Package ‘behaviorchange’

October 21, 2020

Type Package

Title Tools for Behavior Change Researchers and Professionals

Version 0.2.4

Maintainer Gjalt-Jorn Peters <gjalt-jorn@a-bc.eu>

License GPL (>= 3)

Description Contains specialised analyses and
visualisation tools for behavior change science.
These facilitate conducting determinant studies
(for example, using confidence interval-based
estimation of relevance, CIBER, or CIBERlite
plots, see Crutzen, Noijen & Peters (2017)
<doi:10.3389/fpubh.2017.00165>),
systematically developing, reporting,
and analysing interventions (for example, using
Acyclic Behavior Change Diagrams), and reporting
about intervention effectiveness (for example, using
the Numbers Needed for Change, see Gruijters & Peters
(2017) <doi:10.31234/osf.io/2bau7>), and computing the
required sample size (using the Meaningful Change
Definition, see Gruijters & Peters (2019)
<doi:10.31234/osf.io/jc295>).
This package is especially useful for
researchers in the field of behavior change or
health psychology and to behavior change
professionals such as intervention developers and
prevention workers.

URL https://r-packages.gitlab.io/behaviorchange

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

Encoding UTF-8

LazyData true

RoxygenNote 7.1.1

Depends R (>= 3.0.0)

Imports BiasedUrn (>= 1.07), data.tree (>= 0.7.5), DiagrammeR (>= 1.0.0), DiagrammeRsvg (>= 0.1.0), ggplot2 (>= 2.2.1), googlesheets (>= 0.3.0), gridExtra (>= 2.3), gtable (>= 0.2.0), magrittr (>= 0.1.5), png (>= 0.1), ufs (>= 0.3.2), viridis (>= 0.5.1), yum (>= 0.0.1)

Suggests kableExtra, knitr, openxlsx, rmarkdown, rsvg, webshot

VignetteBuilder knitr

NeedsCompilation no

Author Gjalt-Jorn Peters [aut, cre] (<https://orcid.org/0000-0002-0336-9589>), Rik Crutzen [ctb] (<https://orcid.org/0000-0002-3731-6610>), Stefan Gruijters [ctb] (<https://orcid.org/0000-0003-0141-0071>)

Repository CRAN

Date/Publication 2020-10-21 20:20:17 UTC

R topics documented:

---

abcd *Acyclic Behavior Change Diagram*

Description

This function generates an acyclic behavior change diagram (ABCD) from a specification in a google sheet or .csv file. An ABCD is a logic model that illustrates the assumptions underlying a behavior change intervention. Specifically, the ABCD shows the assumed causal and structural assumptions, thereby showing what is assumed to cause what (e.g. which elements of the intervention are assumed to influence which aspects of the target population's psychology?) and what is assumed to consist of what (e.g. which determinants are assumed to contain which specific aspects of the target population's psychology?).

Usage

```
abcd(
  specs,
  specCols = c("bcps", "cnds", "apps", "sdts", "dets", "pobs", "behs"),
  localBackup = NULL,
  title = "Acyclic Behavior Change Diagram\n\n",
  outputFile = NULL,
  outputWidth = 3000,
  outputHeight = 1500,
  includeColNames = TRUE,
  maxLabelLength = 30,
  nodeFontSize = 10,
  edgeFontSize = 8,
  colNameFontSize = nodeFontSize,
  penWidth = 1,
  silent = FALSE,
  returnGraphOnly = FALSE,
  returnSvgOnly = FALSE,
  regExReplacements = list(c("\\\"", "`"), c("\\'", "`"), c("\\\\", "/"))
)

## S3 method for class 'abcdiagram'
print(
  x,
  width = x$input$width,
  height = x$input$height,
  title = DiagrammeR::get_graph_name(x$output$graph),
  ...
)
```

Arguments

`specs`	The specifications: either a google sheets URL, the path to a local file, a character vector with both, or a matrix or data frame
`specCols`	The order of the columns. This character vector specified the order of the elements of an ABCD. In the default order, from left to right, these are (see below for definitions and more details): • `bcps` = **Behavior Change Principles (BCPs)**; • `cnds` = **Conditions for effectiveness**; • `apps` = **Applications**; • `sdts` = **Sub-determinants**; • `dets` = **Determinants**; • `pobs` = **Performance Objectives**; • `behs` = **Behaviors**;

`localBackup`	Whether to write the specifications to a local backup
`title`	The title of the diagram
`outputFile`	If specified, the ABCD is written to this file using DiagrammeR::export_graph.
`outputWidth, outputHeight`	If an `outputFile` is specified, these determine its width and height (in pixels)
`includeColNames`	Whether to include the column names as titles/legend for the entities in each ’column’ of the ABCD.
`maxLabelLength`	At which width to word wrap the labels.
`nodeFontSize, edgeFontSize, colNameFontSize`	Font sizes of the nodes (i.e. the text in boxes), edges (basically the conditions for effectiveness) and the column names (at the bottom).
`penWidth`	The width of the pen to draw the strokes.
`silent`	Whether to suppress (`TRUE`) or show (`FALSE`) more detailed information.
`returnGraphOnly, returnSvgOnly`	Whether to return the full results object or only either the DiagrammeR::DiagrammeR graph or a one-value character vector containing a Scalable Vector Graphic as produced by `DiagrammeRsvg::export_svg()`.
`regExReplacements`	A list of pairs of regular expressions that will be applied to the specifications before generating the ABCD. This can be used to sanitize problematic characters (e.g. ’, " and \).
`x`	The ABCD object to print (as generated by a call to `abcd`).
`width, height`	Width and height to use when printing the ABCD.
`...`	Any additional arguments are passed on to `DiagrammeR::render_graph()`.

Details

Specifically, a full ABCD is a model that shows the following elements:

- **Behavior Change Principles (BCPs)**: The specific psychological principles engaged to influence the relevant sub-determinants, usually selected using the determinants to which the sub-determinants ’belong’. These are also known as methods of behavior change in the Intervention Mapping framework, or behavior change techniques, BCTs, in the Behavior Change Wheel approach. For a list of 99 BCPs, see Kok et al. (2016).
- **Conditions for effectiveness**: The conditions that need to be met for a Behavior Change Principle (BCP) to be effective. These conditions depend on the specific underlying Evolutionary Learning Processes (ELPs) that the BCP engages (Crutzen & Peters, 2018). If the conditions for effectiveness (called *parameters* for effectiveness in the Intervention Mapping framework) are not met, the method will likely not be effective, or at least, not achieve its maximum effectiveness.
- **Applications**: Since BCP’s describe aspects of human psychology in general, they are necessarily formulated on a generic level. Therefore, using them in an intervention requires translating them to the specific target population, culture, available means, and context. The result of this translation is the application of the BCP. Multiple BCPs can be combined into one application; and one BCP can be applied in multiple applications (see Kok, 2014).

