Package 'qmethod'

March 28, 2023

Version 1.8.4

Date 2023-03-23

Title Analysis of Subjective Perspectives Using Q Methodology

Description Analysis of Q methodology, used to identify distinct perspectives existing within a group. This methodology is used across social, health and environmental sciences to understand diversity of attitudes, discourses, or decision-making styles (for more information, see <https://qmethod.org/>).
A single function runs the full analysis. Each step can be run separately using the corresponding functions: for automatic flagging of Q-sorts (manual flagging is optional), for statement scores, for distinguishing and consensus statements, and for general characteristics of the factors.
The package allows to choose either principal components or centroid factor extraction, manual or automatic flagging, a number of mathematical methods for rotation (or none), and a number of correlation coefficients for the initial correlation matrix, among many other options.
Additional functions are available to import and export data (from raw *.CSV, 'HTMLQ' and 'FlashQ' *.CSV, 'PQMethod' *.DAT and 'easy-htmlq' *.JSON files), to print and plot, to import raw data from individual *.CSV files, and to make printable cards.
The package also offers functions to print Q cards and to generate Q distributions for study administration.
See further details in the package documentation, and in the web pages below, which include a cookbook, guidelines for more advanced analysis (how to perform manual flagging or change the sign of factors), data management, and a graphical user interface (GUI) for online and offline use.

License GPL (>= 2)

Imports methods, psych, tools, digest, knitr, xtable,

Suggests shiny, rjson, MCMCpack

LazyData true

Repository CRAN

URL https://github.com/aiorazabala/qmethod, http://aiorazabala.github.io/qmethod/

BugReports https://github.com/aiorazabala/qmethod/issues

NeedsCompilation no

Author Aiora Zabala [aut, cre] (Main author,
<https://orcid.org/0000-0001-8534-3325>),
Maximilian Held [aut] (Author of additional data management functions),
Frans Hermans [aut] (Author of centroid extraction function)

Maintainer Aiora Zabala <aiora.zabala@gmail.com>

Date/Publication 2023-03-28 10:10:02 UTC

R topics documented:

qmethod-package *Package for Q methodology analysis*

Description

Q is a methodology to study distinct perspectives existing within a group on a topic of interest. It is used across social, health, and environmental studies. See the references below for more details about the methodology.

This package performs the analysis of Q methodology data (both forced and non-forced distributions). The user can choose the extraction method (principal components analysis or centroid factor extraction) and the rotation method (none, varimax or other rotations, uncommon in Q but implemented via `principal`). The default analysis conducts automatic flagging, and manual flagging is optional.

The following steps of the analysis correspond to separate functions: automatic flagging of Q-sorts (`qflag`), z-scores and factor scores for statements (`qzscores`), distinguishing and consensus statements (`qdc`), and general characteristics of the factors (`qfcharact`). The function `qmethod` wraps them all.

The functions for each step may be used separately for advanced analysis, for example, for manual flagging (see details in `qzscores`).

The package also includes additional functions for the following:

- An interactive Graphical User Interface with the basic functionality (`runInterface`)
- Import data from PQMethod software (`import.pqmethod`), both HTMLQ and FlashQ (`import.htmlq`), and easy-htmlq (`import.easyhtmlq`) tools for online data collection of Q-sorts.
- Export a plain-text report of the analysis for interpretation in two flavours (`export.qm`).
- Generic methods to `print.QmethodRes` and `plot.QmethodRes` Q method results. The specific dotchart visualisation of Q results in `plot.QmethodRes` was first developed and introduced in this package, in preparation for the study in Zabala et al. (2017).
- Functions to explore the analysis and facilitate interpretation:
  - explore automatic pre-flagging, (`loa.and.flags`)
  - to rename the factors in the results, with short, meaningful names (`q.fnames`).
- Generate printable cards for the administration of a Q study. The function `make.cards` produces a PDF with full item wording and codes, ready for printout on business card templates that can be easily broken into individual Q-cards.
- Several functions to aid reproducible research, by importing the following from raw, separate *.CSV or *.TEX files for each respondent or item:
  - Q-sorts (`import.q.sorts`)
  - Participant item feedback (`import.q.feedback`)
  - Complete concourses (`import.q.concourse`)
  - Item samples (`build.q.set`)

Use `help(package="qmethod")` for a list of all the functions.

Terminology: The functions for analysis use the terms standard in Q methodology. In addition, the optional functions to import raw data from separate *.CSV files (`import.q.sorts`, `import.q.concourse`, `build.q.set`, `import.q.feedback`) and the card printing function (`make.cards`) refer to items in three distinct ways:

1. Item **full wording**, is the complete item, such as: "'One small community of indomitable Q-methodologists ...'". This item can be read in from individual *.TEX files by using `import.q.concourse`. The wording is not passed on to any other function, but can be readily retrieved from the object returned from `import.q.concourse`.
2. The item **handle** is a shorthand way of referring to an item, which should be *meaningful* to the researcher (e.g. "life-with-q"). Item handles are *researcher-facing* and can be used to refer to items during data *analysis*. They are read in from the *filenames* of individual *.TEX files when using `import.q.concourse`. Handles can be used to identify items in other functions and their outputs. For example, the resulting array or matrix from `import.q.sorts` carries these handles as row names.
3. The item **ID** is another shorthand way of referring to an item, that should be *meaningless* to humans (so as not to influence the participants in unintended ways), such as an arbitrary string of characters. Item IDs are *participant-facing* and are used to identify items during data *entry*. The item ID can take two forms, depending on function arguments specified by the user:
   (a) Standard **IDs** (such as `sta12`, `sta13`) which are generated automatically in `qmethod` or can be provided by the user using the respective `manual.lookup` arguments in `make.cards`, `import.q.sorts` and `import.q.feedback`. See the documentation of these functions for details.
   (b) A set of hexadecimal **hashed IDs** (such as `ae128fs`) can be automatically generated and expected by the functions `make.cards`, `import.q.sorts` and `import.q.feedback` if the argument `manual.lookup` remains empty and defaults to `NULL`. In that case, IDs are computed by 'summarising' the full item wordings (e.g. "'Q Method is used by a crazy, but charming community ...'") into a hexadecimal number (e.g. "ae128fs"), a process known as cryptographic *hashing* (for more details see `digest`)). These hash values change whenever *anything* in the full item wordings is changed, and allow a precise identification of different versions of an item. This function never exposes users to the hash values. Automatic, hashed IDs are generally recommended and easier to use, but some caveats apply (see `make.cards`).

For more information on this terminology and the rationale behind it, consider the best practices suggested by Maximilian Held on the data management page.

Suggested File Structure: For studies in which each Q-sort and item are kept in separate *.CSV files, the import functions `import.q.sorts`, `import.q.concourse`, `build.q.set`, `import.q.feedback` and the print function `make.cards` require a nested directory structure in the study folder. An example of such structure can be found in `../qmethod/extdata/importexample`. Although recommended for complex studies, this structure is not necessary for using the data analysis functions or for exploring and exporting results.

If the suggested file structure is followed, the subdirectories for (within-subjects) *conditions* and *languages* are optional, and need to be used only if there are more than one condition and language, respectively. In such case, the arguments `conditions` and `languages` for the above import functions must be specified accordingly.

For more information on the file structure and the rationale behind it, consider the best practices suggested by Maximilian Held on the data management page.

Bootstrapping: A set of functions are available to perform bootstrapping with Q data (see Zabala and Pascual, 2016, Bootstrapping Q Methodology to Improve the Understanding of Human Perspectives. PLoS ONE for details). The main bootstrap functions are as follows:

- `qmboots` performs a full bootstrap. It calls internally the functions `qbstep` (for each bootstrap step), and to either `qindtest` or `qpcrustes` in order to correct the **alignment problem**.
- `qmb.summary` summarises the object resulting from `qmboots` into two tables: (1) summary of **factor loadings** (standard factor loadings, bootstrapped factor loadings, and flagging frequency) and (2) summary of **statement scores** (bootstrapped and standard z-scores, bootstrapped and standard factor scores, standard error of bootstrapped z-scores, and differences between standard and bootstrapped values).
- `qmb.plot` plots an object resulting from `qmb.summary` in a figure designed for bootstrapped Q data (either the factor loadings or the statement z-scores).

Author(s)

Aiora Zabala
Main author and maintainer
https://aiorazabala.net/
<aiora.zabala@gmail.com>

Maximilian Held
Author of data management functions: `import.q.sorts`, `import.q.concourse`, `build.q.set`, `import.q.feedback` and `make.cards`
https://maxheld.de/

Frans Hermans
Author of centroid function: `centroid`
https://www.researchgate.net/profile/Frans-Hermans-3

References

- Zabala, A., 2014. qmethod: A Package to Explore Human Perspectives Using Q Methodology. *The R Journal*, 6(2):163-173.
  Available from: https://journal.r-project.org/archive/2014-2/zabala.pdf.
- Zabala A. and Pascual, U., 2016. Bootstrapping Q Methodology to Improve the Understanding of Human Perspectives. *PLoS ONE*.
  Available from: doi:10.1371/journal.pone.0148087.
- Watts, S., and and P. Stenner, 2012. *Doing Q Methodological Research: Theory, Method & Interpretation*, London: Sage Publications Ltd.
- Van Exel, J., and G. de Graaf, 2005. *Q Methodology: A Sneak Preview*
  https://qmethodblog.files.wordpress.com/2016/01/qmethodologyasneakpreviewreferenceupdate.pdfAvailable at this link.
- Brown, S. R., 1980. *Political subjectivity: Applications of Q methodology in political science*, New Haven, CT: Yale University Press.
  https://qmethodblog.files.wordpress.com/2016/01/brown-1980-politicalsubjectivity.pdfAvailable at this link.
- https://qmethod.org/
  The website of the *International Society for the Scientific Study of Subjectivity*.

- http://schmolck.org/qmethod/
  Peter Schmolck's Q Method Page, with further references, datasets and the PQMethod software.

Examples

```
data(lipset)
results <- qmethod(lipset[[1]], nfactors = 3, rotation = "varimax")
summary(results)
results
plot(results)
```

`build.q.set` *Q methodology: sample a Q set from a concourse*

Description

Subsets a concourse of items into a sample of selected items. Returns a dataframe with handles as row names, and languages (if applicable) as columns.

