

# Package ‘conquestr’

October 12, 2022

**Type** Package

**Title** An R Package to Extend 'ACER ConQuest'

**Version** 0.9.96

**Description** Extends 'ACER ConQuest' through a family of functions designed to improve graphical outputs and help with advanced analysis (e.g., differential item functioning). Allows R users to call 'ACER ConQuest' from within R and read 'ACER ConQuest' System Files (generated by the command 'put'). Requires 'ACER ConQuest' version 5.19.5 or later. A demonstration version can be downloaded from <https://shop.acer.org/acer-conquest-5.html>.

**License** GPL-3

**URL** <https://www.acer.org/au/conquest>, <https://conquestmanual.acer.org>,  
<https://shop.acer.org/acer-conquest-5.html>

**Imports** dplyr, ggplot2 (>= 3.3.0), ggrepel, kableExtra, magrittr, methods, Rcpp, rlang, stats, stringr, tidyr, tidyselect

**Suggests** knitr, rmarkdown, testthat (>= 3.0.0)

**LinkingTo** Rcpp

**VignetteBuilder** knitr

**Config/testthat/edition** 3

**Encoding** UTF-8

**RoxygenNote** 7.1.2

**SystemRequirements** ACER ConQuest (>=5.19.5)

**Collate** 'RcppExports.R' 'ReadConQuestLibrary.R'  
'ReadConQuestRout\_createDF.R' 'ReadConQuestState.R'  
'ReadConQuestState\_createDF.R' 'conquestr.R' 'conquestrFunc.R'  
'generateHelpers.R' 'infoHelpers.R' 'itanalHelpers.R'  
'plotGeneral.R' 'plotRout.R' 'pvHelpers.R' 'residHelpers.R'  
'thrstThrsh.R' 'cleaningHelpers.R'

**NeedsCompilation** yes

**Author** Dan Cloney [aut, cre] (<https://orcid.org/0000-0002-2130-237X>),  
Ray Adams [aut]

**Maintainer** Dan Cloney <dan.cloney@acer.org>

**Repository** CRAN

**Date/Publication** 2022-04-01 10:50:02 UTC

## R topics documented:

checkItemRespValid . . . . .	3
checkVars . . . . .	4
ConQuestCall . . . . .	4
ConQuestRout . . . . .	5
ConQuestSys . . . . .	5
createConQuestProject . . . . .	6
fisherTrnsfrm . . . . .	7
fmtCqItanal . . . . .	7
genResponses . . . . .	8
getCqChain . . . . .	9
getCqData . . . . .	10
getCqDataDf . . . . .	10
getCqFit . . . . .	11
getCqHist . . . . .	12
getCqItanal . . . . .	12
getCqParams . . . . .	13
getCqParams2 . . . . .	13
getCqTerms . . . . .	14
getCqVars . . . . .	15
infoFunc . . . . .	15
informationWrightMap . . . . .	16
informationWrightMap2 . . . . .	17
infoWI . . . . .	18
itemInfoAtTheta . . . . .	19
itemInfoOverTheta . . . . .	19
makeItemList . . . . .	20
plotCqHist . . . . .	21
plotDif . . . . .	22
plotItemMap . . . . .	23
plotMCC . . . . .	24
plotRout . . . . .	25
pvMeanVar . . . . .	26
q3ExpCorrect . . . . .	26
ReadSys . . . . .	27
recodeResps . . . . .	27
replaceInDataFrame . . . . .	28
replaceInVector . . . . .	28
searchConQuestSys . . . . .	29
steigerStat . . . . .	29
summariseCqChain . . . . .	30
sysToBMatrixDf . . . . .	30

<i>checkItemRespValid</i>	3
sysToItemDifDf . . . . .	31
testInfoAtTheta . . . . .	32
testInfoOverTheta . . . . .	32
thrstThrsh . . . . .	33
timesTwo . . . . .	34
transformPvs . . . . .	34
<b>Index</b>	<b>35</b>

---

checkItemRespValid	<i>checkItemRespValid</i>
--------------------	---------------------------

---

### Description

Check that the item responses in raw data are: (1) valid, (2) each valid response mapped to an item appears at least once, and (3) each valid reponse mapped to an item has sufficently many responses (defaults to a minimum of 10 observations for each response category)