- **Sub-determinants**: Behavior change interventions engage specific aspects of the human psychology (ideally, they specifically, target those aspects found most important in predicting the target behavior, as can be established with `CIBER` plots. These aspects are called sub-determinants (the Intervention Mapping framework references *Change Objectives*, which are sub-determinants formulated according to specific guidelines). In some theoretical traditions, sub-determinants are called *beliefs*.
- **Determinants**: The overarching psychological constructs that are defined as clusters of specific aspects of the human psychology that explain humans' behavior (and are targeted by behavior change interventions). Psychological theories contain specific definitions of such determinants, and make statements about how they relate to each other and to human behavior. There are also theories (and exists empirical evidence) on how these determinants can be changed (i.e. BCPs), so althought the sub-determinants are what is targeted in an intervention, the selection of feasible BCPs requires knowing to which determinants those sub-determinants belong.
- **Performance objectives**: The specific sub-behaviors that often underlie (or make up) the ultimate target behavior. These are distinguished from the overarching target behavior because the relevant determinants of these sub-behaviors can be different: for example, the reasons why people do or do not *buy* condoms can be very different from the reasons why they do or do not *carry* condoms or why they do or do not *negotiate* condom use with a sexual partner.
- **Behavior**: The ultimate target behavior of the intervention, usually an umbrella that implicitly contains multiple performance objectives.

For details, see Peters et al. (2019).

Value

A list consisting of an `input`, `intermediate`, and `output` list, where the ABCD is stored in the `output` list as a DiagrammeR::DiagrammeR called `graph`.

Author(s)

Gjalt-Jorn Peters, <`gjalt-jorn@a-bc.eu`>, with contributions from Matti Heino and Sander Eggers.

References

Crutzen, R., & Peters, G.-J. Y. (2018). Evolutionary learning processes as the foundation for behaviour change. *Health Psychology Review,* 12(1), 43–57. https://doi.org/10.1080/17437199.2017.1362569

Kok, G. (2014). A practical guide to effective behavior change: How to apply theory- and evidence-based behavior change methods in an intervention. *European Health Psychologist*, 16(5), 156–170. https://doi.org/10.31234/osf.io/r78wh

Kok, G., Gottlieb, N. H., Peters, G.-J. Y., Mullen, P. D., Parcel, G. S., Ruiter, R. A. C., ... Bartholomew, L. K. (2016). A taxonomy of behavior change methods: an Intervention Mapping approach. *Health Psychology Review*, 10(3), 297–312. https://doi.org/10.1080/17437199.2015.1077155

Peters, G.-J. Y., et al. (2019) The core of behavior change: introducing the Acyclic Behavior Change Diagram to report and analyze interventions.

Examples

```
### Load one of the ABCD matrices supplied
### with the behaviorchange package
data(abcd_specification_example_xtc);

### Create ABCD matrix (using 'print' to allow pkgdown() to print properly).
print(behaviorchange::abcd(abcd_specification_example_xtc));
```

---

abcd_specs_examples *Simple example datasets for ABCDs*

Description

This are three (nested) datasets illustrating the logic model of change for a simple condom use intervention in a way that can be visualised using the abcd function. The full dataset is `abcd_specs_full`, a subset that does not explicitly include the conditions for effectiveness (instead showing letters that can then be explained in, for example, the manuscript text) is called `abcd_specs_without_conditions`, and a version that only contains the information about one sub-behavior (performance objective) is available as `abcd_specs_single_po_without_conditions`. The variables in the full dataset are:

Usage

```
data(abcd_specs_complete)

data(abcd_specs_without_conditions)

data(abcd_specs_single_po_without_conditions)

data(abcd_specification_example_xtc)

data(abcd_specs_dutch_xtc)

data(abcd_specification_empty)
```

Format

For `abcd_specs_complete`, a data frame with 7 variables and 7 rows; for `abcd_specs_without_conditions`, a data frame with 6 variables and 7 rows; for `abcd_specs_single_po_without_conditions`, a data frame with 5 variables and 4 rows; for `abcd_specification_example_xtc` and `abcd_specs_dutch_xtc`, a data frame with 7 variables and 5 rows' and for `abcd_specification_empty`, a data frame with 7 variables and 1 row.

Details

- Behavior Change Principles: The behavior change principles (BCPs), also known as methods for behavior change or 'behavior change techniques' (BCTs), that describe the psychological principles that are assumed to realise the change in the (sub-)determinants.

- Conditions for effectiveness\\n(e.g. parameters for use): The conditions for effectiveness that describe the constraints and considerations taken into account in the translation of the BCPs to practical applications for the relevant target population, context, culture, etc.
- `Applications`: The applications of these BCPs. Where the BCPs describe theoretical principles, the applications are more or less tangible intervention elements.
- Sub-determinants\\n(e.g. beliefs; can be formulated as Change Objectives): The specific aspects of teh target population's psychology that are targeted by the BCPs (e.g. beliefs, or in Intervention Mapping vocabulary, Change Objectives).
- `Determinants`: The determinants, psychological constructs, that the targeted sub-determinants are a part of, and that together predict the Performance Objectives (sub-behaviors).
- Performance Objectives: Explicitly defined sub-behaviors at a level of specificity that distinguishes them from other sub-behaviors, and that together form the target behavior.
- Target Behavior: The ultimate target behavior, usually defined at a relatively general level.

In addition to these three datasets, a Dutch example specification is included named `abcd_specs_dutch_xtc`, and the same in English as `abcd_specification_example_xtc`.

Finally, `abcd_specification_empty` is an empty 'template' ABCD matrix.

`apply_graph_theme` *Apply multiple DiagrammeR global graph attributes*

Description

Apply multiple DiagrammeR global graph attributes

Usage

```
apply_graph_theme(graph, ...)
```

Arguments

`graph`	The DiagrammeR::DiagrammeR graph to apply the attributes to.
`...`	One or more character vectors of length three, where the first element is the attribute, the second the value, and the third, the attribute type (`graph`, `node`, or `edge`).

Value

The DiagrammeR::DiagrammeR graph.

Examples

```
abcd_complete <- behaviorchange::abcd(behaviorchange::abcd_specs_complete)$output$graph;
abcd_complete <- apply_graph_theme(abcd_complete,
                                   c("penwidth", 5, "node"),
                                   c("penwidth", 15, "edge"));
```

CIBER *Confidence Interval-Based Estimation of Relevance (CIBER)*

Description

This function generates a high-level plot consisting of several diamond plots. This function is useful for estimating the relative relevance of a set of determinants of, for example, behavior. The plot in the left hand panel shows each determinant's distribution with a diamond representing the confidence interval. The right hand plot shows the determinants' associations to one or more 'target' variables, such as behavior or determinants of behavior.