Usage

```
build.q.set(q.concourse, q.sample, q.distribution)
```

Arguments

`q.concourse`	A matrix with handles as row names, (optional) languages as columns, and full item wordings in cells as produced by `import.q.concourse`.
`q.sample`	A character vector of handles (such as `q-is-great`). The items identified by the handles will be sampled.
`q.distribution`	The chosen Q distribution as a vector of integers, such as `c(1,3,1)`.

Details

Q studies are carried out letting participants rank a *sample* of statements (items), collectively referred to as the *Q set*. These Q sets are drawn (by some sampling strategy) from a *concourse*, or universe of items. This function subsets the concourse generated by `import.q.concourse`, based on a vector of handles provided, and returns it as `q.set`.

The function implements a number of tests on the validity and consistency of inputs.

If you are not familiar with the terminology of item *handle*, *ID* and *wording* or the file structure expected for import functions, please read the respective sections in the documentation for qmethod-package first or consider the package website.

Value

Returns a matrix with handles as row names, languages (if applicable) as column names and full item wordings in cells.

Note

This function currently does *not* actually *draw* a sample, but merely builds the Q set from a *given* sample.

This function currently requires input in the argument `q.distribution`, but it only checks for the sum, so if you are working with a distribution-free study that still has a fixed number of items, you can just enter a vector of length 1 with your total sum of items.

Author(s)

Maximilian Held

See Also

`import.q.concourse`, `import.q.feedback`, `import.q.sorts`, `make.cards`

Examples

```
# Build a Q Set from a concourse and a sample
data(importexample)
q.set <- build.q.set(
  q.concourse = importexample$q.concourse,  # as created by import.q.concourse
  q.sample = c("life-with-q","q-uprising","r-dominance","small-village"),
  # add vector with items to be selected from concourse
  # q.sample is ideally read in from a separate *.CSV file
  q.distribution = c(1,2,1) # very simple distribution
  )
```

---

centroid *Q methodology: centroid extraction*

Description

Extracts factors/ components using the centroid approach as an alternative to Principal Components Analysis.

Usage

```
centroid(tmat, nfactors = 7, spc)
```

Arguments

`tmat`	a correlation matrix between Q-sorts.
`nfactors`	number of factors/ components to extract. Defaults to 7.
`spc`	the threshold to accept factor results, set to 0.00001 by default (in Brown 1980, this is set to 0.02).

Details

This functions implement the centroid method for extraction of factors, an alternative to Principal Components that can be used in Q methodology. The calculations are based in Brown (1980; below).

The function is called from within `qmethod` where the attribute `extraction` is set to `centroid`.

This function can be used independently where conducting each step of the analysis separately, preceded by a correlation between Q-sorts and followed by the rotation of factors/ components (see below), calculation of z-scores, etc.

Value

Returns a matrix with Q-sorts as rows, and rotated factors as columns.

Note

This is a function used within `qmethod`. Rarely to be used independently.

Author(s)

Frans Hermans

References

Brown, S. R., 1980 *Political subjectivity: Applications of Q methodology in political science*, New Haven, CT: Yale University Press, pages 208-224.

See further references on the methodology in `qmethod-package`.

Examples

```
#### Example
require('qmethod')
require ("psych")

# Load data
data("lipset")
lip <- lipset[[1]]

# Correlation matrix
corlip <-cor(lip)

# Centroid extraction
lipcent <- centroid(corlip)
lipcent


## To finalise the full analysis, continue with the following steps
# Rotation (in this example, varimax over 3 factors)
vmax <- varimax(lipcent[,1:3])

# Automatic pre-flagging of Q-sorts
```

```
flags <- qflag(unclass(vmax$loadings), nstat = 33)

# Calculate z-scores and general characeristics
results <- qzscores(lip, 3, loa=vmax$loadings, flagged=flags)
summary(results)

# Consensus and distinguishing statements
results$qdc <- qdc(lip, 3, zsc=results$zsc, sed=results$f_char$sd_dif)

plot(results)

## All of the above can be done with:
results2 <- qmethod(lip, 3, extraction="centroid")
```

`export.pqmethod` *Q methodology: export to PQMethod *.DAT and *.STA files*

Description

Exports Q data to *.DAT and *.STA files readable in PQMethod software.

Usage

```
export.pqmethod(dataset, study.name,
     study.description, col.range,
     filename='Q_data_forPQmethod',
     left.zeros, right.zeros, statements)
```

Arguments

`dataset`	a matrix or data frame qwith Q data: Q-sorts as columns and statements as rows. The names of the columns will be used as Q-sort IDs in the *.DAT file.
`study.name`	a string with a short name of the study. No space characters are allowed.
`study.description`	a string with a one-sentence description of the study).
`col.range`	a two-element numerical vector with the values at the two extremes of the Q distribution (e.g. `c(-4, 4)`).
`filename`	a filename. The extension *.DAT is added automatically).
`left.zeros`	number of zeros before the distribution, in the second line of *.DAT file.
`right.zeros`	number of zeros after the distribution, in the second line of *.DAT file).
`statements`	a matrix with statements, one in each row).

Details

Exports the raw data of a Q methodology study into the native format used in PQMethod. Returns a message with some basic information about the data.

Note that **no checks are made on the data**, such as whether there are duplicated or non-forced Q-sorts.

This function is not applicable to non-forced distributions.

Note

This function is experimental. Use with caution and verify that the output is as desired.

Author(s)

Aiora Zabala

References

Schmolck, Peter, 2014. *PQMethod Software*, Available at: http://schmolck.org/qmethod/

File descriptions in *PQMethod Manual*: http://schmolck.org/qmethod/pqmanual.htm#appdxa

Examples

```
# data(lipset)
# db <- lipset[[1]]
# export.pqmethod(dataset = db,
#                 study.name = 'mystudy',
#                 study.description = 'great study',
#                 col.range = c(-4, 4),
#                 filename = 'mystudy',
#                 statements=lipset[[2]])
```

`export.qm` *Q Methodology: export results to a plain text document*

Description

Exports an object of class `QmethodRes` to a plain text file (*.TXT). All the objects within the list resulting from `qmethod` are exported as they are. This is intended for interpretation rather than for further analysis.

Usage

```
export.qm(qmobject, file, style= c("R", "PQMethod"))
```

Arguments

qmobject	an object of Q methodology results, obtained from the function qmethod.
file	the file name. Note that in some operating systems, the file name will need an extension *.TXT so that other software opens it.
style	the structure and formatting of the results in the exported document. Defaults to "R" where the qmobject will be written as is. Option "PQMethod" provides an output with similar structure and elements as those provided by PQMethod software in the *.LIS files (see details of *.LIS files in the References below). Note that the latter creates a much longer document.

Author(s)

Aiora Zabala

References

Schmolck. *PQMethod Software (Version 2.35)*, 2014. http://schmolck.org/qmethod/

File descriptions in *PQMethod Manual*: http://schmolck.org/qmethod/pqmanual.htm#appdxa

---

import.easyhtmlq *Q methodology: import data from easy-HTMLQ*

Description

Imports data from *.JSON files created with easy-HTMLQ software for Q-sort administration.

Usage

```
import.easyhtmlq(filename, ...)
```

Arguments

filename	a file with extension *.JSON (see full description of the file below in References).
...	further arguments to be passed to read.csv2.

Details

Extracts the raw data of a Q methodology study from the native format saved in *easy-HTMLQ*. Returns a list with two objects.

The first object contains a data frame with items as rows and Q-sorts as columns, ready to be used in qmethod. It sets the Q-sort names to the values in the column 'uid' or else in 'sid'.

The second object contains the additional data collected. Columns npos, nneu and nneg have the number of items allocated to the groups of 'positive', 'neutral', and 'negative' respectively. Columns which name start with comment* and form* contain further information introduced by the respondent. Columns which name start with dur* contain the time that the respondent spent in each screen. Column datetime contains the data stamp when the Q-sort was submitted.

Author(s)

Aiora Zabala

References

Banasick, Shawn, 2021. *easy-htmlq*, Available at: https://github.com/shawnbanasick/easy-htmlq, based on Oschlies, Johannes and Killing, Marvin, 2015. *HTMLQ*, Available at: https://github.com/aproxima/htmlq

---

`import.htmlq` *Q methodology: import data from HTMLQ and FlashQ*

Description

Imports data from *.CSV files created with HTMLQ or FlashQ softwares for Q-sort administration.

Usage

```
import.htmlq(filename, ...)
```

Arguments

`filename`	a file with extension *.CSV, separated by ";" as done by default in HTMLQ (see full description of the file below in References).
`...`	further arguments to be passed to `read.csv2`.

Details

Extracts the raw data of a Q methodology study from the native format saved in both *FlashQ* and *HTMLQ*. Returns a list with two objects.

The first object contains a data frame with items as rows and Q-sorts as columns, ready to be used in `qmethod`. It sets the Q-sort names to the values in the column 'uid' or else in 'sid'.

The second object contains the additional data collected. Columns `npos`, `nneu` and `nneg` have the number of items allocated to the groups of 'positive', 'neutral', and 'negative' respectively. Columns which name start with `comment*` and `form*` contain further information introduced by the respondent. Columns which name start with `dur*` contain the time that the respondent spent in each screen. Column `datetime` contains the data stamp when the Q-sort was responded.

Author(s)

Aiora Zabala

References

Hackert, Christian and Braehler, Gernot, 2007. *FlashQ*, Used to be available at: http://www.hackert.biz/flashq, but offline as tested on Feb 2021.

Oschlies, Johannes and Killing, Marvin, 2015. *HTMLQ*, Available at: https://github.com/aproxima/htmlq

import.pqmethod *Q methodology: import PQMethod *.DAT files*

Description

Imports data from *.DAT files created in PQMethod software.

Usage

```
import.pqmethod(file, ...)
```

Arguments

file	a file with extension *.DAT (see full description of the file below in References).
...	further arguments to be passed to read.table and read.fwf.

Details

Extracts the raw data of a Q methodology study from the native format used in PQMethod. Returns a data frame with statements as rows and Q-sorts as columns.

If the following error occurs: "invalid multibyte string", a possible solution is to either set the right file-encoding in the argument fileEncoding or inspect the file for uncommon characters (see details in read.table).