### Usage

```
checkItemRespValid(data, caseID, validMap, varLabel, validLabel)
```

### Arguments

data	Raw data, a data frame.
caseID	A string indicating the name of the case id variable in the data.
validMap	A data frame which contains a mapping of valid responses to item lables. This data frame shoudl be in long format, with each valis response * item combination representing a row.
varLabel	A string indicating the name of the variable in validMap that identidies the valis items names/lables.
validLabel	A string indicating the name of the variable in validMap that contains the valid codes/responses for each item. This shoudl include missing values (e.g., "99")

### Value

A list of lists: one list per item in validMap\$varLabel. Within each list, there can be up to three dfs: (1) the case ids and invalid responses for the item, (2) the valid codes not observed in the data set, and (3) the valid codes observed fewer than 10 times in the data. NOTE: a wannring is thrown if the validMap\$varLabel is not found in the data.

---

checkVars	<i>checkVars</i>
-----------	------------------

---

**Description**

Check raw data: are all required variables present and ensure there are no extraneous variables.

**Usage**

```
checkVars(data, varNames, except = NULL)
```

**Arguments**

data	Raw data, a data frame.
varNames	Vector of valid variable names.
except	A vector of variable names to be excluded from the check.

**Value**

A list.

---

ConQuestCall	<i>ConQuestCall</i>
--------------	---------------------

---

**Description**

Call an instance of 'ACER ConQuest' at the command line and run a control file (syntax).

**Usage**

```
ConQuestCall(cqc, cqExe, stdout = "")
```

**Arguments**

cqc	The location of the control file (syntax) to be run.
cqExe	The path to the 'ACER ConQuest' executable. Note, if this argument is missing, conquestr will find a local installation of ACER ConQuest by first searching the default installation locations (Program Files on Windows and Applications on Mac) then searching other local directories (Appdata and the HOME path).
stdout	On Mac only, can be toggled to NULL (or a connection) to suppress output to R console.

**Value**

prints 'ACER ConQuest' output to stdout.

**Examples**

```
## Not run:
ConQuestCall(cqExe = file.path("/Applications", "ConQuest BETA", "ConQuest"))

## End(Not run)
```

---

ConQuestRout	<i>ConQuestRout</i>
--------------	---------------------

---

**Description**

Read an "ACER ConQuest" rout file created by a 'plot' command in 'ACER ConQuest'.

**Usage**

```
ConQuestRout(myRout)
```

**Arguments**

myRout	The location of an 'ACER ConQuest' rout file created by 'ACER ConQuest' > 5.1.4.
--------	--

**Value**

A list containing the data objects created by 'ACER ConQuest' plot command.

**Examples**

```
myPlot<- ConQuestRout()
## Not run:
# if you run the above example you will have the points from a plot ICC command.
str(myPlot)

## End(Not run)
```

---

ConQuestSys	<i>ConQuestSys</i>
-------------	--------------------

---

**Description**

Read an "ACER ConQuest" system file created by a 'put' command in 'ACER ConQuest'. The system file must not be compressed. Use the option 'compressed=no' in the put command within 'ACER ConQuest'.

**Usage**

```
ConQuestSys(myCqs)
```

**Arguments**

myCqs            The location of an uncompressed 'ACER ConQuest' system file created by 'ACER ConQuest' > 4.30.2.

**Value**

A list containing the data objects created by 'ACER ConQuest'.

**Examples**

```
mySysData<- ConQuestSys()
myEx1SysData<- ConQuestSys(myCqs = system.file("extdata", "mysysfile.cqs", package = "conquestr"))
## Not run:
# if you run the above example this will return your original 'ACER ConQuest' syntax.
cat(unlist(myEx1SysData$gCommandHistory))

## End(Not run)
```

---

createConQuestProject    *createConQuestProject*

---

**Description**

creates a standard folder structure to work with 'ACER ConQuest' Projects.

**Usage**

```
createConQuestProject(prefix = getwd(), ...)
```

**Arguments**

prefix            a valid file path where to create project folders.  
...               optional params, including "setDebug"

**Value**

Boolean TRUE.

**Examples**

```
## Not run:
createConQuestProject()

## End(Not run)
```

---

fisherTrnsfrm	<i>fisherTrnsfrm</i>
---------------	----------------------

---

**Description**

Helper function to apply Fisher's transformation to a correlation matrix.

