Package 'drhutools'

June 4, 2026

Type Package

Title Political Science Academic Research Gears

Version 1.1.1

Maintainer Yue Hu <yuehu@tsinghua.edu.cn>

Description Using these tools to simplify the research process of political science and other social sciences. The current version can create folder system for academic project in political science, calculate psychological trait scores, visualize experimental and spatial data, set up color-blind palette, and test for Type I error (false positives) in Qualitative Comparative Analysis (QCA) for crisp-set, multi-value, and fuzzy-set variants.

URL https://www.drhuyue.site/software/drhutools/

BugReports https://github.com/sammo3182/drhutools/issues

License GPL (>= 3)

Encoding UTF-8

Depends R (>= 4.1.0)

Imports dplyr, ggplot2, purrr, htmltools, sf, htmlwidgets, jsonlite, leaflet, sp, stats, methods, gganimate, magick, webshot, animation, png, grDevices, graphics

LazyData true

Suggests knitr, remotes, rmarkdown

VignetteBuilder knitr

Config/roxygen2/version 8.0.0

NeedsCompilation no

Author Yue Hu [aut, cre],
Qian Qiu [ctb],
Wen Deng [ctb],
Bear Braumoeller [ctb] (QCAfalsePositive functions)

Repository CRAN

Date/Publication 2026-06-04 04:20:02 UTC

Contents

`Arab.Spring` *Determinants of the Arab Spring Uprising*

Description

A dataset containing fuzzy-set membership scores for ten sets of state characteristics across 20 states in northern Africa and the Arab Peninsula at the time of the Arab Spring uprisings of 2010–12.

Usage

```
Arab.Spring
```

Format

A data frame with 20 rows and 10 variables, measured as fuzzy-set membership scores:

Gdppc Per-capita GDP.

Gini Economic inequality.

Unemp Unemployment.

Urban Degree of urbanization.

Youth Size of youth bulge.

Mobile Mobile phone usage.

Internet Internet penetration.

Fuel Fuel-dependence of economy.

Pol Regime fragility.

Success Social movement success.

Value

A data.frame object

Source

Hussain, Muzammil M., and Philip N. Howard. 2013. "What Best Explains Successful Protest Cascades? ICTs and the Fuzzy Causes of the Arab Spring." *International Studies Review* 15(1): 48–66.

`cdplot` *Visualizing the experimental outcome with cumulative distribution functions.*

Description

Packed `ggplot2` function to compare the empirical cumulative distribution functions (ECDF) between the treatment and control groups in an experiment or quasi-experiment.

Arguments

`data`	A data.frame including two columns, one recording the outcome and the other recording the assignment. The assignment column must be named as `group`.
`ks_test`	A logical option to indicate whether to show the Kolmogorov-Smirnov test result in the bottom-right corner. The default value is FALSE.
`point_size`	An integer to indicate the size of the points at the largest difference. The default value is 3.
`point_color`	An character or function to indicate the color of the points at the largest difference. The default value is `drhutools::gb_cols("red")`.
`link_color`	An character or function to indicate the color of the link at the largest difference. The default value is `drhutools::gb_cols("red")`.

Value

A list of `ggplot2` objects comparing the ECDFs between the control and treatment groups and identifying at most three largest differences.

Examples

```
data("PlantGrowth")

plot_plant <- cdplot(PlantGrowth, ks_test = TRUE)
plot_plant
```

`csQCAbinTest` *A Simple Binomial Test for Type I Error in csQCA*

Description

A Binomial test for crisp-set qualitative comparative analysis (csQCA), designed to calculate the probability of a false positive given the number of hypotheses implicitly tested and the number of confirming cases.

Usage

```
csQCAbinTest(freq.y, configs, total.configs, adj.method = "holm")
```

Arguments

`freq.y`	The frequency with which the dependent variable occurs in the sample (the number of 1s divided by the total number of cases).
`configs`	A list of configurations and the number of cases in which each configuration occurs.
`total.configs`	The total number of configurations used in the original csQCA analysis. This will generally equal the number of lines in the truth table used for Boolean minimization.
`adj.method`	The method used to calculate adjusted p-values (see `stats::p.adjust()` for details).