Usage

```
CIBER(
  data,
  determinants,
  targets,
  conf.level = list(means = 0.9999, associations = 0.95),
  subQuestions = NULL,
  leftAnchors = rep("Lo", length(determinants)),
  rightAnchors = rep("Hi", length(determinants)),
  outputFile = NULL,
  outputWidth = NULL,
  outputHeight = NULL,
  outputUnits = "in",
  outputParams = list(),
  orderBy = NULL,
  decreasing = NULL,
  numberSubQuestions = FALSE,
  generateColors = list(means = c("red", "blue", "green"), associations = c("red",
    "grey", "green")),
  strokeColors = viridis::viridis(length(targets)),
  vLines = c(-0.5, 0, 0.5),
  vLineColors = "grey",
  titlePrefix = "Means and associations (r) with",
  titleVarLabels = NULL,
  titleSuffix = "",
  fullColorRange = NULL,
  associationsAlpha = 0.5,
  returnPlotOnly = TRUE,
  drawPlot = TRUE,
  baseSize = 0.8,
  dotSize = 2.5 * baseSize,
  baseFontSize = 10 * baseSize,
  theme = ggplot2::theme_bw(base_size = baseFontSize),
  xbreaks = NULL,
  rsq = TRUE,
  ...
)

binaryCIBER(
  data,
  determinants,
  targets,
  conf.level = list(means = 0.9999, associations = 0.95),
  subQuestions = NULL,
  leftAnchors = rep("Lo", length(determinants)),
  rightAnchors = rep("Hi", length(determinants)),
  outputFile = NULL,
  outputWidth = NULL,
  outputHeight = NULL,
  outputUnits = "in",
  outputParams = list(),
  orderBy = NULL,
  decreasing = NULL,
  numberSubQuestions = FALSE,
  comparisonColors = viridis::viridis(2, end = 0.5),
  categoryLabels = NULL,
 generateColors = list(means = c("red", "blue", "green"), associations = c("red",
    "grey", "green")),
  strokeColors = viridis::viridis(length(targets)),
  vLines = c(-0.8, 0, 0.8),
  vLineColors = "grey",
  titlePrefix = "Means and associations (d) with",
  titleVarLabels = NULL,
  titleSuffix = "",
  fullColorRange = NULL,
  associationsAlpha = 0.5,
  returnPlotOnly = TRUE,
  drawPlot = TRUE,
  baseSize = 0.8,
  dotSize = 2.5 * baseSize,
  baseFontSize = 10 * baseSize,
  theme = ggplot2::theme_bw(base_size = baseFontSize),
  xbreaks = NULL,
  ...
)

detStructCIBER(
  determinantStructure,
  data,
  conf.level = list(means = 0.9999, associations = 0.95),
  subQuestions = NULL,
  leftAnchors = rep("Lo", length(determinants)),
  rightAnchors = rep("Hi", length(determinants)),
  orderBy = 1,
  decreasing = NULL,
 generateColors = list(means = c("red", "blue", "green"), associations = c("red",
    "grey", "green")),
  strokeColors = NULL,
  titlePrefix = "Means and associations with",
  titleVarLabels = NULL,
  titleSuffix = "",
  fullColorRange = NULL,
  associationsAlpha = 0.5,
  baseSize = 0.8,
  dotSize = 2.5 * baseSize,
  baseFontSize = 10 * baseSize,
  theme = ggplot2::theme_bw(base_size = baseFontSize),
  ...
)
```

Arguments

`data`	The dataframe containing the variables.
`determinants`	The 'determinants': the predictors (or 'covariates') of the target variables(s) (or 'criteria').
`targets`	The 'targets' or 'criteria' variables: the variables predicted by the determinants.
`conf.level`	The confidence levels for the confidence intervals: has to be a named list with two elements: `means` and `associations`, specifying the desired confidence levels for the means and associations, respectively. The confidence level for the associations is also used for the intervals for the proportions of explained variance.
`subQuestions`	The subquestions used to measure each determinants. This can also be used to provide pretty names for the variables if the determinants were not measured by one question each. Must have the same length as `determinants`.
`leftAnchors`	The anchors to display on the left side of the left hand panel. If the determinants 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 `determinants`.
`rightAnchors`	The anchors to display on the left side of the left hand panel. If the determinants 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 `determinants`.
`outputFile`	The file to write the output to (the plot is not stored to disk if `NULL`). The extension can be specified to change the file type.
`outputWidth, outputHeight, outputUnits`	The width, height, and units for the output file.
`outputParams`	More advanced parameters for the output file. This can be used to pass arguments to `ggplot2::ggsave()`, such as passing `outputParams=list(type="cairo-png")` to use anti-aliasing when saving a PNG file.

`orderBy`	Whether to sort the determinants. Set to `NULL` to not sort at all; specify the name or index of one of the `targets` to sort by the point estimates of the associations with that target variable. Use `decreasing` to determine whether to sort in ascending or descending order. For convenience, if `orderBy` is not `NULL`, but `decreasing` is, the determinants are sorted in descending (decreasing) order.
`decreasing`	Whether to sort the determinants. Specify `NULL` to not sort at all, `TRUE` to sort in descending order, and `FALSE` to sort in ascending order. If `decreasing` is nor `NULL`, but `orderBy` is `NULL`, the determinants are sorted by their means. For convenience, if `orderBy` is not `NULL`, but `decreasing` is, the determinants are sorted in descending (decreasing) order.
`numberSubQuestions`	Whether or not to number the subquestions. If they are numbered, they are numbered from the top to the bottom.
`generateColors`	The colors to use to generate the gradients for coloring the diamonds representing the confidence intervals. Has to be a named list with two elements: `means` and `associations`, specifying the desired colors for the means and associations, respectively.
`strokeColors`	The palette to use to color the stroke of the confidence intervals for the associations between the determinants and the targets. Successive colors from this palette are used for the targets.
`vLines, vLineColors`	In the association plot, vertical lines can be plotted to facilitate interpretation. Specify their locations and colors here, or set one or both to `NULL` to eliminate them.
`titlePrefix`	Text to add before the list of target names and the proportions of explained variance for each target. This plot title also serves as legend to indicate which target 'gets' which each color.
`titleVarLabels`	Optionally, variable labels to use in the plot title. Has to be the exact same length as `targets`.
`titleSuffix`	Text to add after the list of target names and the proportions of explained variance for each target.
`fullColorRange`	If colors are specified, this can be used to specify which values, for the determinant confidence intervals in the left hand panel, are the minimum and maximum. This is useful if those scores are not actually in the data (e.g. for extremely skewed distributions). If `NULL`, the range of all individual scores on the determinants is used. For the associations, `c(-1,1)` is always used as `fullColorRange`.
`associationsAlpha`	The alpha level (transparency) of the confidence interval diamonds in the right hand plot. Value between 0 and 1, where 0 signifies complete transparency (i.e. invisibility) and 1 signifies complete 'opaqueness'.
`returnPlotOnly`	Whether to return the entire object that is generated (including all intermediate objects) or only the plot.
`drawPlot`	Whether the draw the plot, or only return it.
`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.
`xbreaks`	Which breaks to use on the X axis (can be useful to override `ggplot2`'s defaults).
`rsq`	Whether to compute the R squared values.
`...`	These arguments are passed on to `biAxisDiamondPlot` (for the left panel) and `diamondPlot` (for the right panel). Note that all argument are passed to both those functions.
`comparisonColors`	Colors to use for the two groups in a binary CIBER plot with one (dichotomous) target.
`categoryLabels`	Labels for the two values of the target.
`determinantStructure`	When using `detStructCIBER`, the determinant structure as generated by `determinantStructure` is included here. `determinants`, `targets`, `subQuestions`, `leftAnchors`, and `rightAnchors` are then read from the `determinantStructure` object. In other words: once a `determinantStructure` has been generated, only `dat` and `determinantStructure` have to be provided as argument to generate a CIBER diamond plot.