**Usage**

```
fisherTrnsfrm(myCorMat)
```

**Arguments**

myCorMat      A correlation matrix.

**Value**

A correlation matrix with Fisher's transform applied to values  $-1 < x < 1$ .

---

fmtCqItanal	<i>fmtCqItanal</i>
-------------	--------------------

---

**Description**

helper function to produce nicely formatted summary tables from a ConQuest Itanal.

**Usage**

```
fmtCqItanal(
  cqItanal,
  itemNumber = "all",
  ptBisFlag = 0,
  textColHighlight = "red",
  valueDecPlace = 2
)
```

**Arguments**

cqItanal      An ACER ConQuest itanal list object returned by function 'getCqItanal'.

itemNumber    a vector of generalised item numbers to format.

ptBisFlag      Something.

textColHighlight      Something.

valueDecPlace      Something.

**Value**

A list

**Examples**

```
myEx1Sys <- ConQuestSys()
myEx1Sys_itanal <- getCqItanal(myEx1Sys)
myItanalSummary <- fmtCqItanal(myEx1Sys_itanal)
print(myItanalSummary[[1]])
```

---

genResponses

*genResponses*

---

**Description**

Generates responses to *i* items for *n* cases given some item parameters, (true) person abilities, and other inputs.

**Usage**

```
genResponses(
  abilities,
  itemParams,
  BMatrix = 1,
  discrim = 1,
  mcarP = 0,
  perturbP = NULL
)
```

**Arguments**

<code>abilities</code>	A matrix of person abilities. One column per dimension. Rows are cases.
<code>itemParams</code>	A list of item params of the structure used in ‘simplef’ (a matrix of <i>k</i> categories by three (category score, delta dot, tau)). See <code>conquestr::makeItemList</code> for a helper to generate this list.
<code>BMatrix</code>	A simplified B-matrix mapping dimensions (columns) to items (rows). Or the integer "1" if items are dichotomous and ability is uni-dimensional.
<code>discrim</code>	A vector of item discrimination paramters, or an integer where all items have constant discrimination.
<code>mcarP</code>	A double indicating the proportion of missing data under the MCAR assumption.
<code>perturbP</code>	A list, where each element of the list contains a data frame referring to an item. Each data frame is either a 1 * 4 data frame describing the general perturbation to apply to the probabilities for that item (currently developed) or a pair of vectors mapping theta to probabilities (not developed). In the first case - the data frame is in this order: - "item": item number (int), - "type": the type of perturbation to



apply (string, "flat", or "steep"), - "pivot": probability pivot point around which the perturbation is applied (double in  $0 < x < 1$ ), - "factor": magnitude of the perturbation (double - when in  $0 < x < 100$  probs remain positively correlated with theta (0 = no perturbation, 100 = maximum perturbation) when in  $100 < x < \text{Inf}$  probs are negatively correlated with theta).

**Value**

A matrix.

**Examples**

```
## Not run:
myResponses<- genResponses(rnorm(1000, 0, 1), runif(10, -2, 3)) # TODO - update this

## End(Not run)
```

---

getCqChain

*getCqChain*

---

**Description**

creates a data frame representation of the estimation chain from an MCMC model. The burn is discarded and only the unskipped iterations in MCMC chain are retained.

**Usage**

```
getCqChain(myCqs)
```

**Arguments**

myCqs            A system file.

**Value**

A data frame.

**Examples**

```
## Not run:
getCqChain(ConQuestSys())

## End(Not run)
```

---

 getCqData

*getCqData*


---

### Description

Get data objects from an R object of class ConQuestSys. This function returns person IDs, response data, case estimates, regression and weight data. Each data type is stored as a data frame, and each data frame is a named element of a list. 1. PID, 2. Responses, 3. Estimates, 4. Regression.

### Usage

```
getCqData(mySys)
```

### Arguments

`mySys` An R object of class ConQuestSys, returned by the function `conquestr::ConQuestSys`

### Value

A List of data frames.

### See Also

`conquestr::ConQuestSys()`

### Examples

```
mySys <- ConQuestSys()
myData <- getCqData(mySys)
```

---

 getCqDataDf

*getCqDataDf*


---

### Description

Takes a list object returned by `'conquestr::getCqData'` and coerces it to a wide data frame. This can sometimes cause issues in complex data, for example where there are multiple response vectors for each case (for example a many-facets model). This is because it is assumed that the data can be reduced to a matrix of `_gNCases x m variables_` (where `_m_` is the number of id, item, estimate and regression variables in the analysis). For more complex data, the user should use the outputs of `'conquestr::getCqData'` to manually merge together a data frame.