Value

An object containing the results of the Binomial test.

Note

Adopted from the archived CRAN package **QCAfalsePositive** by Bear Braumoeller. The original package has been removed from CRAN and is no longer maintained. It is included in **drhutools** for continued accessibility.

Examples

```
test <- csQCAbinTest(freq.y=0.7, configs=list(aB=5, bCD=3, Ce=2),
   total.configs=20)
summary(test)
```

`folderSystem` *Folder hierarchy creator for academic research*

Description

Folder hierarchy creator for academic research

Usage

```
folderSystem(overwrite = FALSE)
```

Arguments

`overwrite`	Logical. Should existing template files in `paper/` be overwritten? Defaults to `FALSE`, leaving any files the user has already edited untouched.

Details

The function constructs a standardized folder hierarchy encompassing `codes/`, `data/`, `output/`, `paper/`, and `document/`. These directories are designated for storing programming scripts, data sets, processed outputs, manuscript drafts (including related images and submission documents), and assorted materials, respectively. This organization facilitates easy retrieval of project components, ensuring that members of Dr. Hu's Amazing Team can efficiently navigate and manage shared resources. This structure is advocated for collaborative projects to maintain uniformity across team operations.

In addition, the manuscript-writing templates bundled with the package (a Quarto/Typst paper template, its main and appendix bibliographies, the OSCOLA citation style, and the citation-prefix Lua filter) are copied into the `paper/` folder so a new project starts ready to write.

Value

Invisibly returns `NULL`. Called for its side effect of creating local folders, placeholder files, a default `.gitignore`, and the manuscript templates under the project's working directory.

`fsQCApermTest` *A Simple Permutation Test for Type I Error in fsQCA*

Description

A permutation test for fuzzy-set qualitative comparative analysis (fsQCA), designed to calculate the probability of a false positive given the number of hypotheses implicitly tested and the distribution of the data.

Usage

```
fsQCApermTest(
  y,
  configs,
  total.configs,
  num.iter = 10000,
  my.seed = 123,
  adj.method = "holm"
)
```

Arguments

y	The outcome variable of interest.
configs	A list of configurations to be tested against y.
total.configs	The total number of configurations used in the original fsQCA analysis. This will generally equal the number of lines in the truth table used for Boolean minimization.
num.iter	The number of iterations to use for the permutation test. Larger numbers of iterations result in more precise p-values.
my.seed	The seed used to generate random numbers.
adj.method	The method used to calculate adjusted p-values (see stats::p.adjust() for details).

Value

An object containing the aggregate results of the permutation test as well as the individual permutations.

Note

Adopted from the archived CRAN package **QCAfalsePositive** by Bear Braumoeller. The original package has been removed from CRAN and is no longer maintained. It is included in **drhutools** for continued accessibility.

Examples

```
intersect  <- pmin(social.revolutions$breakdown, social.revolutions$pop.ins)
intersect2 <- pmin(social.revolutions$breakdown, (1 - social.revolutions$pop.ins))
intersect3 <- pmin((1 - social.revolutions$breakdown), social.revolutions$pop.ins)
intersect4 <- pmin((1 - social.revolutions$breakdown), (1 - social.revolutions$pop.ins))

test <- fsQCApermTest(y = social.revolutions$soc.rev,
  configs = list(BI = intersect, Bi = intersect2,
                 bI = intersect3, bi = intersect4),
  total.configs = 4)
summary(test)
plot(test)
```

gb_cols *Function to extract colors as hex codes.*

Description

Function to extract colors as hex codes.

Usage

```
gb_cols(...)
```

Arguments

...	A character vector indicating the names of colors. Options includes "gold", "black", "orange", "red", "light green", "green", "light blue", "blue", "light grey", "grey", and "dark grey".

Value

A character of hex codes.

gb_pal *Function to interpolate a gold-black color palette.*

Description

Function to interpolate a gold-black color palette.

