records to return
zotero_download_and_export_items

Value

The URL in a character vector.

Examples

```r
zotero_construct_export_call(2425237);
```

Description

Download and save all items in a public Zotero group

Usage

```r
zotero_download_and_export_items(
  group,
  file,
  format = "bibtex",
  showKeys = TRUE
)
```

Arguments

- `group`: The group ID
- `file`: The filename to write to
- `format`: The format to export
- `showKeys`: Whether to show the keys

Value

The bibliography as a character vector

Examples

```r
## Not run:
tmpFile <- tempfile(fileext = "bib");
zotero_download_and_export_items(
  2425237,
  tmpFile
);
writtenBibliography <- readLines(tmpFile);
writtenBibliography[1:7];
## End(Not run)
```
**zotero_get_all_items**  Get all items in a public Zotero group

Description
----------------
Get all items in a public Zotero group

Usage
----------------
`zotero_get_all_items(group, format = "bibtex")`

Arguments
----------------
group  The group ID
format  The format to export

Value
----------------
A character vector

Examples
----------------
`zotero_get_all_items(2425237);`

**zotero_nr_of_items**  Get number of items in a public Zotero group

Description
----------------
Get number of items in a public Zotero group

Usage
----------------
`zotero_nr_of_items(group)`

Arguments
----------------
group  The group ID

Value
----------------
The number of items as a numeric vector.

Examples
----------------
`zotero_nr_of_items(2425237);`
%IN%  

Case insensitive version of %in%

**Description**

This is simply 'in', but applies `base::toupper()` to both arguments, first.

**Usage**

```r
find %IN% table
```

**Arguments**

- **find**  The element(s) to look up in the vector or matrix.
- **table**  The vector or matrix in which to look up the element(s).

**Value**

A logical vector.

**Examples**

```r
letters[1:4] %IN% LETTERS
```
Index

* **bivar**
  computeStatistic_t, 32
  confIntOmegaSq, 35
  crammersV, 44
* **character**
  escapeRegex, 62
  sharedSubString, 134
* **datasets**
  opts, 112
  testRetestSimData, 145
* **data**
  bfi-data, 17
* **file**
  getData, 76
* **graphs**
  ggProportionPlot, 82
* **hplot**
  associationsDiamondPlot, 10
  biAxisDiamondPlot, 17
  diamondCoordinates, 52
  diamondPlot, 55
  duoComparisonDiamondPlot, 58
  factorLoadingDiamondCIplot, 68
  factorLoadingHeatmap, 70
  ggBoxplot, 78
  ggEasyBar, 79
  ggProportionPlot, 82
  ggqq, 85
  meansDiamondPlot, 101
  meanSDtoDiamondPlot, 104
* **htest**
  confIntProp, 36
  confIntR, 37
  pwr.confIntR, 117
  pwr.omegasq, 118
* **manip**
  escapeRegex, 62
* **programming**
  escapeRegex, 62
* **univariate**
  descr, 49
  invrItem, 91
  iqrOutlier, 92
* **univar**
  associationMatrix, 7
  cohensdCI, 29
  confIntProp, 36
  getData, 76
  pomegasq, 113
  scaleDiagnosis, 124
  scaleStructure, 127
  scatterMatrix, 131
  testRetestCES, 141
* **utilities**
  areColors, 5
  associationMatrix, 7
  checkDataIntegrity, 24
  computeStatistic_t, 32
  convert, 40
  convert.cer.to.d, 41
  dataShape, 45
  findShortestInterval, 72
  getData, 76
  knitFig, 97
  multiResponse, 106
  multiVarFreq, 108
  normalHist, 109
  scaleDiagnosis, 124
  scaleStructure, 127
  scatterMatrix, 131
  testRetestCES, 141
* **utils**
  extractVarName, 66
  %IN%, 152