### Usage

```
getCqDataDf(cqData)
```

**Arguments**

`cqData` An R object of class list, returned by the function `conquestr::getCqData`

**Value**

A data frame containing R data frames based on the list objects in the ConQuest system file that has been read in.

**See Also**

`conquestr::ConQuestSys()`

`conquestr::getCqData`

**Examples**

```
mySys <- ConQuestSys()
myData <- getCqData(mySys)
myDataDf <- getCqDataDf(myData)
```

---

*getCqFit**getCqFit*

---

**Description**

creates a data frame representation of the fit of parameters in the item response model

**Usage**

```
getCqFit(myCqs)
```

**Arguments**

`myCqs` A system file.

**Value**

A data frame.

**Examples**

```
## Not run:
getCqFit(ConQuestSys())

## End(Not run)
```

---

getCqHist	<i>getCqHist</i>
-----------	------------------

---

**Description**

creates a data frame representation of the iteration history for all parameters.

**Usage**

```
getCqHist(myCqs)
```

**Arguments**

myCqs            A system file.

**Value**

A data frame.

**Examples**

```
## Not run:
getCqHist(ConQuestSys())

## End(Not run)
```

---

getCqItanal	<i>getCqItanal</i>
-------------	--------------------

---

**Description**

helper function to return list of lists, each list relates to one generalised item from an ACER ConQuest 'itanal' output. Each list contains: (1) item-total and item-rest correlations ....

**Usage**

```
getCqItanal(sysFile, matrixPrefix = "", isDebug = FALSE)
```

**Arguments**

sysFile            An ACER ConQuest system file.  
matrixPrefix      to define which itanal to process  
isDebug            report debug output

**Value**

A list.

**Examples**

```
myItanal <- getCqItanal()
print(myItanal[[1]])
```

---

getCqParams

*getCqParams*


---

**Description**

creates a data frame representation of the parameters of the model, including both estimated and constrained parameters. Parameters are either freely estimated ('ParamType' == 0) or constrained ('ParamType' == 1). Parameters are indexed (0 offset) by the column 'ParamNumber'. There is a separate index for free and constrained parameters.

**Usage**

```
getCqParams(myCqs)
```

**Arguments**

myCqs            A system file.

**Value**

A data frame.

**Examples**

```
## Not run:
getCqParams(ConQuestSys())

## End(Not run)
```

---

getCqParams2

*getCqParams2*


---

**Description**

returns a list of data frames, one for each term in the model. The data frames include the estimate, location constraint, or anchored value for each parameter. The standard error of estimated parameters is also included if it has been estimated. The fit (un/weighted MNSQ, sometimes called outfit and infit) of all parameters is also included if it has been estimated.

**Usage**

```
getCqParams2(myCqs)
```

**Arguments**

myCqs            A system file.

**Value**

A list of data frames of length 'gTerms'.

**Examples**

```
## Not run:  
getCqParams2(ConQuestSys())  
  
## End(Not run)
```

---

getCqTerms

*getCqTerms*

---

**Description**

creates a data frame representation of the terms of the model statement, including interactions.

**Usage**

```
getCqTerms(myCqs)
```

**Arguments**

myCqs            A system file.

**Value**

A data frame.

**Examples**

```
## Not run:  
getCqTerms(ConQuestSys())  
  
## End(Not run)
```

---

`getCqVars`*getCqVars*

---

**Description**

creates a data frame representation of the variables in the model statement. Note that steps are not variables.

**Usage**

```
getCqVars(myCqs)
```

**Arguments**

`myCqs`            A system file.

**Value**

A data frame.

**Examples**

```
## Not run:  
getCqVars(ConQuestSys())  
  
## End(Not run)
```

---

`infoFunc`*infoFunc*

---

**Description**

Calculate test information function, given a vector of item difficulties and a vector of abilities. Returns a data frame that can be easily plotted.