Usage

```
gb_pal(palette = "main", reverse = FALSE, ...)
```

Arguments

palette	A character vector indicating the name of palette in gb_palettes. Available palettes: • `main`: Gold and black colors. • `tricol`: Gold, black, and dark grey to create a gradual effect. • `digitMixed`: Five-pack colors specified for digital publications. • `printMixed`: Five-pack colors specified for printed publications. • `full`: A palette including all the colors `gb_cols` can call.
reverse	A logic vector indicating whether the palette should be reversed; the default is FALSE.
...	Additional arguments to pass to `colorRampPalette()`

Value

A function that takes an integer argument (the required number of colors) and returns a character vector of colors interpolating the given sequence.

---

goodmap — *The* goodmap *function is designed to create interactive PNG Map or GIF Map from a provided data file. It supports two types of maps:* point *and* polygon*. The function can visualize data by either plotting points based on geographical coordinates or highlighting regions polygon based on their administrative boundaries (province or city level). Additionally, the function can generate animated that showcase the change of data.*

Description

If the map type is point, the color and size of the points will be determined by the value_set column in the data file, which means the different value of each point. If the map type is polygon, the color of the polygons will be determined by the average value of the value_set column for each city or province in the data file.

Usage

```
goodmap(
  data_file,
  type = "point",
  level = NULL,
  animate = FALSE,
  animate_var = NULL,
  map_center = c(35.8617, 104.1954),
  zoom_level = 4,
  tile_source = "amap",
  coord = "WGS-84",
  color_type = "numeric",
  custom_colors = NULL,
  point_radius = 5,
  legend_opacity = 0.7,
  legend_name = NULL,
  width = 800,
  height = 900
)
```

Arguments

data_file	Dataframe. When generate point map, data_file should include required columns such as g_lat and g_lon. When generate polygon map, data_file should include required columns such as prov or city. The prov columns must be complete, official names rather than any shortened form or abbreviation. If there is

only incomplete names or geocodes in your data_file, we recommend you to use function `regioncode` as a one-step solution to these conversion from incomplete names. Ensure the file is formatted correctly with appropriate column headers.

`type` A string specifying the type of map to generate. Options are `point`for point maps using `g_lat` and `g_lon`, or `polygon` for maps with administrative boundaries.

`level` A string specifying the level of administrative boundaries for polygon maps. Acceptable values are `province` or `city`. This parameter is required if `type` is `polygon`.

`animate` A logical value indicating whether to generate an animation from the maps. The default is FALSE. If animate is FALSE, the whole data will be generated as a PNG file. If animate is TRUE, an animation will be generated from all panel data, the animate_var must be assigned.

`animate_var` A string specifying the variable to animate over. The default is `NULL`, the whole data will be generated as a PNG file. If an animation is needed, `data_file` should include required the `animate_set` column. `animate_set` includes a series of unique values that represents year or month, or other categorical variable. The frame number of the animation depends on the `animate_set`.

`map_center` A numeric vector of length 2 specifying the latitude and longitude for the center of the map view. Default is `c(35.8617, 104.1954)`, which is approximately the center of China.

`zoom_level` A numeric value specifying the zoom level for the map. Default is 3.

`tile_source` A string specifying the basemap tile provider. Options are `amap` for the Gaode/Amap Chinese basemap (default), or `osm` for OpenStreetMap. Amap offers fuller Chinese labelling and coverage; note its tiles use the GCJ-02 coordinate system (see `coord`). Baidu is not supported because its BD-09 projection is incompatible with the standard leaflet tile grid.

`coord` A string giving the coordinate system of the **point** input data (`g_lat`/`g_lon`). Options are `WGS-84` (default, GPS/international geocoders), `GCJ-02` (Amap/Gaode/Tencent geocoders), or `BD-09` (Baidu geocoder). Points are reprojected to match `tile_source` so markers align with the basemap. This argument is ignored for polygon maps, whose boundaries are bundled in GCJ-02 and align with the Amap basemap.

`color_type` If the data is discrete, such as types or categories, choose `factor`. If the data is continuous, such as temperature or pressure, choose `numeric`. Default is numeric.

`custom_colors` A vector of colors for customizing the color gradient. Default is `NULL`, which uses the predefined color palette.