Author(s)

Aiora Zabala

References

Schmolck, Peter, 2014. *PQMethod Software*, Available at: http://schmolck.org/qmethod/

File descriptions in *PQMethod Manual*: http://schmolck.org/qmethod/pqmanual.htm#appdxa

import.q.concourse *Q methodology: import concourse of Q items*

Description

Imports a full set of items (statements in a concourse) from a directory of *.TEX files (one file per item), including possible translations in separate folders.

Usage

```
import.q.concourse(q.concourse.dir, languages = NULL)
```

Arguments

`q.concourse.dir`
: A directory of *individual* item wordings in *.TEX files with handles as filenames (e.g. `happy-feeling.tex`). If `languages` are specified, the directory should contain one folder per language, with all full item wordings as individual *.TEX files in *each* language folder. Items should have the *same* file name across all languages (e.g. `happy-feeling.tex`). Directories end with a trailing slash, such as `study/q-sample/q-concourse/`.

`languages`
: A character vector of languages, same as folders within `q.concourse.dir`. If the concourse is monolingual, leave empty. Defaults to `NULL`.

Details

Q studies are conducted by asking participants (or a P set) to rank order a *sample* (or Q Set) of items, drawn from a universe (or concourse) of items, based on some sampling strategy. A concourse is, simply put, *the sum of all things people could say about a subject matter*.

It is helpful to keep the *entire* concourse readily available, so as to draw samples from it.

For some studies, it is necessary to have the complete items available in several languages.

This function simply imports all full item wordings and assigns a *handle* for the item, based on the filename (see qmethod-package). These filenames should be short and meaningful to the researcher.

Individual items as *.TEX files should include minimal markup, and no trailing whitespace or empty newlines. If you do not need any additional formatting, you can just save plain text files (*.TXT) with the extension *.TEX. There is no need to know LaTeX.

Returns error if items are not available in all translations.

Defaults to monolingual variant.

If you are not familiar with the terminology of Q item *handle*, *ID* and *wording* or the file structure expected for import functions, please read the respective sections in the documentation for qmethod-package first or consider the package website.

Value

Returns a character matrix with handles as row names, languages (if applicable) as columns and full item wording per language in cells.

Author(s)

Maximilian Held

See Also

`build.q.set`, `import.q.feedback`, `import.q.sorts`, `make.cards`

Examples

```
## Import a full q concourse from 'importexample' dataset
path.concourse <- paste(           # this part is only for the example!
  path.package("qmethod"),         # just to make sure, use absolute path
  # import example files are in root/extdata of package
  "/extdata/importexample/sample/concourse/",  # location of concourse
  sep = ""
  )
q.concourse <- import.q.concourse(  # import concourse
  q.concourse.dir = path.concourse, # insert your applicable path here
  languages = c("english","german") # choose your languages from path here
  )
```

`import.q.feedback` *Q methodology: imports feedback on Q items*

Description

Turns raw item feedback (in *.CSV files) into a verified array or matrix.

Usage

```
import.q.feedback(q.feedback.dir,q.sorts, q.set,manual.lookup = NULL)
```

Arguments

`q.feedback.dir` A relative path to a directory structure where:

- (optional) folders are `conditions` (such as `before` and `after`), if there is more than one condition. Conditions are inferred from the specified `q.sorts`. If there are no conditions, there should be no folders.
- filenames of *.CSV are participant names (might be given pseudonyms).
- *.CSV files within folders contain raw feedback, beginning with an arbitrary header line (ignored), and the following columns, starting from the left:
  1. An ID, either as an automatic hash or manually specified (see `qmethod-package`), as specified per the `manual.lookup` option of `make.cards`. Each ID only occurs once.
  2. The full feedback in plain text, enclosed in quotes.
  3. Optionally, a logical indicator whether current line should be ignored (in which case it should be set to `TRUE`). If there is no such column, all feedback will be imported.

`q.sorts` A matrix or array with handles as row names, participant as column names, (optional) conditions as 3rd dimension and cells as Q-sort ranks, as produced by `import.q.sorts`.

`q.set` A matrix with handles as row names, languages (if applicable) in columns, as produced by `build.q.set`.

`manual.lookup` A matrix with handles as row names, and IDs (such as `"sta121"`, as printed on the Q-cards by `make.cards`) in any of the columns. Defaults to `NULL` in which case items IDs are expected to be item wording hashes, as produced by `make.cards`.

Details

Participants in Q studies are often invited to provide open-ended feedback on items, giving researchers additional information on participants' viewpoints. This feedback is conveniently entered in a spreadsheet editor (2nd column), where each line of feedback corresponds to an item ID (1st column) An additional, optional (3rd) column indicates whether the current line should be ignored (`TRUE`), as may be the case for privacy reasons or when the feedback is merely a correction of a typographic error. If no such 3rd column is included, all feedback will be imported.

The automatic summary of full item wordings, technically known as *hashing*, proceeds internally by passing the full item wording to the `digest` function of the package **digest** (with arguments set to `algo = crc32, serialize = FALSE`.)

After an (arbitrary) header line, a *.CSV file may look like this:
'`sta001,"This q-item sounds like r-research to me!",FALSE`', indicating that it should *not* be ignored (`FALSE`).

If you are not familiar with the terminology of item *handle*, *ID* and *wording* or the file structure expected for import functions, please read the respective sections in the documentation for qmethod-package first or consider the package website.

Value

Returns a matrix or array (if there is more than one condition) with handles as row names, people as column names, (optional) conditions as 3rd dimension name and item feedback in cells. The return parallels the output from `import.q.sorts`, but with feedback as array cells, rather than Q-sort ranks.

Author(s)

Maximilian Held

See Also

`import.q.concourse`, `import.q.sorts`, `build.q.set`, `make.cards`, `qmethod`

Examples

```
data(importexample)
path.feedback <- paste(           # this part is only for the example!
  path.package("qmethod"),        # just to make sure, use absolute path
  # import example files are in root/extdata of package:
  "/extdata/importexample/feedback/",  # location of sorts
  sep = ""
  )
```

```
q.feedback <- import.q.feedback(   # now import the feedback
  q.feedback.dir = path.feedback,  # add your path here
  q.sorts = importexample$q.sorts,
  q.set = importexample$q.set,     # as produced by build.q.set
  manual.lookup = matrix(  # ideally empty for automatic hashing, or read in from *.CSV
    c("i01","i02","i03","i04"),
    ncol = 1,
    nrow = 4,
    dimnames = list(c("r-dominance","q-uprising","small-village","life-with-q"),"ID")
    )
  )
```

`import.q.sorts` *Q methodology: import Q-sorts from CSV*

Description

Turns raw Q-sorts (from *.CSV) into a Q-sorts array (when there are > 2 conditions) or matrix (with single condition).

Usage

```
import.q.sorts(q.sorts.dir,q.set, q.distribution,
               conditions = NULL, manual.lookup = NULL)
```

Arguments

`q.sorts.dir`	A relative path to a directory structure where: • (optional) folders are (within-subjects) `conditions` (such as *before* or *after*), if there is more than one condition as per the `conditions` argument. If there are no `conditions` (defaults to `NULL`), there should be no folders. • file names of *.CSV files are participant names (might be given pseudonyms). • *.CSV files contain *raw* Q-sorts, where each line contains item IDs, such as '`,,sta12,sta64,,`'. All lines below the maximum Q-sort height will be ignored and can have arbitrary input. For example, in a study with a highest column of 8 cards, everything below the 8th line in the file will be ignored. There is no need to include the values of the x-axis (say, `-4` to `+4`) in these files. If they are included, they should be the last row. • cells contains either all manual or all automatic item IDs (such as `"sta12"`), both as produced by `make.cards`.
`q.set`	A matrix with handles as row names, languages (if applicable) in columns, as read in by `build.q.set`.
`q.distribution`	The chosen Q distribution as a vector of integers, such as `c(1,3,1)`.
`conditions`	A character vector of (optional) study (within-subjects) conditions, such as `c("before","after")`, same as folders under `q.sorts.dir`. Defaults to `NULL` in which case there is only one condition, and *.CSV files are expected directly under `q.sorts.dir`.

`manual.lookup` A matrix with handles (such as `q-is-great`, same as in `build.q.set`, `import.q.concourse`) as row names, and arbitrary strings (item IDs, such as `"it212"`) in any of the columns as printed on the Q-cards by `make.cards`. Defaults to `NULL` in which case items are automatically identified by automatic hash IDs, as also produced by `make.cards`.

Details

This function imports Q-sorts from their raw format stored in *.CSV files, in the form in which they were sorted by participants (applicable to Q-sorts with forced distributions only).

Q-sorts in their raw form have columns as ranks (from, say, `-6` to `+6`) with cards (items) sorted in rows. The vertical dimension of Q-sorts is *meaningless*.

Q-sorts are conveniently entered as *.CSV (comma separated values) files in standard spreadsheet editors. This function ignores any rows in the files below the maximum height of columns expected from `q.distribution`.

It is recommended that Q-sort data are kept in their rawest form, with clear documentation of any processing applied to this data. This is also good practice for reproducible research.

Q-sorts are best entered not by typing up the full form of an item, but some unique string (ID) printed on the card. This function, and, analogously, `make.cards` and `import.q.feedback` offer a manual and automatic way to create these IDs, which are then expected as input (see qmethod-package for details).

The automatic summary of full item wordings, technically known as *hashing*, proceeds internally by passing the full item wording to the `digest` function of the package **digest** (with arguments set to `algo = crc32, serialize = FALSE`.)

Q-sorts are conveniently entered as *.CSV (comma separated values) files in standard spreadsheet editors.

This function includes a number of tests to verify the integrity of entered Q-sorts:

1. `manual.lookup` tables provided are tested for duplicate identifiers.
2. Function returns a warning if some participants do not have Q-sort files under all conditions (applies only if there are more than one `conditions`).
3. Function errors out if there are item IDs in a Q-sort not matched by any manually or automatically specified ID, respectively (see `qmethod-package` for details).
4. Function errors out if the distribution in a given Q-sort does not conform to the defined `q.distribution`.
5. Function errors out if there are items in the sample `q.set` that cannot be found in any given Q-sort.
6. Function errors out if there are items in a given Q-sort that cannot be found in the sample `q.set`.

If you are not familiar with the terminology of item *handle*, *ID* and *wording* or the file structure expected for import functions, please read the respective sections in the documentation for `qmethod-package` first or consider the package website.