**Usage**

```
infoFunc(  
  myItems,  
  myAbilities,  
  type = "empirical",  
  minTheta = NA,  
  maxTheta = NA,  
  stepTheta = NA  
)
```

**Arguments**

myItems	A vector of item deltas.
myAbilities	A vector of person abilities.
type	A character String. Should the test information be calculated empirically ("empirical" - default) or analytically using moments of distribution ("approx").
minTheta	The smallest value of ability PDF to plot.
maxTheta	The largest value of ability PDF to plot.
stepTheta	The increment to iterate over the ability PDF. Defaults to 0.01.

**Value**

A data frame.

**Examples**

```
myInfo<- infoFunc(runif(10, -2, 3), rnorm(1000, 0, 1))
```

---

```
informationWrightMap  informationWrightMap
```

---

**Description**

Plots test information function, relative to ability density, and item locations.

**Usage**

```
informationWrightMap(
  myItems,
  myAbilities,
  type = "empirical",
  minTheta = NA,
  maxTheta = NA,
  stepTheta = NA,
  scaleInfo = 1
)
```

**Arguments**

myItems	A vector of item deltas.
myAbilities	A vector of person abilities.
type	A character String. Should the test information be calculated empirically ("empirical" - default) or analytically using moments of distribution ("approx").
minTheta	The smallest value of ability PDF to plot.
maxTheta	The largest value of ability PDF to plot.



stepTheta	The increment to iterate over the ability PDF. Defaults to 0.01.
scaleInfo	A scaling factor to apply to the plot of test information. Because ability distribution is a PDF with area one, and a test information function has area L, this can make the plot more interpretable. Defaults to 1.

**Value**

A ggplot2 object.

**Examples**

```
myInfoPlot<- informationWrightMap(runif(10, -2, 3), rnorm(1000, 0, 1), minTheta=-10, maxTheta=10)
```

---

```
informationWrightMap2 informationWrightMap2
```

---

**Description**

Plots test information function, relative to ability density, and item locations (delta dots).

**Usage**

```
informationWrightMap2(
  myItems,
  myAbilities,
  minTheta = -6,
  maxTheta = 6,
  stepTheta = 0.1,
  scaleInfo = 1
)
```

**Arguments**

myItems	A list of sets of item parameters.
myAbilities	A vector of person abilities.
minTheta	The smallest value of ability PDF to plot.
maxTheta	The largest value of ability PDF to plot.
stepTheta	The increment to iterate over the ability PDF. Defaults to 0.01.
scaleInfo	A scaling factor to apply to the plot of test information. Because ability distribution is a PDF with area one, and a test information function has area L, this can make the plot more interpretable. Defaults to 1.

**Value**

A ggplot2 object.

**Examples**

```

myDetaDot <- data.frame(
  id = 1:10,
  itemid = paste0("item_", letters[1:10]),
  delta = runif(n = 10, min = -3, max = 2)
)
myTau<- data.frame(
  id = c(rep(1:2, each = 1), rep(3, 2)),
  itemid = paste0("item_", c(rep(letters[1:2], each = 1), rep(letters[3], 2))),
  step = c(1, 1, 1, 2),
  tau = c(-1.2, -0.3, -0.2, -0.1)
)
myItemParams<- makeItemList(myDetaDot, myTau)
myAbil <- rnorm(1000, 0, 1)
myInfoPlot <- informationWrightMap2(myItems= myItemParams, myAbilities = myAbil)

```

---

 infoWI

*infoWI*


---

**Description**

Calculates an index representing the product of a test information function and an ability distribution.

**Usage**

```
infoWI(myItems, myAbilities, type = "empirical")
```

**Arguments**

<code>myItems</code>	A vector of item deltas.
<code>myAbilities</code>	A vector of person abilities.
<code>type</code>	A character String. Should the test information be calculated empirically ("empirical" - default) or analytically using moments of distribution ("approx").

**Value**

A double.

**Examples**

```
infoWIOut<- infoWI(runif(10, -2, 3), rnorm(1000, 0, 1))
```

---

itemInfoAtTheta	<i>itemInfoAtTheta</i>
-----------------	------------------------

---

**Description**

Calculates item information at a value of theta given a set of item parameters.

**Usage**

```
itemInfoAtTheta(myItem, theta)
```

**Arguments**

myItem	A matrix of item parameters of the structure used in ‘simplef‘
theta	a number.