Details

Details are explained in Crutzen & Peters (2017).

Value

Depending on the value of `returnPlotOnly`, either the plot only (a `gtable` object) or an object containing most objects created along the way (in which case the plot is stored in `$output$plot`).

The plot has `width` and `height` attributes which can be used when saving the plot.

References

Crutzen, R., Peters, G.-J. Y., & Noijen, J. (2017). How to Select Relevant Social-Cognitive Determinants and Use them in the Development of Behaviour Change Interventions? Confidence Interval-Based Estimation of Relevance. http://dx.doi.org/

See Also

`determinantStructure`

Examples

```
### This example uses the determinant study Party Panel 17.1;
### see ?behaviorchange::BBC_data for more information.
data(BBC_pp17.1);
behaviorchange::CIBER(data=BBC_pp17.1,
                      determinants=c('epw_AttExpect_hearingDamage',
                                     'epw_AttExpect_highTone',
                                        'epw_AttExpect_musicVolume',
                                        'epw_AttExpect_musicFidelity',
                                        'epw_AttExpect_loudConversation',
                                        'epw_AttExpect_musicFocus',
                                        'epw_AttExpect_musicEnjoy'),
                          targets=c('epw_attitude'));

### With a binary target
data(BBC_pp17.1);
behaviorchange::binaryCIBER(data=BBC_pp17.1,
                              determinants=c('epGeneralBeliefs_loudnessPreference',
                                             'epGeneralBeliefs_loudnessGenre',
                                             'epGeneralBeliefs_loudnessTooMuch',
                                             'epGeneralBeliefs_priceFoam',
                                             'epGeneralBeliefs_priceSilicon',
                                             'epGeneralBeliefs_priceCustom'),
                              targets=c('epPossession'),
                              categoryLabels = c('no',
                                                 'yes'));
```

`CIBERlite` *CIBERlite*

Description

CIBERlite plots can be used to quickly get an idea of means and correlations of a small number of determinants. They were developed to facilitate conducting and interpreting determinant studies by prevention professionals.

Usage

```
CIBERlite(
  data,
  determinants,
  targets,
  determinantOrder = NULL,
  determinantLabels = NULL,
  subDeterminantLabels = NULL,
  title = NULL,
  conf.level = 0.95,
  scaleRange = NULL,
  determinantAesthetics = list(fill = "black", color = NA, alpha = 0.5),
  subDeterminantAesthetics = list(fill = "black", color = NA, alpha = 0.5),
  rDiamondAesthetics = list(fill = "#c4c4c4", color = NA, alpha = 0.75)
)
```

Arguments

`data`	The dataframe containing the variables.
`determinants`	Either a character vector with the names of the determinants, or a list of named character vectors, where each vector contains a number of subdeterminants, and each vector's name is the name of the more proximal determinant (i.e. that 'contains' those subdeterminants).
`targets`	A character vector with the names of the targets (i.e. more proximal determinants, behavior, etc).
`determinantOrder`	The order in which to display the determinants (if this needs to be different from the order as provided in `determinants`).
`determinantLabels`	The labels to use for the determinants.
`subDeterminantLabels`	The labels to use for the subdeterminants.
`title`	The title of the plot.
`conf.level`	The confidence levels: a list with two named values; the confidence level for the means, named `means`, and the confidence level for the associations, named `associations`.
`scaleRange`	The full range of the scale of the determinants/subdeterminants; the minimum and maximum values are used if this is not provided.
`determinantAesthetics, subDeterminantAesthetics, rDiamondAesthetics`	The aesthetics for the determinants, subdeterminants, and correlation diamonds, each a list containing three named values: `fill`, `color`, and `alpha`.

Details

More details will be provided in a forthcoming paper; until then, see https://CIBERlite.com

Value

A `ggplot`.

Examples

```
### This example uses the determinant study Party Panel 15.1;
### see ?behaviorchange::BBC_data for more information.
data(BBC_pp15.1);
CIBERlite(data=BBC_pp15.1,
          determinants=c('highDose_attitude',
                         'highDose_perceivedNorm',
                         'highDose_pbc'),
          targets=c('highDose_intention'));
```

complecs *Create a COMPLECS graph*

Description

COMPLECS was developed to help make sense of complex systems. It reads data from a number of worksheets in a spreadsheet and generates a diagram according to those specifications. Originally, COMPLECS was developed to visualise a problem during the needs assessment phase of intervention development.

Usage

```
complecs(
  input,
  title = "COMPLECS overview",
  layout = "neato",
 graph_styling = list(c("outputorder", "nodesfirst", "graph"), c("overlap", "false",
    "graph"), c("fixedsize", "false", "node")),
  directed = TRUE,
  outputFile = NULL,
  outputWidth = NULL,
  outputHeight = NULL,
  returnSvgOnly = FALSE,
  maxLabelLength = 20,
  regExReplacements = list(c("\\\"", "`"), c("\\'", "`"), c("\\\\", "/"))
)

## S3 method for class 'complecs'
print(
  x,
  width = x$input$width,
  height = x$input$height,
  title = DiagrammeR::get_graph_name(x$output$graph),
  ...
)
```

Arguments

`input`	Either a link to a Google Sheet, or a path to an Excel file.
`title`	The title of the COMPLECS graph.
`layout`	The layout to use; has to be one of the `DiagrammeR` layout types (`dot`, `neato`, `circo` and `twopi`).
`graph_styling`	Additional styling to apply; a list with three-element vectors, where the three elements correspond to, respectively, the `attr`, `value`, and `attr_type` arguments for [DiagrammeR::add_global_graph_attrs().
`directed`	Whether to draw directed arrows or not.