`point_radius` A numeric value specifying the radius for point markers on point maps. Default is 5.

`legend_opacity` A numeric value specifying the opacity of the legend. Default is 0.7.

`legend_name` The name of the legend. Default is `Value`.

`width` A numeric value specifying the width of the map images. Default is 800.

`height` A numeric value specifying the height of the map images. Default is 900.

Value

Image in the viewer.

Examples

```
## Not run:
goodmap(
   toy_poly,
   type = "polygon",
   level = "province"
)

## End(Not run)
```

---

leafletcn.map.names *Leaflet Map Names Dataset*

Description

A internal dataset providing city names and corresponding file names for mapping with Leaflet.

Usage

```
leafletcn.map.names
```

Format

A data frame with city names and file names.

Value

A data.frame object

---

mvQCAbinTest *A Simple Binomial Test for Type I Error in mvQCA*

Description

A Binomial test for multi-value qualitative comparative analysis (mvQCA), designed to calculate the probability of a false positive given the number of hypotheses implicitly tested and the number of confirming cases.

Usage

```
mvQCAbinTest(freq.y, configs, total.configs, adj.method = "holm")
```

Arguments

`freq.y`	The frequency with which the dependent variable occurs in the sample (the number of 1s divided by the total number of cases).
`configs`	A list of configurations and the number of cases in which each configuration occurs.
`total.configs`	The total number of configurations used in the original mvQCA analysis. This will generally equal the number of lines in the truth table used for Boolean minimization.
`adj.method`	The method used to calculate adjusted p-values (see `stats::p.adjust()` for details).

Value

An object containing the results of the Binomial test.

Note

Adopted from the archived CRAN package **QCAfalsePositive** by Bear Braumoeller. The original package has been removed from CRAN and is no longer maintained. It is included in **drhutools** for continued accessibility.

Examples

```
test <- mvQCAbinTest(freq.y=0.7, configs=list(aB=5, bCD=3, Ce=2),
   total.configs=20)
summary(test)
```

`p.threshold.adjust` *Calculate Vector of p-value Thresholds for Multiple Inference*

Description

Internal function. Calculates the adjusted thresholds for multiple significance tests, assuming that the original threshold for a single test is p<0.05. Used by `fsQCApermTest()` to calculate confidence intervals.

Usage

```
p.threshold.adjust(total.configs, my.method)
```

Arguments

`total.configs`	The total number of hypotheses tested, or the number of configurations utilized by the Quine-McCluskey algorithm in fsQCA (including logical remainders, if they are used in the analysis).
`my.method`	The adjustment method used to calculate p-values (see `stats::p.adjust()` for details).

Value

Numeric vector giving adjusted p-value thresholds, from smallest to largest.

Note

Adopted from the archived CRAN package **QCAfalsePositive** by Bear Braumoeller. The original package has been removed from CRAN and is no longer maintained. It is included in **drhutools** for continued accessibility.

Examples

```
p.threshold.adjust(10, "holm")
```

`plot.fsQCApt` *Plot Results of fsQCA Permutation Test*

Description

Plots distributions of consistencies and counterexamples from permutation tests of fsQCA data, including confidence intervals adjusted to account for multiple inference. Also prints observed consistency values and number of counterexamples as black dots along the x-axis, for comparison.

Usage

```
## S3 method for class 'fsQCApt'
plot(x, y = x$config.names, statistic = "both", ...)
```

Arguments

`x`	Object returned by `fsQCApermTest()`.
`y`	A vector of configurations to examine. Default behavior is to examine all configurations.
`statistic`	The statistic to examine (`consistency`, `counterexamples`, or `both`).
`...`	Additional parameters to pass on.

Value

Plots of distributions of consistencies, counterexamples, or both.

Note

Adopted from the archived CRAN package **QCAfalsePositive** by Bear Braumoeller. The original package has been removed from CRAN and is no longer maintained. It is included in **drhutools** for continued accessibility.