**Examples**

```
anItem<- matrix(c(0,0,0,1,1,0), nrow = 2, byrow = TRUE)
itemInfoAtTheta(anItem, 0)
```

---

itemInfoOverTheta	<i>itemInfoOverTheta</i>
-------------------	--------------------------

---

**Description**

Calculates item information over a range of theta given a set of item parameters. Returns a data frame with item information at a discrete set of values of theta - useful for plotting item information functions.

**Usage**

```
itemInfoOverTheta(myItem, minTheta = -6, maxTheta = 6, stepTheta = 0.1)
```

**Arguments**

myItem	A matrix of item parameters of the structure used in ‘simplef‘
minTheta	The smallest value of ability PDF to plot.
maxTheta	The largest value of ability PDF to plot.
stepTheta	The increment to iterate over the ability PDF. Defaults to 0.01.

**Examples**

```
anItem<- matrix(c(0,0,0,1,1,0), nrow = 2, byrow = TRUE)
itemInfoOverTheta(anItem)
```

---

<code>makeItemList</code>	<i>makeItemList</i>
---------------------------	---------------------

---

### Description

creates a list of matrices. Each matrix represent item parameters of the structure used in ‘simplef’ (a matrix of k categories by three (category score, delta dot, tau)).

### Usage

```
makeItemList(deltaDot, tau, scores = NULL)
```

### Arguments

<code>deltaDot</code>	a data frame or matrix of delta dots. If a data frame, column labels should be: "id", "itemid", "delta". "itemid" should be populated with an item label or be missing for all values. If a matrix, column order should be: "id", a unique item ID for each row; "itemid", item labels for each item (or NA); "delta", a delta dot.
<code>tau</code>	a data frame or matrix of taus for polytomous items. Only polytomous items should be in this file. If an item ID in ‘deltaDot’ is not in ‘tau’ it is assumed that the item is dichotomous. Where a polytomous item has k categories, there should be k-2 rows for that item in ‘tau’. For example, a 3-category item has categories k_1, k_2, and k_3. There will be one value in ‘tau’ for this item. The tau parameters represent the deviation from the delta dot to give the item parameters for adjacent category boundaries (e.g., k_1 and k_2, and k_2 and k_3). The value in ‘tau’ represents the the first category boundary (e.g., k_1 and k_2). The second category boundary is constrained to be the negative sum of the other values of tau for that item. If a data frame, column labels should be "id", "itemid", "step", "tau". If a matrix, the column order should be: "id", a unique item ID for each item matched with values in ‘deltaDot’; "itemid", item labels for each item (or NA); "step", an indicator of which step/item category this threshold represents (minimum value should be 1 and maximum value should be k-1); "tau" the value for the tau parameter associated with this step.
<code>scores</code>	a data frame of item response category scores. Not currently used - items are assumed to be scored as per the values in ‘tau\$step’.

### Value

a list.

### Examples

```
nItems<- 10
myItemsDeltaDot<- data.frame(
  id= seq(nItems),
  itemId= NA,
  delta = runif(nItems, -4, 1) # nItems items in range -4,1
```

```

)
myItemsTau<- data.frame(
  id = NA,
  itemId = NA,
  step = NA,
  tau = NA # this is only needed because the function is...in progress
)
myAbilities<- rnorm(100, 1, 2) # 100 person, offset from items - poor targeting (?)
myItemsList<- conquestr::makeItemList(deltaDot = myItemsDeltaDot, tau = myItemsTau)

```

---

plotCqHist

*plotCqHist*


---

### Description

generates a plot from a history object. Use 'getCqHist' to create a history object from an 'ACER ConQuest' system file.

### Usage

```
plotCqHist(myHist, centre = TRUE, params = c("all"), legend = FALSE)
```

### Arguments

myHist	an R object created by the 'getCqHist' function.
centre	a Boolean representing whether the iteration history should be mean centred (within parameter). This is helpful for plots that include all parameters to ensure the Y axis is sensible. Consider a plot with raw values of the Likelihood _and_ item parameters on it.
params	A vector of which params to plot. Must be one or more of "all", "Likelihood", "Beta", "Variance", "Xsi", "Tau".
legend	Should a legend be plotted?.

### Value

A ggplot2 object.

### Examples

```

## Not run:
myHistPlot<- plotCqHist(getCqHist(ConQuestSys()))

## End(Not run)

```

---

 plotDif

*plotDif*


---

### Description

Creates a plot (ggplot2 object) of item parameter estimates common to two system files (e.g., a DIF analysis).