Value

Returns a matrix (when there is a single condition) or array (with two or more conditions) with handles as row names, people as column names, conditions (if more than one) as 3rd dimension and Q-sort ranks in cells, as expected for analysis by qmethod.

Notice that qmethod expects a matrix (with two dimensions). If you have several conditions, and therefore an array of data, you must pass them to qmethod in individual 'slices' of conditions, using subsetting.

Note

This function currently works only with forced distributions.

When argument manual.lookup is set to NULL, IDs are computed by "summarising" the complete item wordings ("Q Method is used by a crazy, but charming community of ...") into a hexadecimal number ("ae128fs"), a process known as cryptographic hashing. These hash values change whenever anything in the full item wordings is changed, and allow a precise identification of different versions of an item. This function never exposes users to the hash values; the encrypting and decrypting are done under the hood by the respective functions. Automatic, hashed IDs are generally recommended and easier to use, but some caveats apply.

Hashed identification has not been widely tested in Q studies and should be used with great care and only for extra convenience. When using hash identification, researchers should be careful to record the precise item wordings at the time of hashing for the printed Q-cards, preferably with a version control system. Researchers should also record the complete Q-sorts of participants in an *unhashed* form, such as a picture of the completed sort in full wordings, in case problems with the hashing arise.

This function does *not* test whether Q-sorts were entered correctly into the *.CSV files. It is recommended to enter any given Q-sort more than once and have a spreadsheet editor compare several entry attempts for consistency. This function ignores any entries in *.CSV files below the highest row expected by the q.distribution.

Author(s)

Maximilian Held

See Also

import.q.concourse, import.q.feedback, build.q.set, make.cards, qmethod

Examples

```
## Import a Q sample from a directory of *.CSV files
data(importexample)
path.sorts <- paste(                # this part is only for the example!
  path.package("qmethod"),          # just to make sure, use absolute path
  # import example files are in root/extdata of package:
  "/extdata/importexample/qsorts/",  # location of sorts
  sep = ""
  )
q.sorts <- import.q.sorts(          # now import the sorts
```

```
q.sorts.dir = path.sorts,          # add your path here
q.set = importexample$q.set,       # as produced by build.q.set
q.distribution = c(1,2,1),         # very simple distribution
conditions = c("before","after"),  # enter your conditions here, same as in path
manual.lookup = matrix(  # ideally empty for automatic hashing,
                         # or read in from *.CSV file
  c("i01","i02","i03","i04"),
  ncol = 1,
  nrow = 4,
  dimnames = list(c("r-dominance","q-uprising","small-village",
                    "life-with-q"),"ID")
  )
)
```

`importexample` *Import Example*

Description

A minimum working example (MWE) to test the functions `import.q.concourse`, `build.q.set`, `import.q.sorts`, `import.q.feedback` and `make.cards`. The example is too small to run an actual Q analysis. To test out a real study with the same data structure, go to: `https://github.com/maxheld83/keyneson`.

Usage

```
importexample
```

Format

`importexample` is included as a directory in `qmethod` package root folder, including subdirectories as documented in the package documentation, and on the package website. Importexample is *also* partly included as a ready-made RData datafile in the folder `qmethod/data` so that (cumulative) function examples can run.

Source

None.

lipset — Lipset *(1963) Q methodology dataset*

Description

Dataset about *The Value Patterns of Democracy* based on Lipset (1963) to illustrate the **qmethod** package.

Usage

```
lipset
```

Format

A list with two objects. A data frame with 9 Q sorts sorting 33 statements and a data frame with the text corresponding to the statements.

Source

Brown, S. R., 1980. *Political subjectivity: Applications of Q methodology in political science*, New Haven, CT: Yale University Press.

Lipset, S. M., 1963. The value patterns of democracy: A case study in comparative analysis. *American Sociological Review*, 28, 515-531.

loa.and.flags — *Q methodology: show factor loadings next to flags*

Description

Prints a table with factor loadings and flagged Q-sorts are indicated with a star.

Usage

```
loa.and.flags(results, nload = FALSE)
```

Arguments

results	an object of Q method results.
nload	logical; print number of flagged Q-sorts.

Details

Simple function to explore the Q-sorts that are automatically pre-flagged, using the standard criteria implemented in function qflag

Author(s)

Aiora Zabala

Examples

```
data(lipset)
results <- qmethod(lipset[[1]], nfactors = 3, rotation = "varimax")
loa.and.flags(results)
```

`make.cards` *Q methodology: produce printable cards for Q study with ID and full item wording*

Description

Creates cards for administering a Q study. Full item wordings are printed on the front of business cards and item IDs on the back.

Usage

```
make.cards(
    q.set,
    study.language = NULL,
    paper.format = "AveryZweckformC32010.Rnw",
    output.pdf = TRUE,
    manual.lookup = NULL,
    wording.font.size = NULL,
    file.name = "QCards",
    babel.language = NULL
    )
```

Arguments

`q.set`	A matrix with handles as row names ("`q-is-great`", for example), languages (if applicable) in columns, as produced by `build.q.set`.
`study.language`	A character vector of length 1. Must be one of the languages from the column names in the specified `q.set` (which will be the same as the respective Q concourse object). Defaults to `NULL`, in which case the first column from `q.set` is selected.
`paper.format`	A character vector of length 1, choosing among available templates of business card sheets. Defaults to the only currently available `"AveryZweckformC32010.Rnw"`. Must include file extension of template.
`output.pdf`	Logical. If `TRUE`, function invokes `knit2pdf` to create a PDF in the workspace. If `FALSE`, function invokes `knit` to return only a *.TEX in the workspace, may be preferable if no LaTeX installation is available on the used computer. Defaults to `TRUE`.

`manual.lookup` A matrix with handles (same as in `build.q.set`, `import.q.concourse`) as row names, and arbitrary, unique identifying strings in any of the columns as also expected in `import.q.sorts` and `import.q.feedback`. Defaults to `NULL` in which case items are automatically identified by full item hashes, as also detected by `import.q.sorts` and `import.q.feedback`.

`wording.font.size` A character vector of length 1 to set the font size of the full item wording on the cards. Defaults to `NULL` in which case the default font size 12pt is used. Only standard LaTeX font sizes are allowed, from `\tiny` to `\Huge`.

`file.name` A character vector of length 1 to set the output file path relative to the working directory *without file extension*. Defaults to `QCards`.

`babel.language` A character vector of length 1 to set the babel language for appropriate hyphenation, special letters and other international support as provided by the babel LaTeX package. Only available babel options are permissible. Defaults to `NULL`, in which case babel is never called. Changing `babel.language` between function calls can occasionally leave inconsistent LaTeX temp files, which may trip up compilation. Please re-run the function once again or clean up temp files (in the working directory) in that case.

Details

Preparing cards with full items and IDs quickly becomes cumbersome if a study is done several times or if items change frequently. Participants require well-printed, well-designed cards for their sorting task, ideally on heavier paper. Cards should include shorthand, unique identifiers to simplify later data entry.

This function prepares a properly typeset *.PDF (or *.TEX source), where items are printed on readily-available business card templates, from which individual cards can be easily broken out.

The function prints the full item wording on the *right* column of any page, and the identifier (ID) on the *left* column. If templates are duplex printed with the same page on the front and back, and in proper orientation, the front of each card includes the full wording, and the back its unique identifier (ID).

Identifiers (ID) entered manually or automatically hashed from full wordings are also expected in the import functions `import.q.sorts` and `import.q.feedback`. The automatic summary of full item wordings, technically known as *hashing*, proceeds internally by passing the full item wording to the `digest` function of the package **digest** (with arguments set to

`algo = crc32, serialize = FALSE`.)

The function proceeds internally by preparing a dataframe with full item wordings and identifiers (ID), and then invokes a prepared `*.RNW` template included with this package, which in turn includes a **knitr** chunk, which in turn calls **xtable** to return a neatly layed-out multi-page table.

If you are not familiar with the terminology of item *handle*, *ID* and *wording* or the file structure expected for import functions, please read the respective sections in the documentation for qmethod-package first or consider the package website.

Value

Writes a *.PDF file or its source *.TEX file to the working directory ready for printout.

Note

Hashed identification has not been widely tested in Q studies and should be used with great care and only for extra convenience. When using hash identification, researchers should be careful to record the precise item wordings at the time of hashing for the printed Q-cards, preferably with a version control system. Researchers should also record the complete Q-sorts of participants in an *unhashed* form, such as a picture of the completed sort in full wordings, in case problems with the hashing arise.

When `output.pdf = TRUE`, the function will sometimes fail with the error message `"Running 'texi2dvi' on ... failed"`. This is not a bug with the function, but simply indicates that the path to pdflatex is not available in the current R environment. To fix this issue, compile the resulting *.TEX manually, use RStudio or try this fix.

This function does *not* automatically scale the font size to fit the given card size. Instead, users will have to proceed by trial and error, using a `wording.font.size` that works for their longest item. The default value should work for most Q items.

This function currently only works for Avery Zweckform C32010 templates, designed in `/cardtemplates/AveryZweckformC32010.Rnw`. If you would like support for other templates, check out / chip in here: https://github.com/aiorazabala/qmethod/issues/34.

Author(s)

Maximilian Held

See Also

`build.q.set`, `import.q.feedback`, `import.q.sorts`, `import.q.concourse`

Examples

```
## Make cards from importexample
data(importexample)
# use your own output file name or leave NULL for `file.name`
# tempfile() is used here to avoid leaving files behind example code runs
make.cards(importexample$q.set, output.pdf = FALSE, file.name = tempfile())
```

`make.distribution` *Q methodology: create Q normal distribution*

Description

Creates a distribution close to a standard normal distribution given a number of statements and a maximum Q sort rank.

Usage

```
make.distribution(
 nstat,
 max.bin = 5
)
```

Arguments

nstat	Number of desired statements in a Q sample for a given study. Must be a positive integer, vector of length 1.
max.bin	Maximum positive value to be entered by participants. Must be a positive integer, vector of length 1. Defaults to frequent value 5.

Details

Participants in Q studies are frequently asked to sort Q items under a quasi-normal distribution. This function generates such a Q distribution, given a number of statements `nstat` and a desired maximum positive value `max.bin` in the Q distribution.

The function always returns an *uneven* number of columns, ensuring that there is always a modal column at the zero mark.