`outputFile`	A character vector where each element is one path (including filename) to write the graph to.
`outputWidth, outputHeight`	If not `NULL`, a way to override the width and height when calling `complecs` to generate a COMPLECS overview.
`returnSvgOnly`	Whether to only return the SVG in a character vector.
`maxLabelLength`	The number of characters where to wrap the labels.
`regExReplacements`	A list of pairs of regular expressions that will be applied to the specifications before generating the ABCD. This can be used to sanitize problematic characters (e.g. ', " and \\).
`x`	The object to print (i.e. a result of a call to `complecs`).
`width, height`	If not `NULL`, a way to override the width and height when calling `print` to print a COMPLECS overview.
`...`	Any additional arguments for the `print()` method are passed to `DiagrammeR::render_graph()`.

Details

COMPLECS is a recursive acronym for COMPLECS Organises Multiple Players & Linked Environments using Connected Specifications.

Value

A `complecs` object that includes the graph and the graph in SVG in `output$graph` and `output$graphSvg`.

Examples

```
complecs(paste0("https://docs.google.com/spreadsheets/d/",
         "1WMO15xroy4a0RfpuZ8GhT-NfDoxwS34w9PrWp8rGjjk"));
```

`convert.threshold.to.er`

Visualising Numbers Needed for Change

Description

These functions can be used to visualise Numbers Needed for Change (or Numbers Needed to Treat). `erDataSeq` is a helper function to generate an Event Rate Data Sequence, and it uses `convert.threshold.to.er` and `convert.er.to.threshold` to convert thresholds to event rates and vice versa.

Usage

```
convert.threshold.to.er(
  threshold,
  mean,
  sd,
  eventIfHigher = TRUE,
  pdist = stats::pnorm
)

convert.er.to.threshold(
  er,
  mean,
  sd,
  eventIfHigher = TRUE,
  qdist = stats::qnorm
)

erDataSeq(
  er = NULL,
  threshold = NULL,
  mean = NULL,
  sd = NULL,
  eventIfHigher = TRUE,
  pRange = c(1e-06, 0.99999),
  xStep = 0.01
)

ggNNC(
  cerDataSeq,
  d = NULL,
  eventDesirable = TRUE,
  r = 1,
  xlab = "Continuous outcome",
  plotTitle = c("Numbers Needed for Change = ", ""),
  theme = ggplot2::theme_bw(),
  lineSize = 1,
  cerColor = "#EBF2F8",
  eerColor = "#172F47",
  cerLineColor = "#888888",
  eerLineColor = "#000000",
  dArrowColor = "#000000",
  cerAlpha = 0.66,
  eerAlpha = 0.66,
  xLim = NULL,
  xLimAutoDensityTolerance = 0.001,
  showLegend = TRUE,
  verticalLineColor = "#172F47",
  desirableColor = "#00FF00",
  desirableAlpha = 0.2,
  undesirableColor = "#FF0000",
  undesirableAlpha = 0.2,
  desirableTextColor = "#009900",
  undesirableTextColor = "#990000",
  dArrowDistance = 0.04 * max(cerDataSeq$density),
  dLabelDistance = 0.08 * max(cerDataSeq$density)
)
```

Arguments

`threshold`	If the event rate is not available, a threshold value can be specified instead, which is then used in conjunction with the mean (`mean`) and standard deviation (`sd`) and assuming a normal distribution to compute the event rate.
`mean`	The mean of the control group distribution.
`sd`	The standard deviation (of the control distribution, but assumed to be the same for both distributions).
`eventIfHigher`	Whether scores above or below the threshold are considered 'an event'.
`pdist, qdist`	Distributions to use when converting thresholds to event rates and vice versa; defaults to the normal distribution.
`er`	Event rate to visualise (or convert).
`pRange`	The range of probabilities for which to so the distribution.
`xStep`	Precision of the drawn distribution; higher values mean lower precision/granularity/resolution.
`cerDataSeq`	The `cerDataSeq` object.
`d`	The value of Cohen's *d*.
`eventDesirable`	Whether an event is desirable or undesirable.
`r`	The correlation between the determinant and behavior (for mediated NNC's).
`xlab`	The label to display for the X axis.
`plotTitle`	The title of the plot; either one character value, this value if used; if two, they are considered a prefix and suffix to be pre/appended to the NNC value.
`theme`	The theme to use for the plot.
`lineSize`	The thickness of the lines in the plot.
`cerColor`	The color to use for the event rate portion of the control group distribution.
`eerColor`	The color to use for the event rate portion of the experimental group distribution.
`cerLineColor`	The line color to use for the control group distribution.
`eerLineColor`	The line color to use for the experimental group distribution.
`dArrowColor`	The color of the arrow to show the effect size.
`cerAlpha`	The alpha value (transparency) to use for the control group distribution.
`eerAlpha`	The alpha value (transparency) to use for the control group distribution.
`xLim`	This can be used to manually specify the limits for the X axis; if `NULL`, sensible limits will be derived using `xLimAutoDensityTolerance`.

`xLimAutoDensityTolerance`	If `xLim` is `NULL`, the limits will be set where the density falls below this proportion of its maximum value.
`showLegend`	Whether to show the legend (only if showing two distributions).
`verticalLineColor`	The color of the vertical line used to indicate the threshold.
`desirableColor`	The color for the desirable portion of the X axis.
`desirableAlpha`	The alpha for the desirable portion of the X axis.
`undesirableColor`	The color for the undesirable portion of the X axis.
`undesirableAlpha`	The color for the undesirable portion of the X axis.
`desirableTextColor`	The color for the text to indicate the desirable portion of the X axis.
`undesirableTextColor`	The color for the text to indicate the undesirable portion of the X axis.
`dArrowDistance`	The distance of the effect size arrow from the top of the distributions.
`dLabelDistance`	The distance of the effect size label from the top of the distributions.

Details

These functions are used by `nnc()` to show the distributions, and event rates. They probably won't be used much on their own.

Value

`erDataSeq` returns a data sequence; `ggNNC` a `ggplot2::ggplot()`.

