Package: pcir (via r-universe)

May 17, 2026

Title Potential for Conflict Index in R

Version 0.0.0.9000

Description Provides functions to calculate, compare, and visualize the Potential for Conflict Index (PCI), a descriptive statistic that summarizes the level of agreement or disagreement among stakeholders.

License MIT + file LICENSE

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.2

Imports Hmisc, dplyr, ggplot2, magrittr, tidyr, stats, utils, devtools, roxygen2, rprojroot

Suggests testthat, tibble

Config/Needs/website true

URL https://github.com/fblpalmeira/pcir, https://fblpalmeira.github.io/pcir

BugReports https://github.com/fblpalmeira/pcir/issues

Date 2025-05-21

Config/pak/sysreqs cmake libfontconfig1-dev libfreetype6-dev libfribidi-dev git make libharfbuzz-dev libgit2-dev libicu-dev libjpeg-dev libpng-dev libtiff-dev libuv1-dev libwebp-dev libxml2-dev libssl-dev libx11-dev zlib1g-dev

Repository https://ropensci-champions.r-universe.dev

Date/Publication 2025-05-21 19:07:06 UTC

RemoteUrl https://github.com/fblpalmeira/pcir

RemoteRef HEAD

RemoteSha 2481ecb1bad84c93cc4743840a81fdb4f43b5734

Contents

`bubble` *Create a Bubble Plot for PCI Visualization*

Description

This function generates a bubble plot to visualize the results of the PCI calculation. It shows the mean action acceptability on the y-axis and the PCI value as the size of the bubbles.

Create a bubble plot to visualize PCI results. This function generates a bubble plot to visualize PCI results. Each bubble represents an item, with size proportional to total responses, x-axis representing weighted mean, and color representing the PCI value.

Usage

```
bubble(
  data,
  scale_type = c("bipolar_with_neutral", "bipolar_without_neutral", "unipolar"),
  ylim_range = 4,
  unipolar_ylim = c(1, 9),
  xlab = "",
  ylab = "Action acceptability",
  title = NULL,
  bubble_color = "gray80",
  bubble_stroke = "black",
  x_line = NULL
)

bubble(
  data,
  scale_type = c("bipolar_with_neutral", "bipolar_without_neutral", "unipolar"),
  ylim_range = 4,
  unipolar_ylim = c(1, 9),
  xlab = "",
  ylab = "Action acceptability",
  title = NULL,
  bubble_color = "gray80",
  bubble_stroke = "black",
  x_line = NULL
)
```

Arguments

data	A data frame containing at least the columns name, Mean, and PCI.
scale_type	The type of response scale. One of 'bipolar_with_neutral', 'bipolar_without_neutral', or 'unipolar'.
ylim_range	Numeric value to define y-axis range for bipolar scales.
unipolar_ylim	A numeric vector of length 2 specifying y-axis limits for unipolar scale.
xlab	Label for the x-axis.
ylab	Label for the y-axis.
title	Title for the plot.
bubble_color	Fill color for the bubbles.
bubble_stroke	Border color for the bubbles.
x_line	Position of the horizontal reference line.

Details

This plot is useful for visualizing how polarized (PCI) and how acceptable (Mean) each action/item is.

Value

A ggplot2 object representing the bubble plot.

A ggplot2 object.

Examples

```
data <- data.frame(
  name = c('A', 'B', 'C'),
  Mean = c(0.5, -1, 1.2),
  PCI = c(0.2, 0.5, 0.8)
)
bubble(data, scale_type = 'bipolar_with_neutral')
```

---

counting *Create a Count Table with Percentages, Mean, and SD*

Description

This function takes a data frame and computes the count and percentage of each unique value across specified columns. It also calculates the weighted mean and weighted standard deviation for each variable.

Create a count table with percentages, mean, and standard deviation. This function takes a data frame, computes summary statistics like counts, percentages, means, and standard deviations for each unique value in the selected columns.

Usage

```
counting(df1, cols)

counting(df1, cols)
```

Arguments

df1	A data frame containing the data to be processed.
cols	A vector of column names to be included in the calculation.

Details

The output is useful for summarizing responses from ordinal or Likert-type items, showing how values are distributed and summarized across variables.

Value

A data frame in wide format including:

- Counts and percentages for each unique value.
- Weighted mean and standard deviation.