Not every combination of `nstat` and `max.bin` can be neatly fit under a standard normal distribution, in which case the function returns a vector of unequal length to the specified `nstat`. The function will issue a warning in that case. Researchers can either accept the different `nstat`, or try again with a different `max.bin`.

Value

Returns a vector of positive integers (column heights), of the length `max.column * 2 + 1`. An object of this kind is expected in `import.q.sorts`, `build.q.set` and other convenience functions.

Author(s)

Maximilian Held

See Also

`build.q.set`, `import.q.sorts`

Examples

```
## Make Q distribution
make.distribution(nstat=76, max.bin=7)
```

---

plot.QmethodRes *Q Method: plot for statement z-scores*

Description

Takes an object of class `QmethodRes` resulting from `qmethod` and makes a dot-chart with the z-scores for statements and all factors.

Usage

```
## S3 method for class 'QmethodRes'
plot(x, xlab = 'z-scores', ylab = 'statements',
     pchlist = NULL, colours = NULL,
     fnames = NULL, legend = TRUE,
     dist = TRUE, pchlist.fill = NULL,
     leg.pos="bottomright", xlim= NULL,
     sort.items=T, factors = NULL,
     ...)
```

Arguments

x	results object returned by qmethod.
xlab	label for x axis. Defaults to 'z-scores'.
ylab	label for y axis. Defaults to 'statements'.
pchlist	array of pch symbols to be used in plotting the points for each factor. Defaults to a pre-defined set of symbols.
colours	array of colours to be used when plotting the points for each perspective. Defaults to a pre-defined set of colours based on the rainbow palette.
fnames	names for factors to be used in the legend. In results where factor names have not been changed (using, e.g. q.fnames) it defaults to 'Factor 1', 'Factor 2', etc.
legend	logical; if FALSE, no legend will be drawn.
dist	Logical. Should distinguishing statements be indicated in the plot dots? If TRUE, then the z-score values that are distinguishing for a given statement and factor are represented with a different (filled) symbol.
pchlist.fill	List of symbols to indicate distinguishing statements. By default, this is set to NULL, which provides a set of symbols that coincides with those in pchlist, but filled.
leg.pos	Position of the legend.
xlim	Limits for the x axis, given as a vector of two numbers. If this is not provided, the limits are calculated from the sample.
sort.items	Whether and how the items are sorted in the vertical axis. Defaults to TRUE, which sorts the items according to the standard deviation of their z-scores for all factors; items of most disagreement are placed at the top. To invert this order (items of most disagreement at the bottom), set this argument to "consensus.top" A value of FALSE will not sort the items, and these are displayed in the same order as in the raw data. A numerical vector can also be provided in order to sort the statements manually: the vector needs to have the same length as the number of items, and provides the order in which the items are to be ordered.
factors	The factors to plot. Defaults to NULL, which plots all the factors in the object x in the order given. To print a subset of these factors or to print them in a different order, provide a numeric vector here with the factors and the order desired, e.g. c(2,1).
...	other arguments for plot.

Note

The names of items to be plotted are taken from the matrix `x$zsc`.

To change these names, change the row names in that matrix first, e.g.: `rownames(x$zsc) <- vector.of.item.names`.

If the margin width is not enough to read the items, specify `par(mai=...)` first. See `par` for details.

Author(s)

Aiora Zabala

References

This specific dotchart visualisation of Q results implemented in `plot.QmethodRes` was first developed and introduced in this R package, in preparation for the study published in Zabala et al. (2017).

- Zabala, A., 2014. qmethod: A Package to Explore Human Perspectives Using Q Methodology. *The R Journal*, 6(2):163-173.
  Available from: https://journal.r-project.org/archive/2014-2/zabala.pdf.
- Zabala, A., Pascual, U. and Garcia-Barrios, L. 2017. Payments for Pioneers? Revisiting the Role of External Rewards for Sustainable Innovation under Heterogeneous Motivations. *Ecological Economics*, 135:234-245.
  Available from: https://www.sciencedirect.com/science/article/pii/S0921800916302683/.

See Also

`dotchart` and `points`.

Examples

```
data(lipset)
results <- qmethod(lipset[[1]], nfactors = 3, rotation = "varimax")
title <- "Q method z-scores, lipset dataset"
subtitle <- paste0("Three factors, PCA, varimax. Printed on ",
                   Sys.Date())
plot(results, main = title, sub = subtitle)

# Order the items in a different way
plot(results, main = title, sub = subtitle,
     sort.items = c(rev(1:nrow(results$zsc))))
```

print.QmethodRes *Q Method: print method for results*

Description

Takes an object QmethodRes resulting from qmethod and prints it in a synthetic way.

Usage

```
## S3 method for class 'QmethodRes'
print(x, length = 10, digits = 2, ...)
```

Arguments

x	an object of class QmethodRes.
length	maximum number of rows to print from the data frames within QmethodRes. Defaults to 10. Set to NULL to see the full results.
digits	minimum number of significant digits, see print.default.
...	further arguments passed to or from other methods.

Author(s)

Aiora Zabala

Examples

```
data(lipset)
results <- qmethod(lipset[[1]], nfactors = 3, rotation = "varimax")
print(results, length = 5, digits = 1)
```

q.fnames *Change factor names in the results of Q methodology analysis*

Description

This function replaces the automatic names created in an object of Q method results returned by qmethod.

Usage

```
q.fnames(results, fnames)
```

Arguments

`results`	an object of class `QmethodRes`.
`fnames`	a vector with the names of the factors. The number of names provided has to match the number of factors extracted in the object `results`. The names cannot begin with a number. A limit of 50 characters is set, to avoid excessively wide columns. Names should ideally contain no spaces or symbols that are used for other purposes in R (e.g. '-', '+', '/' ,). However '.' are fine.

Value

Returns the object `results` of class `QmethodRes`, with the new factor names.

Author(s)

Aiora Zabala

See Also

`qmethod`

Examples

```
data(lipset)
results <- qmethod(lipset[[1]], nfactors = 3, rotation = "varimax")
factor.names <- c("good", "bad", "ugly")
results.renamed <- q.fnames(results, fnames = factor.names)
results.renamed #shows all results
```

---

qbstep *Q Methodology: Single step for the bootstrap*

Description

Bootstraping of Q methodology using PCA.

Usage

```
qbstep(subdata, subtarget, indet,
       nfactors, nqsorts, nstat,
       qmts = qmts, qmts_log = qmts_log,
       rotation = "unknown",
       flagged = flagged, cor.method="pearson", ...)
```

Arguments

subdata	resampled dataset of Q-sorts.
subtarget	target matrix, adapted to match the rows of the resampled dataset.
indet	method to solve the double indeterminacy issue when bootstrapping Principal Components Analysis (PCA). "procrustes" for procrustes rotation from **MCMCpack**, "qindtest" for simple solution valid for at least up to 3 factors extracted (see references), "both" for a "qindtest" and a "procrustes" rotation, or "none" for no solution. The latter is not recommended because it introduces inflated variability. If "none" is selected, each replication is rotated using the method set in rotation.
nfactors	number of factors in the study.
nqsorts	number of Q-sorts in the study.
nstat	number of statements in the study.
qmts	data frame with two rows and at least one column. This is automatically created when this function is called from qmboots(see *Note* below).
qmts_log	data frame with two rows and at least one column. This is automatically created when this function is called from qmboots(see *Note* below).
rotation	rotation method, defaults to "none".
flagged	matrix or data frame of nqsorts rows and nfactors columns, with TRUE values for the Q-sorts that are flagged. Automatic flagging can be aplied using qflag. Manual flagging can be done by providing a logical matrix with nqsorts rows and nfactors columns to the argument flagged.
cor.method	character string indicating which correlation coefficient is to be computed, to be passed on to the function cor: "pearson" (default), "kendall", or "spearman".
...	other arguments to be passed on to qzscores or to principal.

Details

This function performs a single step within a bootstrap of Q methodology data. It takes one resample, performs the Q method analysis, checks for indeterminacy issues, and corrects them if necessary by calling the function qindtest or qpcrustes.

Value

step_res	summary of the analysis.

Note

This function is called within the function qmboots. Not intended to be used separately.

Author(s)

Aiora Zabala

References

Zabala, Pascual (2016) Bootstrapping Q Methodology to Improve the Understanding of Human Perspectives. PLoS ONE 11(2): e0148087.

See Also

qmethod and qmboots in this package.

---

qdc *Q methodology: distinguishing and consensus statements*

Description

Indicates the distinguishing and consensus statements. It does so by comparing the z-scores between each pair factors.

Usage

```
qdc(dataset, nfactors, zsc, sed)
```

Arguments

dataset	a matrix or a dataframe containing original data, with statements as rows, Q sorts as columns, and grid column values in each cell.
nfactors	number of factors extracted.
zsc	a matrix or a dataframe with the factor z-scores for statements resulting from qzscores.
sed	a matrix or a dataframe with the standard error of differences resulting from qfcharact.

Details

Finds the distinguishing and consensus statements, based on the absolute differences between factor z-scores being larger than the standard error of differences (SED, calculated in qfcharact) for a given pair of factors.

Returns a single data frame with the differences in z-scores between each pair of factors and the variable dist.and.cons, indicating whether each statement is distinguishing or consensus and for which factor(s) it is distinguishing. These are the possible categories in the dist.and.cons variable:

- Where all the comparisons between each pair of factors are significantly different at p-value < .05 the statement is labelled as "Distinguishes all".
- Where the comparisons of a given factor with all other factors are significant at p-value < .05, and comparisons between all other factors are not significant, the statement is labeled as "Distinguishes f*".
- Where none of the comparisons are significantly different, the statement is labeled as "Consensus".

- Statements that have category "" (empty) are not distinguishing for any of the factors in particularly. They distinguish one or more pairs of factors and the star indications may be inspected to understand their role.

Significant differences at p-values:

- p >= 0.05 <- "" (i.e. nothing)
- p < 0.05 <- "*"
- p < 0.01 <- "**"
- p < 0.001 <- "***"
- p < 0.000001 <- "6*"

Note

This is a function used within qmethod. Rarely to be used independently.

Author(s)

Aiora Zabala

References

Brown, S. R., 1980 *Political subjectivity: Applications of Q methodology in political science*, New Haven, CT: Yale University Press.

See further references on the methodology in qmethod-package.