Author(s)

Gjalt-Jorn Peters & Stefan Gruijters

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

References

Gruijters, S. L., & Peters, G. Y. (2019). Gauging the impact of behavior change interventions: A tutorial on the Numbers Needed to Treat. *PsyArXiv.* doi:10.31234/osf.io/2bau7

See Also

`nnc()`

Examples

```
### Show distribution for an event rate value of 125
behaviorchange::ggNNC(behaviorchange::erDataSeq(threshold=125, mean=90, sd=30));

### If the event occurs under the threshold instead of
### above it
behaviorchange::ggNNC(behaviorchange::erDataSeq(threshold=125,
                                                mean=90, sd=30,
                      eventIfHigher = FALSE));

### ... And for undesirable events (note how
### desirability is an argument for ggNNC, whereas
### whether an event occurs 'above' or 'below' the
### threshold is an argument for erDataSeq):
behaviorchange::ggNNC(behaviorchange::erDataSeq(threshold=125,
                                                mean=90, sd=30,
                      eventIfHigher = FALSE),
      eventDesirable = FALSE);

### Show event rate for both experimental and
### control conditions, and show the numbers
### needed for change
behaviorchange::ggNNC(behaviorchange::erDataSeq(threshold=125,
                                                mean=90, sd=30),
                      d=.5);

### Illustration of how even with very large effect
### sizes, if the control event rate is very high,
### you'll still need a high number of NNC
behaviorchange::ggNNC(behaviorchange::erDataSeq(er=.9),
                      d=1);
```

---

determinantStructure *Determinant Structure specification*

Description

These functions can be used to specify a determinant structure: a hierarchical structure of determinants that can then be conveniently plotted and analysed, for example using detStructCIBER. These functions are made to be used together; see the example and the forthcoming article for more information.

Usage

```
determinantStructure(name, selection = NULL, ...)

determinantVar(name, selection = NULL, ...)
```

```
subdeterminants(name, selection = NULL, ...)

subdeterminantProducts(name, selection = NULL, ...)

## S3 method for class 'determinantStructure'
plot(x, useDiagrammeR = FALSE, ...)

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

Arguments

`name`	The name of the variable that is specified.
`selection`	A regular expression to use to select the variables in a dataframe that are considered items that together form this variable. For `determinantStructure`, a list can be provided that also contains a named regular expression with the name 'behaviorRegEx', which specifies the name of the behavior to which this determinant structure pertains.
`...`	Any additional arguments are other determinant structure building functions. These are used to construct the determinant structure 'tree'.
`x`	The `determinantStructure` object to print or plot.
`useDiagrammeR`	Whether to simply use `print(plot(x))` (if `FALSE`) or whether to use data.tree::ToDiagrammeRGraph, tweak it a bit, by setting global graph attributes, and then using DiagrammeR::render_graph (if `TRUE`).

Details

This family of functions will be explained more in detail in a forthcoming paper.

`plot` and `print` methods plot and print a `determinantStructure` object.

Value

A `determinantStructure` object, which is a data.tree object.

Author(s)

Gjalt-Jorn Peters, `<gjalt-jorn@a-bc.eu>`

See Also

`detStructAddVarLabels`, `detStructAddVarNames`, `detStructComputeProducts`, `detStructComputeScales`, `detStructCIBER`

Examples

```
determinantStructure('using R',
                     list('using R',
                          behaviorRegEx = 'some RegEx'),
                     determinantVar("Intention",
                                    "another RegEx",
                                    determinantVar("Attitude",
                                                   "third RegEX",
                                                   subdeterminants("Likelihood",
                                                                   "4th RegEx"),
                                                  subdeterminants("Evaluation",
                                                                  "5th RegEx"),
                                                  subdeterminantProducts("attProduct",
                                                                         c("4th RegEx",
                                                                         "5th RegEx"))),
                                    determinantVar("perceivedNorm",
                                                   "6th RegEx",
                                                   subdeterminants("Approval",
                                                                   "7th RegEx"),
                                                 subdeterminants("Motivation to comply",
                                                                   "8th RegEx"),
                                                   subdeterminantProducts("normProduct",
                                                                          c("7th RegEx",
                                                                         "8th RegEx"))),
                                              determinantVar("pbc",
                                                             "9th RegEx",
                                                     subdeterminants("Control beliefs",
                                                                       "10th RegEx"))));
```

`detStructAddVarLabels` *Functions to preprocess determinant structures*

Description

These functions are used in conjunction with the determinantStructure family of funtions to conveniently work with determinant structures.

Usage

```
detStructAddVarLabels(
  determinantStructure,
  varLabelDf,
  varNameCol = "varNames.cln",
  leftAnchorCol = "leftAnchors",
  rightAnchorCol = "rightAnchors",
  subQuestionCol = "subQuestions",
  questionTextCol = "questionText"
)
```

```
detStructAddVarNames(determinantStructure, names)

detStructComputeProducts(determinantStructure, data, append = TRUE)

detStructComputeScales(
  determinantStructure,
  data,
  append = TRUE,
  separator = "_"
)
```

Arguments

`determinantStructure`	The `determinantStructure` object.
`varLabelDf`	The variable label dataframe as generated by the `processLSvarLabels` in the `userfriendlyscience` package. It is also possible to specify a 'homemade' dataframe, in which case the column names have to specified (see the next arguments).
`varNameCol`	The name of the column of the `varLabelDf` that contains the variable name. Only needs to be changed from the default value if `varLabelDf` is not a dataframe as produced by `processLSvarLabels`.
`leftAnchorCol`	The name of the column of the `varLabelDf` that contains the left anchor. Only needs to be changed from the default value if `varLabelDf` is not a dataframe as produced by `processLSvarLabels`.
`rightAnchorCol`	The name of the column of the `varLabelDf` that contains the right anchor. Only needs to be changed from the default value if `varLabelDf` is not a dataframe as produced by `processLSvarLabels`.
`subQuestionCol`	The name of the column of the `varLabelDf` that contains the subquestion. Only needs to be changed from the default value if `varLabelDf` is not a dataframe as produced by `processLSvarLabels`.
`questionTextCol`	The name of the column of the `varLabelDf` that contains the question text. Only needs to be changed from the default value if `varLabelDf` is not a dataframe as produced by `processLSvarLabels`.
`names`	A character vector with the variable names. These are matched against the regular expressions as specified in the `determinantStructure` object, and any matches will be stored in the `determinantStructure` object.
`data`	The dataframe containing the data; the variables names specified in `names` (when calling `detStructAddVarNames`) must be present in this dataframe.
`append`	Whether to only return the products or scales, or whether to append these to the dataframe and return the entire dataframe.
`separator`	The separator to use when constructing the scale variables names.

Details

This family of functions will be explained more in detail in a forthcoming paper.

Value

`detStructAddVarLabels` and `detStructAddVarNames` just change the `determinantStructure` object; `detStructComputeProducts` and `detStructComputeScales` return either the dataframe with the new variables appended (if append = `TRUE`) or just a dataframe with the new variables (if append = `FALSE`).

References

(Forthcoming)

See Also

`determinantStructure`, `determinantVar`, `subdeterminants`, `subdeterminantProducts`, `detStructCIBER`

Examples

```
### Create some bogus determinant data
detStudy <- mtcars[, c(1, 3:7)];
names(detStudy) <- c('rUse_behav',
                     'rUse_intention',
                     'rUse_attitude1',
                     'rUse_attitude2',
                     'rUse_expAtt1',
                     'rUse_expAtt2');

### Specify the determinant structure

### First a subdeterminant
expAtt <-
  behaviorchange::subdeterminants("Subdeterminants",
                                  "expAtt");