Examples

```
intersect  <- pmin(social.revolutions$breakdown, social.revolutions$pop.ins)
intersect2 <- pmin(social.revolutions$breakdown, (1 - social.revolutions$pop.ins))
intersect3 <- pmin((1 - social.revolutions$breakdown), social.revolutions$pop.ins)
intersect4 <- pmin((1 - social.revolutions$breakdown), (1 - social.revolutions$pop.ins))

test <- fsQCApermTest(y = social.revolutions$soc.rev,
  configs = list(BI = intersect, Bi = intersect2,
                 bI = intersect3, bi = intersect4),
  total.configs = 4)
plot(test)
plot(test, "bi", statistic = "consistency")
plot(test, c("BI", "Bi"), statistic = "both")
plot(test, statistic = "consistency")
plot(test, "BI")
```

`scale_color_gb` *Fill scale constructors for gold & black colors*

Description

Fill scale constructors for gold & black colors

Usage

```
scale_color_gb(palette = "main", discrete = TRUE, reverse = FALSE, ...)
```

Arguments

`palette`	A character vector indicating the name of palette in `gb_palettes`. Available palettes: • `main`: Gold and black colors. • `tricol`: Gold, black, and dark grey to create a gradual effect. • `digitMixed`: Five-pack colors specified for digital publications. • `printMixed`: Five-pack colors specified for printed publications. • `full`: A palette including all the colors `gb_cols` can call.
`discrete`	A logic vector indicating whether color aesthetic is discrete or not; the default is "main".
`reverse`	A logic vector indicating whether the palette should be reversed.
`...`	Additional arguments passed to `discrete_scale()` or `scale_color_gradientn()`, used respectively when discrete is TRUE or FALSE

Value

No return value, called for modifying the appearance of the plot.

scale_fill_gb *Color scale constructors for gold & black colors*

Description

Color scale constructors for gold & black colors

Usage

```
scale_fill_gb(palette = "main", discrete = TRUE, reverse = FALSE, ...)
```

Arguments

palette	A character vector indicating the name of palette in gb_palettes. Available palettes: • main: Gold and black colors. • tricol: Gold, black, and dark grey to create a gradual effect. • digitMixed: Five-pack colors specified for digital publications. • printMixed: Five-pack colors specified for printed publications. • full: A palette including all the colors gb_cols can call.
discrete	A logic vector indicating whether color aesthetic is discrete or not; the default is "main".
reverse	A logic vector indicating whether the palette should be reversed.
...	Additional arguments passed to discrete_scale() or scale_color_gradientn(), used respectively when discrete is TRUE or FALSE

Value

No return value, called for modifying the appearance of the plot.

social.revolutions *Determinants of Social Revolutions*

Description

A data frame containing hypothetical fuzzy-set membership scores for three sets of state characteristics across twenty cases.

Usage

```
social.revolutions
```

Format

A data frame with 20 rows and 3 variables, measured as degree of fuzzy-set membership:

soc.rev Presence of social revolution.

breakdown Degree of state breakdown.

pop.ins Presence of popular insurrection.

Value

A data.frame object

Source

Ragin, Charles C. 2000. *Fuzzy-Set Social Science*. Chicago: University of Chicago Press, p. 220.

---

summary.csQCAbt *Summarize Binomial Tests for csQCA Data*

Description

Displays number of confirming cases and raw and adjusted p-scores following Binomial test of csQCA data.

Usage

```
## S3 method for class 'csQCAbt'
summary(object, ...)
```

Arguments

`object`	Object returned by `csQCAbinTest()`.
`...`	Additional parameters to pass on.

Value

Matrix of values for counterexamples and consistency.

Note

Adopted from the archived CRAN package **QCAfalsePositive** by Bear Braumoeller. The original package has been removed from CRAN and is no longer maintained. It is included in **drhutools** for continued accessibility.

Examples

```
test <- csQCAbinTest(freq.y=0.7, configs=list(aB=5, bCD=3, Ce=2),
   total.configs=20)
summary(test)
```

`summary.fsQCApt` *Summarize Permutation Tests for fsQCA Data*

Description

Displays observed values, confidence intervals, and raw and adjusted p-scores for both consistency and counterexamples following permutation test of fsQCA data.