### Usage

```
plotDif(mySysToItemDifDf, myScale = "centred", mySuffixes)
```

### Arguments

<code>mySysToItemDifDf</code>	An R object of class data frame returned from <code>conquestr::sysToItemDifDf</code>
<code>myScale</code>	A string specifying if the item parameter estimates displayed should be "centred" (default), "scaled" (z scores), or "none" (raw).
<code>mySuffixes</code>	a vector of strings specifying the names for the two groups being analysed, e.g., if the two system files are an analysis of boys and girls, the vector may be <code>c("_male", "_female")</code> .

### Value

A ggplot2 object.

### See Also

`conquestr::sysToItemDifDf()`

### Examples

```
mySys1<- ConQuestSys()
mySys2<- ConQuestSys()
mySysList<- list(mySys1, mySys2)
myDifDf<- sysToItemDifDf(mySysList, mySuffixes = c("_male", "_female"), myDims = "all")
myDifPlot<- plotDif(myDifDf,myScale = "centred", mySuffixes = c("_male", "_female"))
## Not run:
# if you run the above example you will have the plot in the object `myDifPlot`.
plot(myDifPlot)

## End(Not run)
```

---

plotItemMap	<i>plotItemMap</i>
-------------	--------------------

---

### Description

Creates a plot (ggplot2 object) of item parameter estimates and abilities on latent trait. Note this is not for use with 'rout' files. See the method `plotRout.itemMap` to the generic function `'plotRout'`

### Usage

```
plotItemMap(mySys, myDims = "D1", abilityType = "PV", ...)
```

### Arguments

<code>mySys</code>	An 'ACER ConQuest' system file object created using the <code>conquestr::ConQuestSys</code> function.
<code>myDims</code>	A string specifying which specific dimensions should be included. The default is "D1", Specific dimensions are specified by the label "D1" for dimensions 1 etc. Alternatively, you can specify <code>myDims = "all"</code> , though what this produces is not currently supported.
<code>abilityType</code>	What kind of person ability estimate should be used? Defaults to plausible values. Alternatively WLE, MLE, EAP.
<code>...</code>	Optional arguments, mostly for debugging, e.g., <code>'setDebug = TRUE'</code> will print temporary data frames.

### Value

A ggplot2 object.

### Examples

```
mySys1<- ConQuestSys()
myItemMap<- plotItemMap(mySys1, myDims = "all")
## Not run:
# if you run the above example you will have the plot in the object `myItemMap`.
plot(myItemMap)

## End(Not run)
```

---

 plotMCC

*plotMCC*


---

### Description

Creates a plot of an item characteristic curve (by response category). For a dichotomous item, this will yield a single curve, for polytomous items this will produce a curve for each response category. Note this is not for use with 'rout' files. See the generic function 'plotRout' for plotting rout files.

### Usage

```
plotMCC(item, data, range = c(-6, 6), e_linetype = "bins", bins = 6)
```

### Arguments

item	Item parameters for a single item.
data	Two vectors of data in an <code>_n_</code> by 2 matrix or data frame, where <code>_n_</code> are the cases in your analysis. The first vector should be item responses. the second vector should be estimated person abilities.
range	Lower and upper bounds to plot over (defaults to <code>c(-6, 6)</code> OR the minimum and maximum estimated ability, whichever is larger).
e_linetype	A string. Should the empirical lines be based on "bins", or "regression". Defaults to "bins"
bins	If <code>e_linetype_</code> is "bins", how many bins should be used to chunk the empirical lines? defaults to 6. Ignored otherwise.

### Value

A ggplot2 object.

### Examples

```
myRout <- ConQuestRout()
myPlot<- plotRout(myRout)
## Not run:
# if you run the above example you will have an ICC plot in the object `myPlot`.
plot(myPlot)

## End(Not run)
```



---

plotRout	<i>plotRout</i>
----------	-----------------

---

### Description

generates a plot from an 'ACER ConQuest' Rout file. use 'ConQuestRout' to read in an Rout file created by a 'plot' command in 'ACER ConQuest'.