### Then two determinants
attitude <-
  behaviorchange::determinantVar("Determinant",
                                 "attitude",
                                 expAtt);

intention <-
  behaviorchange::determinantVar("ProximalDeterminant",
                                 "intention",
                                 attitude);

### Then the entire determinant strcture
detStruct <-
  behaviorchange::determinantStructure('Behavior',
                                       list('behav',
                                              behaviorRegEx = 'rUse'),
                                              intention);

### Add the variable names
behaviorchange::detStructAddVarNames(detStruct,
                                     names(detStudy));

### Add the determinant scale variable to the dataframe
detStudyPlus <-
  behaviorchange::detStructComputeScales(detStruct,
                                         data=detStudy);

### Show its presence
names(detStudyPlus);
mean(detStudyPlus$rUse_Determinant);
```

`dMCD` *Estimate Cohen's d corresponding to a Meaningful Change Definition*

Description

This function uses a base rate (Control Event Rate, argument `cer`) and a Meaningful Change Definitions (MCD, argument `mcd`) to compute the corresponding Cohen's d. See Gruijters & Peters (2019) for details.

Usage

```
dMCD(
  cer,
  mcd = NULL,
  eer = NULL,
  plot = TRUE,
  mcdOnX = FALSE,
  plotResultValues = TRUE,
  resultValueLineColor = "blue",
  resultValueLineSize = 1,
  returnLineLayerOnly = FALSE,
  theme = ggplot2::theme_bw(),
  highestPossibleEER = 0.999999999,
  xLab = ifelse(mcdOnX, "Meaningful Change Definition", "Control Event Rate"),
  yLab = "Cohen's d",
  dist = "norm",
  distArgs = list(),
  distNS = "stats",
  ...
)
```

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

Arguments

`cer`	The Control Event Rate (or base rate): how many people already perform the target behavior in the population (as a proportion)?
`mcd`	The Meaningful Change Definitions: by which percentage (as a proportion) should the event rate increase to render an effect meaningful?
`eer`	Instead of the MCD, it is also possible to specify the Experimental Event Rate (EER), in which case the MCD is computed by taking the difference with the CER.
`plot`	Whether to show a plot.
`mcdOnX`	Whether to plot the Meaningful Change Definition on the X axis (by default, the CER is plotted on the X axis).
`plotResultValues`	Whether to plot the result values.
`resultValueLineColor, resultValueLineSize`	If plotting the result values, lines of this color and size are used.
`returnLineLayerOnly`	Whether to only return a layer with the plotted line (which can be used to quickly stack lines for different MCDs).
`theme`	The `ggplot2` theme to use.
`highestPossibleEER`	The highest possible EER to include in the plot.
`xLab, yLab`	The labels for the X and Y axes.
`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.
`...`	Any additional arguments to dMCD are passed on to the `ggplot2::geom_line` used to draw the line showing the different Cohen's d values as a function of the base rate (or MCD) on the X axis. Additional arguments for the print method are passed on to the default print method.
`x`	The object to print (i.e. a result from a call to dMCD).

Value

The Cohen's d value, optionally with a `ggplot2` plot stored in an attribute (which is only a `ggplot2` layer if `returnLineLayerOnly=TRUE`).

References

Gruijters, S. L. K., & Peters, G.-J. Y. (2020). Meaningful change definitions: Sample size planning for experimental intervention research. *PsyArXiv*. doi: 10.31234/osf.io/jc295

Examples

```
dMCD(.2, .05);
```

---

lm_rSq_ci	*Obtaining an R squared confidence interval estimate for an lm regression*

Description

The lm_rSq_ci function uses the base R lm function to conduct a regression analysis and then computes the confidence interval for R squared.

Usage

```
lm_rSq_ci(
  formula,
  data = NULL,
  conf.level = 0.95,
  ci.method = c("widest", "r.con", "olkinfinn"),
  env = parent.frame()
)
```

Arguments

formula	The formula of the regression analysis, of the form y ~ x1 + x2, where y is the dependent variable and x1 and x2 are the predictors.
data	If the terms in the formula aren't vectors but variable names, this should be the dataframe where those variables are stored.
conf.level	The confidence of the confidence interval around the regression coefficients.
ci.method	Which method to use for the confidence interval around R squared.
env	The enviroment where to evaluate the formula.

Value

The confidence interval

Author(s)

Gjalt-Jorn Peters

Maintainer: Gjalt-Jorn Peters gjalt-jorn@a-bc.eu

Examples

```
### Do a simple regression analysis
lm_rSq_ci(age ~ circumference, dat=Orange);
```

nnc *Numbers Needed for Change*

Description

This function computes the Numbers Needed for Change, and shows a visualisation to illustrate them. Numbers Needed for Change is the name for a Numbers Needed to Treat estimate that was computed for a continuous outcome as is common in behavior change research.

Usage

```
nnc(
  d = NULL,
  cer = NULL,
  r = 1,
  n = NULL,
  threshold = NULL,
  mean = 0,
  sd = 1,
  poweredFor = NULL,
  thresholdSensitivity = NULL,
  eventDesirable = TRUE,
  eventIfHigher = TRUE,
  conf.level = 0.95,
  dReliability = 1,
  d.ci = NULL,
  cer.ci = NULL,
  r.ci = NULL,
  d.n = NULL,
  cer.n = NULL,
  r.n = NULL,
  plot = TRUE,
  returnPlot = TRUE,
  silent = FALSE
)

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

Arguments

`d`	The value of Cohen's *d*.
`cer`	The Control Event Rate.
`r`	The correlation between the determinant and behavior (for mediated Numbers Needed for Change).
`n`	The sample size.