A data frame in wide format including:

- Count and percentage for each value per variable
- Weighted mean
- Weighted standard deviation (SD)
- Total number of observations per variable

Examples

```
df1 <- data.frame(
  A = c(-1, -1, -1, 0, -1),
  B = c(-1, 1, 0, -1, 1),
  C = c(0, 0, 1, 0, -1),
  D = c(0, -1, 1, 1, 1),
  E = c(1, 1, 0, -1, -1)
)
counting(df1, cols = c('A', 'B', 'C', 'D', 'E'))
```

pci *Calculate the Potential for Conflict Index (PCI2)*

Description

This function computes the PCI2 (a generalized Potential for Conflict Index) for each item in a summarized data frame (typically from the `counting()` function), or for a single named numeric vector of counts. It works with three types of scales: `bipolar_with_neutral`, `bipolar_without_neutral`, and `unipolar`.

Calculate the Potential for Conflict Index (PCI). This function computes the PCI2 (a generalized Potential for Conflict Index) for each item in a summarized data frame (typically from the `counting()` function), or for a single named numeric vector of counts. It works with three types of scales: `bipolar_with_neutral`, `bipolar_without_neutral`, and `unipolar`.

Usage

```
pci(
  data,
  scale_type = c("bipolar_with_neutral", "bipolar_without_neutral", "unipolar"),
  min_scale_value,
  max_scale_value,
  exponent = 1
)

pci(
  data,
  scale_type = c("bipolar_with_neutral", "bipolar_without_neutral", "unipolar"),
  min_scale_value,
  max_scale_value,
  exponent = 1
)
```

Arguments

`data`	A data frame with columns named like 'Count X' (where X is a scale value), or a named numeric vector of counts.
`scale_type`	Type of scale: `'bipolar_with_neutral'`, `'bipolar_without_neutral'`, or `'unipolar'`.
`min_scale_value`	The minimum value of the scale (e.g., -2 or 1).
`max_scale_value`	The maximum value of the scale (e.g., 2 or 9).
`exponent`	A number to raise distances to. Use `1` for linear distance, `2` for squared distance, etc. Default is `1`.

Details

For unipolar scales, PCI2 is based on all pairwise distances between categories.

For bipolar scales, PCI2 considers distances between opposing poles, ignoring neutral (0) in the case of `bipolar_without_neutral`. In `bipolar_with_neutral`, 0 is included in the scale but not in the calculation of conflict.

The maximum possible polarization (used for normalization) assumes a perfectly split distribution.

For unipolar scales, PCI2 is based on all pairwise distances between categories.

For bipolar scales, PCI2 considers distances between opposing poles, ignoring neutral (0) in the case of `bipolar_without_neutral`. In `bipolar_with_neutral`, 0 is included in the scale but not in the calculation of conflict.

The maximum possible polarization (used for normalization) assumes a perfectly split distribution.

Value

If `data` is a data frame, it returns the same data frame with an added `PCI` column. If `data` is a named numeric vector, it returns the PCI value directly.

If `data` is a data frame, it returns the same data frame with an added `PCI` column. If `data` is a named numeric vector, it returns the PCI value directly.

Examples

```
df1 <- data.frame(
  A = c(-1, -1, -1, 0, -1),
  B = c(-1, 1, 0, -1, 1),
  C = c(0, 0, 1, 0, -1),
  D = c(0, -1, 1, 1, 1),
  E = c(1, 1, 0, -1, -1)
)
df2 <- counting(df1, cols = c('A', 'B', 'C', 'D', 'E'))
pci(df2, scale_type = 'bipolar_with_neutral', min_scale_value = -1, max_scale_value = 1)

df1 <- data.frame(
  A = c(-1, -1, -1, 0, -1),
  B = c(-1, 1, 0, -1, 1),
  C = c(0, 0, 1, 0, -1),
  D = c(0, -1, 1, 1, 1),
  E = c(1, 1, 0, -1, -1)
)
df2 <- counting(df1, cols = c('A', 'B', 'C', 'D', 'E'))
pci(df2, scale_type = 'bipolar_with_neutral', min_scale_value = -1, max_scale_value = 1)
```

`reinstall_and_test_addin` *Reinstall and Test the pcir Package*

Description

This RStudio Addin reinstalls the 'pcir' package from the local source, reloads it, and runs all functions on example data for testing.

Usage

```
reinstall_and_test_addin()
```

Index