Package 'qmethod'

May 5, 2015

Version 1.3.1

Date 2015-04-24

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 http://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.
Additional functions are available to import and export data, to print and plot, to import raw data from individual *.CSV files, and to make printable cards.
The package uses principal components and it allows manual or automatic flagging, a number of mathematical methods for rotation, and a number of correlation coefficients for the initial correlation matrix.
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 beta graphical user interface for online and offline use.

License GPL (>= 2)

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

LazyData true

Repository CRAN

URL https://github.com/aiorazabala/qmethod,
https://github.com/aiorazabala/qmethod/wiki

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

NeedsCompilation no

Author Aiora Zabala [aut, cre] (Main author),
Maximilian Held [aut] (Author of additional data management functions)

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

Date/Publication 2015-05-05 23:19:37

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 using principal components analysis, varimax rotation (replaceable by other rotations allowed in `principal`), and automatic flagging (manual flagging is optional).

The following steps of the analysis correspond to separate functions: factor loadings for Q-sorts, 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:

- Import data from PQMethod software (`import.pqmethod`), and from both HTMLQ and FlashQ (`import.htmlq`).

- 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.
- 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`)
- A function to rename the factors in the results, with short, meaningful names (`q.fnames`).

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.

Author(s)

Aiora Zabala
Main author and maintainer
http://www.landecon.cam.ac.uk/directory/aiora-zabala
<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
http://www.maxheld.de/

References

- Zabala, A., 2014. qmethod: A Package to Explore Human Perspectives Using Q Methodology. *The R Journal*, 6(2):163-173.
  Available from: http://journal.r-project.org/archive/2014-2/zabala.pdf.
- 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*
  Available from: http://qmethod.org/articles/vanExel.pdf.
- Brown, S. R., 1980. *Political subjectivity: Applications of Q methodology in political science*, New Haven, CT: Yale University Press.
  Available from: http://qmethod.org/papers/Brown-1980-PoliticalSubjectivity.pdf.
- http://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
```

`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 wiki.

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. Comment and/or chip in if you like want an actual sampling feature: https://github.com/aiorazabala/qmethod/issues/5.

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
  )
```

---

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.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 (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*, Available at: http://www.hackert.biz/flashq

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 wiki.

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 wiki

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 wiki

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. If you would like it to work with free distributions, check out and/or chip in at https://github.com/aiorazabala/qmethod/issues/33.

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 wiki. 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.

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 filename *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 wiki

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. You can also report other suggestions here.

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)
make.cards(importexample$q.set, output.pdf = FALSE)
```

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

Description

Takes an object `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, ...)
```

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.
`...`	other arguments for `plot`.

Author(s)

Aiora Zabala

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)
```

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")
results2 <- q.fnames(results, fnames = factor.names)
results #shows all results
```

---

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 with the factor z-scores for statements resulting from `qzscores`.
`sed`	a matrix 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.

Differences that are significant at a p-value < .05 are indicated with `'*'`, and differences significant at a p-value < .01 are indicated with `'**'`.

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.

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)
```

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, floa, av_rel_coef = 0.8)
```

Arguments

`loa`	a matrix or a data frame containing raw data, with statements as rows, Q-sorts as columns, and the scores of the columns in the distribution in each cell.
`flagged`	a 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.
`floa`	a data frame with the factor loadings, calculated using PCA or Centroid factor analysis.
`av_rel_coef`	average reliability coefficient (the individual variability of a respondent), estimated 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 `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`.

`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.

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

---

qmethod *Q methodology analysis*

Description

This function performs a full Q method analysis using principal components analysis (see Notes). The main results are factor characteristics, statement z-scores and factor scores, and distinguising and consensus statements.

Usage

```
qmethod(dataset, nfactors, rotation = "varimax", forced = TRUE,
        distribution = NULL, cor.method = "pearson", ...)
```

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.
rotation	rotation method, defaults to "varimax". Other possible rotations allowed in principal function can be used: "none", "varimax", "quartimax", "promax", "oblimin", "simplimax", and "cluster".
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.

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 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); 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 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 either if the number of statements and Q-sorts introduced does 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"), type or rotation ("rotation"), method for correlation in the PCA ("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 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

In Q methodology there are commonly two techniques to extract the factors: principal components analysis (PCA) and centroid factor analysis.

This package uses PCA because this function is already implemented in R. If you find a function in R that suitably implements centroid factor analysis as used in Q method (See Brown, 1980), please report to the author of this package: <aiora.zabala@gmail.com>.

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.

See Also

principal in package **psych**

Examples

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

---

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.
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 for flagged. See an example of code to perform manual flagging or to manipulate the loadings in https://github.com/aiorazabala/qmethod/wiki/Advanced-analysis.

The loadings from principal(...)$loadings 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 or rotation ("rotation"), method for correlation in the PCA ("cor.method"), and a summary of this information for display purposes ("info").
rawdata	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 <- as.data.frame(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
```

---

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