`threshold`	If the event rate is not available, a threshold value can be specified instead, which is then used in conjunction with the mean (`mean`) and standard deviation (`sd`) and assuming a normal distribution to compute the event rate.
`mean`	The mean value, used to draw the plot, or, if no CER is provided but instead the threshold value, to compute the CER.
`sd`	The standard deviation, used to draw the plot (and to compute the CER if a threshold value is supplied instead of the CER).
`poweredFor`	The Cohen's *d* value for which the study was powered. This expected Cohen's *d* value can be used to compute the threshold, which then in turn is used to compute the CER. To use this approach, also specify the mean and the standard deviation.
`thresholdSensitivity`	This argument can be used to provide a vector of potential threshold values, each of which is used to compute an NNC. This enables easy inspection of whether the value chosen as threshold matters much for the NNC.
`eventDesirable`	Whether an event is desirable or undesirable.
`eventIfHigher`	Whether scores above or below the threshold are considered 'an event'.
`conf.level`	The confidence level of the confidence interval.
`dReliability`	If Cohen's d was not measured with perfect reliability, `nnc` can disattenuate it to correct for the resulting attenuation using `ufs::disattenuate.d()` before computating the Experimental Event Rate. Use this argument to specify the reliability of the outcome measure. By default, the setting of 1 means that no disattenuation is applied.
`d.ci`	Instead of providing a point estimate for Cohen's *d*, a confidence interval can be provided.
`cer.ci`	Instead of providing a point estimate for the Control Event Rate, a confidence interval can be provided.
`r.ci`	Instead of providing a point estimate for the correlation, a confidence interval can be provided.
`d.n`	In addition to providing a point estimate for Cohen's *d*, a sample size can be provided; if it is, the confidence interval is computed.
`cer.n`	In addition to providing a point estimate for the Control Event Rate, a sample size can be provided; if it is, the confidence interval is computed.
`r.n`	In addition to providing a point estimate for the correlation, a sample size can be provided; if it is, the confidence interval is computed.
`plot`	Whether to generate and show the plot.
`returnPlot`	Whether to return the plot (as an attribute), or to only display it.
`silent`	Whether to suppress notifications.
`x`	The `nnc` object to print.
`digits`	The number of digits to round to.
`...`	Any additional arguments are passed to the `print` function.

Details

Numbers Needed to Treat is a common and very useful effect size measure in use in the medical sciences. It is computed based on the Control Event Rate (CER) and the Experimental Event Rate (EER), and expresses how many people would need to received a treatment to yield a beneficial result for one person. In behavior change research, a similar measure would be useful, but the outcome is normally not dichotomous as is common in the medical literature (i.e. whether a participants survives or is cured), but continuous. Numbers Needed for Change fills this lacuna: it is simply the Numbers Needed to Treat, but converted from a Cohen's d value. `nnt` is an alias for `nnc`.

For more details, see Gruijters & Peters (2019) for details.

Value

The Numbers Needed for Change (NNC), potentially with a plot visualising the NNC in an attribute.

Author(s)

Gjalt-Jorn Peters & Stefan Gruijters

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

References

Gruijters, S. L., & Peters, G. Y. (2019). Gauging the impact of behavior change interventions: A tutorial on the Numbers Needed to Treat. *PsyArXiv.* doi:10.31234/osf.io/2bau7

Examples

```
### Simple example
behaviorchange::nnc(d=.4, cer=.3);

### Or for a scenario where events are undesirable, and the
### intervention effective (therefore having a negative value for d):
behaviorchange::nnc(d=-.4, cer=.3, eventDesirable=FALSE);
```

`opts` *Options for the behaviorchange package*

Description

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

Usage

```
opts
```

Format

An object of class `list` of length 4.

Details

It is normally not necessary to get or set `behaviorchange` options.

The following arguments can be passed:

... For `behaviorchange::opts$set`, the dots can be used to specify the options to set, in the format `option = value`, for example, `EFFECTSIZE_POINTESTIMATE_NAME_IN_DF = "\n"`. For `behaviorchange::opts$reset`, a list of options to be reset can be passed.

option For `behaviorchange::opts$set`, the name of the option to set.

default For `behaviorchange::opts$get`, the default value to return if the option has not been manually specified.

The following options can be set:

The name of the column with the effect size values.

The name of the column with the effect size variance.

The name of the column with the missing values.

Examples

```
### Get the default utteranceMarker
behaviorchange::opts$get(complecs_entitySheet);

### Set it to a custom version, so that every line starts with a pipe
behaviorchange::opts$set(complecs_entitySheet = "sheet_with_entities");

### Check that it worked
behaviorchange::opts$get(complecs_entitySheet);

### Reset this option to its default value
behaviorchange::opts$reset(complecs_entitySheet);

### Check that the reset worked, too
behaviorchange::opts$get(complecs_entitySheet);
```

`partypanelData` *Subsets of Party Panel datasets*

Description

These are subsets of Party Panel datasets. Party Panel is an annual semi-panel determinant study among Dutch nightlife patrons, where every year, the determinants of another nightlife-related risk behavior are mapped.

Usage

```
data(BBC_pp15.1)

data(BBC_pp16.1)

data(BBC_pp17.1)

data(BBC_pp18.1)
```

Format

For BBC_pp15.1, a `data.frame` with 123 columns and 829 rows. For BBC_pp16.1, a `data.frame` with 63 columns and 1077 rows. For BBC_pp17.1, a `data.frame` with 94 columns and 943 rows. For BBC_pp18.1, a `data.frame` with 84 columns and 880 rows. Note that many rows contain missing values; the columns and rows were taken directly from the original Party Panel datasets, and represent all participants that made it past a given behavior.

Details

The behaviors of the Party Panel waves were:

- 2015: Behaviors related to using highly dosed ecstasy pills
- 2016: Behaviors related to visiting nightlife first-aid facilities
- 2017: Behaviors related to hearing protection
- 2018: Behaviors related to flirting and boundary crossing
- 2019: Behaviors related to sleeping hygiene surrounding nightlife participation

The full datasets are publicly available through the Open Science Framework (https://osf.io/s4fmu/). Also see the GitLab repositories (https://gitlab.com/partypanel) and the website at https://partypanel.eu.

Examples

```
data('BBC_pp17.1', package='behaviorchange');
behaviorchange::CIBERlite(data=BBC_pp17.1,
                          determinants=c("epw_attitude",
                                         "epw_perceivedNorm",
                                         "epw_pbc",
                                         "epw_habit"),
                          targets=c("epw_intention"));
```

`pies` *Practically Important Effect Sizes*

Description

Practically Important Effect Sizes

Usage

```
pies(
  data = NULL,
  controlCol = NULL,
  expCol = NULL,
  d = NULL,
  cer = NULL,
  r = 1,
  n = NULL,
  threshold = NULL,
  mean = 0,
  sd = 1,
  bootstrapA = FALSE,
  conf.level = 0.95
)
```

Arguments

`data`	Optionally, if you want to get A, a data frame.
`controlCol, expCol`	Optionally, if you want to get A, the names of the columns with control and experimental data.
`d`	Cohen's d.
`cer`	The control even rate (see `behaviorchange::nnt()`).
`r, threshold, mean, sd`	Arguments for the `behaviorchange::nnt()` function.
`n`	The sample size.
`bootstrapA`	Whether to use bootstrapping to compute A.
`conf.level`	The confidence level of confidence intervals.

Value

A dataframe with all values.

Examples

```
pies(d = .5, n = 100, cer = .2, threshold = 2);
```

`vecTxt` *Easily parse a vector into a character value*

Description

Easily parse a vector into a character value

Usage

```
vecTxt(
  vector,
  delimiter = ", ",
  useQuote = "",
  firstDelimiter = NULL,
  lastDelimiter = " & ",
  firstElements = 0,
  lastElements = 1,
  lastHasPrecedence = TRUE
)

vecTxtQ(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));
```

Index