Usage

```
## S3 method for class 'fsQCApt'
summary(object, ...)
```

Arguments

`object`	Object returned by `fsQCApermTest()`.
`...`	Additional parameters to pass on.

Value

Two matrices of values for counterexamples and consistency.

Note

Adopted from the archived CRAN package **QCAfalsePositive** by Bear Braumoeller. The original package has been removed from CRAN and is no longer maintained. It is included in **drhutools** for continued accessibility.

Examples

```
intersect  <- pmin(social.revolutions$breakdown, social.revolutions$pop.ins)
intersect2 <- pmin(social.revolutions$breakdown, (1 - social.revolutions$pop.ins))
intersect3 <- pmin((1 - social.revolutions$breakdown), social.revolutions$pop.ins)
intersect4 <- pmin((1 - social.revolutions$breakdown), (1 - social.revolutions$pop.ins))

test <- fsQCApermTest(y = social.revolutions$soc.rev,
  configs = list(BI = intersect, Bi = intersect2,
                 bI = intersect3, bi = intersect4),
  total.configs = 4)
summary(test)
```

summary.mvQCAbt *Summarize Binomial Tests for mvQCA Data*

Description

Displays number of confirming cases and raw and adjusted p-scores following Binomial test of mvQCA data.

Usage

```
## S3 method for class 'mvQCAbt'
summary(object, ...)
```

Arguments

object	Object returned by mvQCAbinTest().
...	Additional parameters to pass on.

Value

Matrix of values for counterexamples and consistency.

Note

Adopted from the archived CRAN package **QCAfalsePositive** by Bear Braumoeller. The original package has been removed from CRAN and is no longer maintained. It is included in **drhutools** for continued accessibility.

Examples

```
test <- mvQCAbinTest(freq.y=0.7, configs=list(aB=5, bCD=3, Ce=2),
   total.configs=20)
summary(test)
```

toy_poly *Toy Dataset for* goodmap

Description

A sample dataset designed to illustrate the functionality of the goodmap function.

Usage

```
toy_poly
```

Format

A geographic data frame with 6 rows and 5 variables:

id Optional data identifier.

city Prefectural names, required for city-level map plotting.

prov Provincial names, required for provincial-level map plotting.

animate_set A numeric vector used to group data for animated plotting.

value_set Values used to fill each polygon in the map.

Value

A data.frame object

`traits` *Calculate psychological traits based on*

Description

Calculate psychological traits based on

Usage

```
traits(survey)
```

Arguments

`survey`	Psychological survey data as a `data.frame`.

Details

The current version can calculate the TOSCA-3SC scores and Grit-O score.

Value

A data frame with trait scores for each observations

References

Duckworth, Angela L., Christopher Peterson, Michael D. Matthews, and Dennis R. Kelly. 2007. "Grit: Perseverance and Passion for Long-Term Goals." Journal of Personality and Social Psychology 92(6): 1087-1101. doi:10.1037/0022-3514.92.6.1087.

Tangney, June P. 1990. "Assessing Individual Differences in Proneness to Shame and Guilt: Development of the Self-Conscious Affect and Attribution Inventory." Journal of Personality and Social Psychology 59(1): 102-11. doi:10.1037/0022-3514.59.1.102.

Examples

```
column_names <- c("Q3|R3", "Q3|R4", "Q4|R3", "Q4|R4", "Q5|R5", "Q5|R6", "Q6|R3", "Q6|R4", "Q7|R3",
"Q7|R4", "Q8|R5", "Q8|R6", "Q9|R5", "Q9|R6", "Q10|R5", "Q10|R6", "Q11|R5", "Q11|R6", "Q12|R3",
"Q12|R4", "Q13|R3", "Q13|R4", "Q14|1", "Q15|1", "Q16|1", "Q17|1", "Q18|1", "Q19|1", "Q20|1",
"Q21|1", "Q22|1", "Q23|1", "Q24|1", "Q25|1")

toy_data <- data.frame(matrix(sample(1:5, 10 * length(column_names), replace = TRUE),
ncol = length(column_names)))

names(toy_data) <- column_names

traits(toy_data)
```

Index