Examples

```
data(lipset)
results <- qmethod(lipset[[1]], nfactors = 3, rotation = "varimax")
sed <- as.data.frame(results[[7]][[3]])
zsc <- results[[5]]
qdc(lipset[[1]], nfactors = 3, zsc = zsc, sed = sed)
```

---

qdc.zsc *Q methodology: distinguishing and consensus statements*

Description

Extracts the z-score of distinguishing statements, in order to plot.

Usage

```
qdc.zsc(results)
```

Arguments

results	an object of class QmethodRes.

Note

This is a function used within `plot.QmethodRes`. Rarely to be used independently.

Author(s)

Aiora Zabala

Examples

```
data(lipset)
results <- qmethod(lipset[[1]], nfactors = 3, rotation = "varimax")
qdc.zsc(results)
```

`qfcharact` *Q methodology: factor characteristics*

Description

Calculates the general factor characteristics: number of flagged Q-sorts, composite reliability, standard errors of factor scores, and comparisons between factors.

Usage

```
qfcharact(loa, flagged, zsc, nfactors, av_rel_coef = 0.8)
```

Arguments

`loa`	matrix or data frame of as many rows as Q-sorts (`nqsorts`) and `nfactors` columns, with values of factor loadings for Q-sorts, calculated using, e.g., `principal(...)$loadings`.
`flagged`	matrix or data frame of type *logical*, indicating which Q-sorts are flagged for each factor. Provided manually or automatically using `qflag`.
`zsc`	a data frame with the z-scores for statements, calculated using `qzscores`.
`nfactors`	number of factors extracted.
`av_rel_coef`	average reliability coefficient (the individual variability of a respondent), set by default as 0.8.

Value

Returns a list with three objects:

`characteristics`

data frame with the following values for each factor:

- `"av_rel_coef"`: average reliability coefficient.
- `"nload"`: number of loading Q-sorts.
- `"eigenvals"`: eigenvalues.
- `"expl_var"`: percentage of explained variance.

- `"reliability"`: composite reliability.
- `"se_fscores"`: standard error of factor scores (SE).

`cor_zsc` matrix of correlation coefficients between factors z-scores.

`sd_dif` matrix of standard errors of differences (SED).

Note

This is a function used within `qzscores`. Rarely to be used independently.

Author(s)

Aiora Zabala

References

Brown, S. R., 1980 *Political subjectivity: Applications of Q methodology in political science*, New Haven, CT: Yale University Press.

See further references on the methodology in `qmethod-package`.

---

qflag *Q methodology: automatic flagging of Q-sorts*

Description

Applies the two standard algorithms to pre-flag Q-sorts automatically, for posterior calculation of the statement scores.

Usage

```
qflag(loa, nstat)
```

Arguments

`loa` a Q-sort factor loading matrix obtained, for example from `unclass(principal(...)$loadings)`, or from `qmethod(...)$loa`.

`nstat` number of statements in the study.

Details

These are the two standard criteria for automatic flagging used in Q method analysis:

1. Q-sorts which factor loading is higher than the threshold for p-value < 0.05, and
2. Q-sorts which square loading is higher than the sum of square loadings of the same Q-sort in all other factors.

Returns a logical matrix with Q-sorts as rows, and factors as columns.

The function also runs two checks: Q-sorts flagged that have negative loadings and Q-sorts flagged in more than one factor. If any of these is true, the function returns a warning for the user to inspect the automatic pre-flagging (which should be done in all cases, but particularly in these ones). To conduct manual flagging, see guidelines here: http://aiorazabala.github.io/qmethod/Advanced-analysis

Note

This is a function used within qmethod. Rarely to be used independently.

Author(s)

Aiora Zabala

References

Brown, S. R., 1980 *Political subjectivity: Applications of Q methodology in political science*, New Haven, CT: Yale University Press.

Van Exel, J., de Graaf, G., Rietveld, P., 2011. "'I can do perfectly well without a car!'" *Transportation* 38, 383-407 (Page 388, footnote 8).

See further references on the methodology in qmethod-package.

Examples

```
data(lipset)
library(psych)
loa <- unclass(principal(lipset[[1]], nfactors = 3,
               rotate = "varimax")$loadings)
flagged <- qflag(loa = loa, nstat = nrow(lipset[[1]]))
summary(flagged)

# Remember to manually inspect the automatic pre-flagging:
results=list(loa=loa, flagged=flagged, brief=list(nfactors = ncol(loa)))
loa.and.flags(results)
```

---

qfsi *Q Methodology: Factor Stability index*

Description

Calculates a Factor Stability index and a Normalised Factor Stability index to bootstrapped Q method results (experimental).

Usage

```
qfsi(nfactors, nstat, qscores, zsc_bn, qm)
```

Arguments

nfactors	number of factors to extract.
nstat	number of statements in the study.
qscores	all possible factor score values in the Q grid distribution.
zsc_bn	bootstrapped factor scores.
qm	original Q method results from qmethod function.

Details

Applies the Factor Stability index to a bootstrapped Q method results. Returns a data frame with two variables and as many rows as factors extracted. The first variable is the raw Factor Stability index. The second variable is the Normalised Factor Stability index which ranges from 0 to 1.

Note

IMPORTANT: This function is experimental. Please contact the author for details.

Author(s)

Aiora Zabala

See Also

qmboots.

Examples

```
data(lipset)
boots <- qmboots(lipset[[1]], nfactors=3, nsteps=10,
           rotation="varimax", indet="qindtest",
           fsi=FALSE)
fsi <- qfsi(nfactors=3, nstat=33, qscores=boots[[6]],
           zsc_bn=boots[[1]][[1]], qm=boots[[5]])
fsi
```

---

qindtest *Q Methodology: PCA bootstrap indeterminacy tests*

Description

This is a simple test and implementation of the 'reordering-reflection' solution for the indeterminacy problem (alignment problem) when bootstrapping Principal Components Analysis (PCA) that causes factor order swaps and factor sign swaps.

Usage

```
qindtest(loa, target, nfactors)
```

Arguments

loa	data frame with factor loadings from the subsample analysis.
target	data frame with factor loadings from the full sample analysis, excluding qsorts that are not present in the bootstrap step.
nfactors	number of factors extracted.

Details

This function tests whether there is any or both of the indeterminacy issues in bootstrapped PCA factor loading values. For testing, it looks at correlation coefficients between the target factor loadings and the bootstrapped factor loadings for each factor.

First, if *factor swap* is detected (Is the absolute value of diagonal coefficients bigger than non-diagonal coefficients for the same factor?) and it is only between two factors, these are swaped. After, the test is again performed to ensure that there is no need for further swaps. If the test fails, then the original factor loadings are recovered and the failure is reported. If the need for factor swap is detected for 1, 3 or more factors, this is reported and left unresolved. This is because an algorithm to determine which factors should swap with which has not been implemented.

Second, *sign swap* is tested for (Are all diagonal coefficients positive?). If it is detected, then the sign of factor loadings is shifted. This is not tested again afterwards, for it is given for granted that swaping signs will solve the issue.

Value

qindtest	returns a list with three data frames: the factor loadings of the corrected bootstrap step, results from order swap and sign swap tests, and report of errors.

Note

this function is called within the function qmboots. Not intended to be used separately.

Author(s)

Aiora Zabala

References

Zabala, Pascual (2016) Bootstrapping Q Methodology to Improve the Understanding of Human Perspectives. PLoS ONE 11(2): e0148087.

See also:

Timmerman, M.E., Kiers, H. a L., Smilde, A.K., 2007. Estimating confidence intervals for principal component loadings: a comparison between the bootstrap and asymptotic results. The British journal of mathematical and statistical psychology 60, 295-314.

Zhang, G., Preacher, K.J., Luo, S., 2010. Bootstrap Confidence Intervals for Ordinary Least Squares Factor Loadings and Correlations in Exploratory Factor Analysis. Multivariate Behavioral Research 45, 104-134.

Examples

```
data(lipset)
nf <- 3

# 1. Create target matrix
qm <- qmethod(lipset[[1]], nfactors = nf, rotation = "varimax")

# 2. Resample
qselim <- sample(1:3, 2, replace = FALSE) ##q sorts to eliminate
subdata <- lipset[[1]][ , -qselim]

# 3. Calculate factor loadings with the resample
library(psych)
loa <- as.data.frame(unclass(principal(subdata,
           nfactors = nf, rotate = "varimax")$loadings))

# 4. Reorder target matrix
target <- as.matrix(as.data.frame(qm[3]))
colnames(target) <- paste0("target_f", 1:nf)
subtarget <- target[c(-qselim),]

# 5. Apply test and solution for indeterminacy issue
qindt <- qindtest(loa, subtarget, nf)
qindt
```

`qmb.plot` *Q Methodology: Plot of bootstrap results*

Description

Plots the summary of bootstrap results, either z-scores or factor loadings.

Usage

```
qmb.plot(qmbsum, type = c("zsc", "loa"), nfactors,
         cex = 0.7, cex.leg = 0.8, errbar.col = "black",
         lwd = 1, lty = 1, vertdist = 0.2, limits = NULL,
         r.names = NA, sort = c("none", "difference", "sd"),
         sbset = NULL, leg.pos = "topleft",
         bty = "n", plot.std = TRUE, pch= NULL,
         col=NULL, grid.col="gray", ...)
```

Arguments

`qmbsum`	an object with the summary of bootstrap results, as produced by `qmb.summary`.
`type`	the subject to plot, either z-zcores of statements or factor loadings of Q-sorts.
`nfactors`	number of factors extracted.

cex	a numerical value giving the amount by which plotting text and symbols should be magnified relative to the default (see par).)
cex.leg	a numerical value giving the amount by which the legend should be magnified relative to cex.)
errbar.col	colour used for the error bars. Defaults to "black".
lwd	line width (see par).
lty	line type (see par).
vertdist	distance between the values for each factor.
limits	axis limits for the numerical values. If set to NULL, the limits are automatically set as c(-1, 1) when type = "loa", and as the minimum and maximum values of z-scores (including the error bars) when type = "zsc"
r.names	names of the items to be printed in the axis ticks(either Q-sorts when type = "loa", or statements when type = "zsc"). When the value is NULL, it defaults to rownames.
sort	ordering of the items in the axis. If set to "none", items are ordered by the default order in the dataset. If set to "difference", items are ordered according to the variability in the values across factors. If set to "sd", items are ordered according to the sum of the errors obtained in the bootstrap.
sbset	How many items are to be printed? When the value is NULL, it plots all the items.
leg.pos	Position of the legend.
bty	Legend box (see legend).
plot.std	logical value. When set to TRUE (default), it prints the points for values obtained with the standard analysis (non bootstrapped).
pch	plotting symbols. Defaults to NULL, in which case the symbols are selected automatically. If provided, the vector needs to contain at least as many elements as number of factors. In addition, if argument plot.std == TRUE (default) the vector needs to contain at least double as many elements as vectors, in order to extract (a) the plotting symbols for bootstrapped values (the first elements) and (b) the plotting symbols for standard values (the next elements).
col	colours for the points. At least as many elements as number of factors have to be provided.
grid.col	colour of the grid.
...	additional arguments to be passed to the functions dotchart, mtext, segments, points, abline or legend.