A_VarghaDelaney, 14
aipedjmv, 4
aiperjmv, 5
areColors, 5
INDEX

arr, 6
as.data.frame, 108
as.data.frame(), 51
as.data.frame(descr)(descr), 49
associationMatrix, 7
associationMatrixESDefaults
  (computeStatistic_t), 32
associationMatrixStat Defaults
  (computeStatistic_t), 32
associationsDiamondPlot, 10
associationsDiamondPlot(), 19, 61
associationsToDiamondPlotDf
  (associationsDiamondPlot), 10
attenuate.d, 13
attenuate.r, 14
BAC_plot, 15
base::cat(), 98
base::cut(), 136
base::sprintf(), 90
base::toupper(), 152
bfi (bfi-data), 17
bfi-data, 17
biAxisDiamondPlot, 17
biDimColors, 20
binom.test().36, 37, 84
bootES::bootES(), 115
carelessObject, 21
carelessObject(), 139
carelessReport, 22
cat, 23, 98
cat(), 97, 98
cat0, 23
checkDataIntegrity, 24
checkPKgs, 26
CIM, 27
CIM_partial(CIM), 27
cohensdCI, 29
cohensdDistribution(cohensdCI), 29
col2rgb(), 28, 53
colnames(), 38
colorRamp(), 28, 53
Comparison, 24
computeEffectSize_d
  (computeStatistic_t), 32
computeEffectSize_etasq
  (computeStatistic_t), 32
computeEffectSize_omegasq
  (computeStatistic_t), 32
computeEffectSize_r
  (computeStatistic_t), 32
computeEffectSize_v
  (computeStatistic_t), 32
computeStatistic_chisq
  (computeStatistic_t), 32
computeStatistic_f
  (computeStatistic_t), 32
computeStatistic_r
  (computeStatistic_t), 32
computeStatistic_t, 32
confIntD (cohensdCI), 29
confIntDjmv, 34
confIntOmegaSq, 35
confIntProp, 36
confIntProp(), 84, 116
confIntR, 37
confIntR(), 38
confIntRjmv, 39
confIntSD, 39
confIntV (cramersV), 44
correct, 40
correct.cer.to.d, 41
correct.d (cohensf.to.omegasq), 118, 119
correct.d.to.eer (convert.cer.to.d), 41
correct.d.to.nnc (convert.cer.to.d), 41
correct.d.to.t(), 31
correct.d.to.U3, 43
correct.eer.to.d (convert.cer.to.d), 41
correct.f (cohensf.to.omegasq()), 114
correct.ncf.to.omegasq(), 35, 36, 114
correct.omegasq.to.cohensf, 118, 119
correct.omegasq.to.f(), 114
correct.t.to.d(), 31
correctToNumeric, 44
cramersV, 44
cut(), 137
dataShape, 45
dCohensd (cohensdCI), 29
dd (cohensdCI), 29
descr, 49
descriptives(descr), 49
df(), 114
diamondCoordinates, 52
diamondPlot, 55
INDEX

pwr.omegasq, 118
qCohensd (cohensdCI), 29
qd (cohensdCI), 29
qf(), 114
qnorm, 86
qomegaSq (pomegaSq), 113
qt(), 30, 31
quantile, 51
quietGitLabUpdate
   (quietRemotesInstall), 119
quietRemotesInstall, 119
qVec, 120
qVecSum (qVec), 120
rawDataDiamondLayer
   (diamondCoordinates), 52
rbind_df_list, 121
rbind_dfs, 121
rCohensd (cohensdCI), 29
rd (cohensdCI), 29
regrInfluential, 122
rep(), 87
repeatStr, 123
report, 123
repStr (repeatStr), 123
require(), 26
reset (opts), 112
rf(), 114
romegaSq (pomegaSq), 113
rownames(), 38
rt(), 31
safeRequire, 124
samplingDistribution (dataShape), 45
scaleDiagnosis, 124
scaleDiagnosis_partial
   (scaleDiagnosis), 124
scaleReliability (scaleStructure), 127
scales::rescale(), 136
scaleStructure, 127
scaleStructure(), 125
scaleStructure_partial
   (scaleStructure), 127
scaleStructurePartial (scaleStructure), 127
scatterMatrix, 131
scatterMatrix(), 126
set (opts), 112
setCaptionNumberingKnitrHook, 133
setFigCapNumbering
   (setCaptionNumberingKnitrHook), 133
setFigCapNumbering(), 89
setTabCapNumbering
   (setCaptionNumberingKnitrHook), 133
setTabCapNumbering(), 89
sharedSubString, 134
simDataSet, 135
spearmanBrown, 138
spearmanBrown_requiredLength
   (spearmanBrown), 138
spearmanBrown_reversed (spearmanBrown), 138
stats::quantile(), 64
stats::wilcox.test(), 15
strToFilename, 138
strwrap(), 148
summary, 51
suspectParticipants, 139
Sys.time, 98
table(), 109
testRetestAlpha, 140
testRetestCES, 141
testRetestReliability, 143
testRetestSimData, 145
uniDimColors (biDimColors), 20
varsToDiamondPlotDf
   (diamondCoordinates), 52
vecTxt, 147
vecTxtB (vecTxt), 147
vecTxtM (vecTxt), 147
vecTxtQ (vecTxt), 147
viridisPalette, 148
wrapVector, 148
zotero_construct_export_call, 149
zotero_download_and_export_items, 150
zotero_get_all_items, 151
zotero_nr_of_items, 151