Author(s)

Aiora Zabala

References

Zabala, Pascual (2016) Bootstrapping Q Methodology to Improve the Understanding of Human Perspectives. PLoS ONE 11(2): e0148087.

See Also

qmethod, qmboots, qmb.summary

Examples

```
data(lipset)
boots <- qmboots(lipset[[1]], nfactors = 3, nsteps = 50,
                load = "auto", rotation = "varimax",
                indet = "qindet", fsi = TRUE)

boots.summary <- qmb.summary(boots)

qmb.plot(boots.summary, 3, type = "loa", sort="difference")
```

---

qmb.summary *Q Methodology: Summary of bootstrap results*

Description

Summarises bootstrap results for Q-sorts and statements into two tables.

Usage

```
qmb.summary(qmboots)
```

Arguments

qmboots	an object of bootstrap results, as produced by qmboots.

Value

Returns a list with two data frames:

qsorts	data frame with Q-sort as rows, and the following columns: the factor loadings from the standard analysis (*.std), the bootstrap (*.loa), the bootstrap SE (*.SE), the frequency of flagging (*.freq*) and the estimate of bias (*.bias).
statements	data frame with statements as rows, and the following columns: the z-scores from the standard analysis (*.std), from the bootstrap (*.bts), bootstrap SE (*.SE), estimate of bias of z-scores (*.bias), factor scores from the standard analysis (fsc_f*), from the bootstrap (fsc.bts.*), estimate of bias of factor scores, distinguishing and consensus statements from the standard results (see qdc) and from the bootstrap values.

Author(s)

Aiora Zabala

References

Zabala, Pascual (2016) Bootstrapping Q Methodology to Improve the Understanding of Human Perspectives. PLoS ONE 11(2): e0148087.

See Also

qmethod, qmboots

Examples

```
data(lipset)
boots <- qmboots(lipset[[1]], nfactors = 3, nsteps = 50,
                 load = "auto", rotation = "varimax",
                 indet = "qindet", fsi = TRUE)

boots.summary <- qmb.summary(boots)

# First rows of the summary for Q-sorts:
head(boots.summary$qsorts)

# First rows of the summary for statements:
head(boots.summary$statements)
```

---

qmboots *Q Methodology: Bootstrap*

Description

Implementation of the bootstrap to Q methodology using Principal Components Analysis (PCA).

Usage

```
qmboots(dataset, nfactors, nsteps, load = "auto",
        rotation = "varimax", indet = "qindtest", fsi = TRUE,
        forced = T, distribution = NULL,
        cor.method="pearson", ...)
```

Arguments

dataset	a matrix or dataframe containing original data, with statements as rows, Q sorts as columns, and Q board column values in each cell.
nfactors	number of factors to extract using PCA.
load	a matrix of factor loadings to be used as target. If "auto", the target matrix is generated using the rotation indicated ("varimax" by default).
nsteps	number of steps (repetitions) for the bootstraping.

rotation	rotation method, set to "varimax" by default. Other possible rotations from **psych** principal function "none", "varimax", "quatimax", "promax", "oblimin", "simplimax", and "cluster" are possible.
indet	method to solve the double indeterminacy issue in PCA bootstrapping. "procrustes" for procrustes rotation, "qindtest" for simple solution valid for up to 3 factors extracted, "both" for a qindtest and a procrustes rotation, or "none" for no rotation. The latter is not recommended for it introduces inflated variability. If "none" is selected, each replication is rotated using varimax.
fsi	logical; Shall the Factor Stability index be calculated? (experimental index).
forced	logical; Is the ranking of the items forced to match the distributions? Set to TRUE if all respondents ranked the items strictly following the distribution scores, in which case the values of the distribution are calculated automatically. Set to FALSE if respondents had the possibility to rank the items without following the distribution, and the values of the distribution have to be provided as an array in the argument distribution.
distribution	logical; when forced = FALSE, the distribution has to be provided as a vector of numbers, such as c(-2, -1, -1, 0, 1, 1, 2, 2).
cor.method	character string indicating which correlation coefficient is to be computed, to be passed on to the function cor: "pearson" (default), "kendall", or "spearman".
...	Other arguments passed on to qmethod.

Value

zscore-stats	summary of the analysis. List of one object, plus as many objects as factors extracted: the bootstrapped factor scores, and the z-score statistics of the bootrstrap. The z-score statistics of interest are mean (the bootstrap estimate of the z-score), and sd (the bootstrap estimate of the SE).
full.bts.res	full bootstrap results. List with as many objects as factors extracted, each object containing three data frames: flagged, zsc and loa. These data frames have as many columns as bootstrap steps, and contain the results of the analysis of each iteration. See description of these three data frames in qmethod.
indet.tests	indeterminacy tests.
resamples	index of the Q-sorts selected for each step.
orig.res	original results. See details of all the objects in qmethod.
q.array	array of values in the distribution grid.
loa.stats	statistics of factor loadings. List with as many objects as factors extracted, each object containing one data frame with the factor loading statistics of the bootrstrap. The factor loading statistics of interest are mean (the bootstrap estimate of the factor loading), and sd (the bootstrap estimate of the SE). This table includes flag_freq, which indicates the frequency with which the given Q-sort was flagged for the given factor.
q.array	array of values in the distribution grid.
fsi	factor stability index (optional; experimental).

Author(s)

Aiora Zabala

References

Zabala, Pascual (2016) Bootstrapping Q Methodology to Improve the Understanding of Human Perspectives. PLoS ONE 11(2): e0148087.

See Also

qmethod

Examples

```
data(lipset)
boots <- qmboots(lipset[[1]], nfactors = 3, nsteps = 10, load = "auto",
          rotation = "varimax", indet = "qindtest",
          fsi = TRUE)
boots
boxplot(t(boots[[2]][[1]][[2]]), horizontal = TRUE,
main = "Statement z-score boxplot for the first factor", las = 1)

#See the table summaries:
qms <- qmb.summary(boots)
round(qms$statements, digits=2) # statements
round(qms$qsorts, digits=2)     # Q-sorts

# A more synthetic visualisation:
# z-scores:
qmb.plot(qms, nfactors=3, type="zsc", sort="difference")
# factor loadings:
qmb.plot(qms, nfactors=3, type="loa", sort="difference")
```

---

qmethod *Q methodology analysis*

Description

This function performs a full Q methodology analysis. Both principal components analysis or centroid factor extraction can be used. The main results are factor characteristics, statement z-scores and factor scores, and distinguishing and consensus statements.

Usage

```
qmethod(dataset, nfactors, extraction = "PCA", rotation = "varimax",
        forced = TRUE, distribution = NULL, cor.method = "pearson",
        silent = FALSE, spc = 10^-5, ...)
```

Arguments

dataset	a matrix or a data frame containing original data, with statements as rows, Q-sorts as columns, and the column scores in the distribution in each cell. The matrix or data frame should not contain character strings. The results keep row names and column names if set in the dataset (see 'Details').
nfactors	number of factors to extract.
extraction	extraction method, either Principal Components Analysis or centroid factor extraction. It defaults to "PCA".
rotation	rotation method, defaults to "varimax". For "centroid" extraction, "none" and "varimax" are implemented. For "PCA" other possible rotations allowed in principal function can be used: "none", "varimax", "quartimax", "promax", "oblimin", "simplimax", and "cluster". Note that only 'varimax' and manual rotation are standard in Q methodology.
forced	logical; Is the ranking of the items forced to match the distributions? Set to TRUE if all respondents ranked the items strictly following the distribution scores, in which case the values of the distribution are calculated automatically. Set to FALSE if respondents were able to rank the items without following the distribution, and the values of the distribution have to be provided as an array in the argument distribution. See more details below in 'Notes'.
distribution	logical; when forced = FALSE, the distribution has to be provided as a vector of numbers, such as c(-2, -1, -1, 0, 1, 1, 2, 2). See more details below in 'Notes' number 2.
cor.method	character string indicating which correlation coefficient is to be computed, to be passed on to the function cor: "pearson" (default), "kendall", or "spearman".
silent	logical; when = TRUE, a summary message is printed.
spc	If centroid extraction is selected, this is the threshold to accept factor results, set to 0.00001 by default (in Brown 1980, this is set to 0.02; see centroid).
...	other parameters to pass to functions such as principal

Details

This function wraps together all the steps required for a complete analysis: extracting component loadings (principal or centroid); flagging Q-sorts (qflag); calculating weights, z-scores, and rounded scores (qzscores), calculating general characteristics (qfcharact), and finding distinguishing and consensus statements (qdc).

The default qmethod performs automatic flagging and uses varimax rotation. Varimax rotation can be replaced by "none" or other methods for rotation allowed in principal from **psych** package.

If the input data contains row names and variable names, these will be kept throughout the analysis. Input data is validated, and it will give an error if there are non numerical values or if either the number of statements and Q-sorts introduced do not match the input data. It also returns error if the argument forced is set to TRUE but Q-sorts contain differing distributions.

Value

Returns a list of class QmethodRes, with eight objects:

`brief`	a list with the basic values of the analysis: date (`"date"`), number of statements (`"nstat"`), number of Q-sorts (`"nqsort"`), whether the distribution was 'forced' (`"distro"`), number of factors extracted (`"nfactors"`), method for factor extraction (`"extraction"`), method for rotation (`"rotation"`), method for correlation (`"cor.method"`), package version (`"pkg.version"`), and a summary of this information for display purposes (`"info"`).
`dataset`	original data.
`loa`	factor loadings for Q-sorts.
`flagged`	logical dataframe of flagged Q-sorts.
`zsc`	statements z-scores.
`zsc_n`	statements factor scores, matched to the ordered array of values in the first row of the dataset.
`f_char`	factor characteristics (see `qfcharact`): • `"characteristics"`: data frame with the following values for each factor: average reliability coefficient, number of loading Q-sorts, eigenvalues, percentage of explained variance, composite reliability, standard error of factor scores. • `"cor_zsc"`: matrix of correlation coefficients between factors z-scores. • `"sd_dif"`: matrix of standard errors of differences.
`qdc`	distinguishing and consensus statements (see `qdc`).

Note about non-forced distribution studies

The forced/ non-forced distribution (argument `forced`) refers to whether respondents were able to sort the items freely or they had to fit them in the distribution (i.e. the pyramid). If the `qmethod` function returns the following error: `"Q method input: The argument 'forced' is set as 'TRUE', but ..."` and you are unsure of how to solve it, continue reading.

First, ensure that the data are correctly introduced. For example, typos in the numbers entered result from forced distribution Q-sorts appearing as non-forced.

Second, if you data are indeed non-forced, set the argument `"forced = FALSE"` and specify the argument `"distribution = ..."`. For the argument `"distribution"`, specify a numerical vector with as many elements as there are cells in your original distribution (i.e. as many items in the Q-set), and with the values of the columns. Repeat the values of each column as many times as there are cells in that column. For example, for the distribution shown in Figure 1 in this paper at *The R Journal*, the argument `distribution` should be:

```
c(-4, -4,
  -3, -3, -3,
  -2, -2, -2, -2,
  -1, -1, -1, -1, -1,
   0,  0,  0,  0,  0,
   1,  1,  1,  1,  1,
   2,  2,  2,  2,
   3,  3,  3,
   4,  4 )
```

Or alternatively (a different way of getting the same vector):

```
c(rep(-4, 2), rep(-3, 3), rep(-2, 4), rep(-1, 5), rep( 0, 5),
  rep( 1, 5), rep( 2, 4), rep( 3, 3), rep( 4, 2))
```

If you don't want to specify a given distribution, you can specify `distribution = c(1:nrow(dataset))` and then ignore the factor scores in the output of results.

IMPORTANT: The arguments `forced` and `distribution` are relevant only for the calculation of factor (normalised) scores. All other values in the results (e.g. z-scores) are unaffected by these specifications. **If in doubt in a study with non-forced distribution, best to interpret the z-scores instead of the factor scores**.

Author(s)

Aiora Zabala

References

Zabala, A., 2014. qmethod: a package to analyse human perspectives using Q methodology. *The R Journal*, 6(2):163-173. Available from: https://journal.r-project.org/archive/2014-2/zabala.pdf (Open access).

Brown, S. R., 1980 *Political subjectivity: Applications of Q methodology in political science*, New Haven, CT: Yale University Press.

See further references on the methodology in `qmethod-package`.

See Also

`qzscores` and `centroid` in this package, and `principal` in package **psych**.

Examples

```
data(lipset)
results <- qmethod(lipset[[1]], nfactors = 3, rotation = "varimax")
summary(results)
results #shows all results

# Remember to manually inspect the automatic pre-flagging:
loa.and.flags(results)
```

`qpcrustes` *Q Methodology: Procrustes rotation of loadings*

Description

This is a wrap of `procrustes` rotation from **MCMCpack** for bootstrapping Q methodology in the function `qmboots`.

Usage

```
qpcrustes(loa, target, nfactors)
```

Arguments

loa	factor loadings from the analysis of a resample.
target	factor loadings from the analysis of a subsample.
nfactors	fumber of factors

Details

Returns the factor loadings for the subsample after applying Procrustes rotation to correct the indeterminacy issue. Use procrustes from **MCMCpack**. Used within the function qmboots, not intended for independent use.

Note

this function is called within the function qmboots. Not intended to be used separately. The function calls procrustes from **MCMCpack**, a package that requires the package graph. As from April 2016 the package has been moved to Bioconductor, and therefore it needs to be installed manually. If you get errors of missing packages when using this function or qmboots, install graph manually: source("https://bioconductor.org/biocLite.R") biocLite("graph")

Author(s)

Aiora Zabala

References

Zabala, Pascual (2016) Bootstrapping Q Methodology to Improve the Understanding of Human Perspectives. PLoS ONE 11(2): e0148087.

See Also

Function procrustes from **GPArotation** package.

Examples

```
# This example requires installing 'MCMCpack':
data(lipset)
qm <- qmethod(lipset[[1]], nfactors=3, rotation="varimax")
qselim <- sample(1:3, 2, replace=FALSE) ##q sorts to eliminate
subdata <- lipset[[1]][ , -qselim]
library(psych)
loa <- as.data.frame(unclass(principal(subdata,
         nfactors=3, rotate="varimax")$loadings))
target <- as.matrix(as.data.frame(qm[3]))
colnames(target) <- paste("target_f", 1:3, sep="")
subtarget <- target[c(-qselim),]
qindt <- qpcrustes(loa, subtarget, 3)
```

qindt

---

qzscores *Q methodology: z-scores from loadings*

Description

Calculates factor characteristics, z-scores, and factor scores, provided a matrix of loadings and a matrix of (manually or automatically) flagged Q-sorts.

Usage

```
qzscores(dataset, nfactors, loa, flagged, forced = TRUE,
         distribution = NULL)
```

Arguments

dataset	a matrix or a data frame containing raw data, with statements as rows, Q-sorts as columns, and the column scores in the distribution in each cell.
nfactors	number of factors to extract.
loa	matrix or data frame of nqsorts rows and nfactors columns, with values of factor loadings for Q-sorts, calculated using, e.g., principal(...)$loadings or centroid.
flagged	matrix or data frame of nqsorts rows and nfactors columns, with TRUE values for the Q-sorts that are flagged. Automatic flagging can be aplied using qflag. Manual flagging can be done by providing a logical matrix with nqsorts rows and nfactors columns to the argument flagged.
forced	logical; Is the distribution of items forced? Set to TRUE if all respondents ranked the items following strictly the distribution scores, and the values of the distribution are calculated automatically. Set to FALSE if respondents were able to rank the items without following the distribution, and the values of the distribution have to be provided as an array in the argument distribution.
distribution	logical; when forced = FALSE, the distribution has to be provided as a vector of numbers, such as c(-2, -1, -1, 0, 1, 1, 2, 2).

Details

In order to implement manual flagging, use a manually created data frame (or matrix) for flagged. See an example of code to perform manual flagging or to manipulate the loadings in the website.

The loadings from principal(...)$loadings or centroid can be explored to decide upon flagging. The loa data frame should have Q-sorts as rows, and factors as columns, where TRUE are the flagged Q-sorts.

Value

Returns a list of class `QmethodRes`, with seven objects:

`brief`	a list with the basic values of the analysis: date (`"date"`), number of statements (`"nstat"`), number of Q-sorts (`"nqsort"`), whether the distribution was 'forced' (`"distro"`), number of factors extracted (`"nfactors"`), type of extraction (`"extraction"`), type of rotation (`"rotation"`), method for correlation (`"cor.method"`), and a summary of this information for display purposes (`"info"`).
`dataset`	original data.
`loa`	factor loadings for Q-sorts.
`flagged`	logical dataframe of flagged Q-sorts.
`zsc`	statements z-scores.
`zsc_n`	statements rounded scores, rounded to the values in the first row of the original dataset.
`f_char`	factor characteristics obtained from `qfcharact`.

Note

This is a function used within `qmethod`. Rarely to be used independently.

Author(s)

Aiora Zabala

References

Brown, S. R., 1980 *Political subjectivity: Applications of Q methodology in political science*, New Haven, CT: Yale University Press.

See further references on the methodology in `qmethod-package`.

Examples

```
data(lipset)
library(psych)
loa <- unclass(principal(lipset[[1]],
               nfactors = 3, rotate = "varimax")$loadings)
flagged <- qflag(nstat = 33, loa = loa)
qmzsc <- qzscores(lipset[[1]], nfactors = 3, flagged = flagged, loa = loa)
qmzsc # Show results
```

`runInterface` *Q methodology: Graphical User Interface (GUI)*

Description

Launch an interactive interface to run Q methodology analysis using the basic features. The interface is also [available online](https://azabala.shinyapps.io/qmethod-gui/).

Usage

```
runInterface()
```

Details

This GUI allows the user to conduct a full Q methodology analysis, choosing:

- either PCA or centroid extraction method
- varimax or no rotation method (for PCA and centroid) and other uncommon rotation methods (for PCA)
- selecting from 2 to 7 factors/components.

The GUI conducts analysis with forced distribution and automatic flagging. See Note.

The GUI shows the full results from the analysis, and also:

- Plot of z-scores
- Automatically flagged Q-sorts
- Information to explore how many factors to extract (including a screeplot)
- Plot of z-scores

Note

This GUI has limited functionality in comparison to that through the command-line. For full functionality (such as specifying non-forced analysis, manual flagging, and much more), use the command-line directly in the R console. See, for example, a tutorial for manual manipulation of Q-sort loadings and/or manual flagging.

To run this same analysis directly in R, see the code generated in the GUI in *Run the analysis directly in R*.

Examples

```
## Only run this example in interactive R sessions
if (interactive()) {
  runInterface()
}
```

summary.QmethodRes *Q methodology: summary for class 'QmethodRes'*

Description

Shows a summary of the results of Q methodology from the qmethod function: factor scores and factor characteristics.

Usage

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

Arguments

object	an object of class QmethodRes created after qmethod function.
...	any other argument for the summary function.

Value

Returns the summary of the analysis:

- Statements factor scores normalized to the values in the first row of the original dataset, and
- Factor characteristics: Average reliability coefficient, Number of loading Q-sorts, Eigenvalues, Percentage of explained variance, Composite reliability, Standard error of factor scores, Correlation coefficients between factors z-scores, Standard errors of differences

Author(s)

Aiora Zabala

References

Brown, S. R., 1980 *Political subjectivity: Applications of Q methodology in political science*, New Haven, CT: Yale University Press.

See Also

qmethod in this package

Examples

```
data(lipset)
results <- qmethod(lipset[[1]], nfactors = 3, rotation = "varimax")
summary(results)
```

Index