### Usage

```
plotRout(myRout, ...)  
  
## S3 method for class 'TestInfo'  
plotRout(myRout, ...)  
  
## S3 method for class 'InformationWithLatentDist'  
plotRout(myRout, ...)  
  
## S3 method for class 'ICC'  
plotRout(myRout, ...)  
  
## S3 method for class 'MCC'  
plotRout(myRout, ...)  
  
## Default S3 method:  
plotRout(myRout, ...)
```

### Arguments

myRout	an R object created by the 'ConQuestRout' function.
...	additional arguments passed into plotting functions (e.g., 'plotZero = TRUE' to control whether the 0 category is plotted in ICC and MCC).

### Value

A ggplot2 object.

### Examples

```
myPlot<- plotRout(ConQuestRout())  
## Not run:  
# if you run the above example you will have a ggplot2 object, using the default Rout file (an ICC).  
str(myPlot)  
  
## End(Not run)  
## to see why we import this, see https://ggplot2.tidyverse.org/articles/ggplot2-in-packages.html
```

---

pvMeanVar

*pvMeanVar*

---

### Description

Applies the law of total variance (EVEs law) to calculate the mean and variance of a set of PVs for one dimension.

### Usage

pvMeanVar(myData)

### Arguments

myData            A matrix of PVs for one dimension: m PVs by n cases.

### Value

A list containing the mean and variance of the PVs.

---

q3ExpCorrect

*q3ExpCorrect*

---

### Description

Helper function to apply correction to correlation matrix. When working with standardised residuals, the expectation of the correlations is  $-1/(L-1)$  rather than 0 See DOI: 10.1177/0013164410379322

### Usage

q3ExpCorrect(myCorMat)

### Arguments

myCorMat            A correlation matrix.

### Value

A correlation matrix with the Q3 statistic correction applied.

---

ReadSys	<i>ReadSys</i>
---------	----------------

---

**Description**

Internal function to read an 'ACER ConQuest' system file. Called by `conquestr::ConQuestSys`.

**Usage**

```
ReadSys(myFile)
```

**Arguments**

<code>myFile</code>	An 'ACER ConQuest' system file created by the 'put' command in 'ACER ConQuest'. The put command must use the option 'compressed = no'.
---------------------	--

**Value**

A list containing the data objects created by 'ACER ConQuest'.

**See Also**

`conquestr::ConQuestSys()`

---

<code>recodeResps</code>	<i>recodeResps</i>
--------------------------	--------------------

---

**Description**

Recode raw item responses for analyses.

**Usage**

```
recodeResps(data, recodeMap, varLabel, rawLabel, recodeLabel)
```

**Arguments**

<code>data</code>	Raw data, a data frame.
<code>recodeMap</code>	A data frame which contains the raw responses and corresponding recoded responses for each of the items in long form.
<code>varLabel</code>	A variable name in <code>recodeMap</code> that identifies the item label.
<code>rawLabel</code>	A variable name in <code>recodeMap</code> that identifies the raw item responses to be recoded.
<code>recodeLabel</code>	A variable name in <code>recodeMap</code> that identifies the new values to recode to.

**Value**

a data frame with raw data recoded according to recodeMap.

---

replaceInDataFrame	<i>iterate through a data frame and use replaceInVector</i>
--------------------	---

---

**Description**

iterate through a data frame and use replaceInVector

**Usage**

```
replaceInDataFrame(d, r, x)
```

**Arguments**

d	A DataFrame.
r	A double - the value to be replaced if it is < -1e300.
x	A double - the value to replace r with.

---

replaceInVector	<i>replace a very large neagtive number with something - usually NA_REAL</i>
-----------------	--

---

**Description**

replace a very large neagtive number with something - usually NA\_REAL

**Usage**

```
replaceInVector(v, r, x)
```

**Arguments**

v	A NumericVector.
r	A double - the value to be replaced if it is < -1e300.
x	A double - the value to repalce r with.

---

searchConQuestSys	<i>searchConQuestSys</i>
-------------------	--------------------------

---

**Description**

Search for object names within a ConQuest System file object.

**Usage**

```
searchConQuestSys(searchString, mySys, value = TRUE, ignore.case = TRUE)
```

**Arguments**

searchString	A string to search within the names of mySys.
mySys	An 'ACER ConQuest' system file object created using the <code>conquestr::ConQuestSys</code> function.
value	Should searchConQuestSys return the name of the object or its index.
ignore.case	Should searchConQuestSys ignore the case of the search term.

**Value**

a string including object names matching the search term

---

steigerStat	<i>steigerStat</i>
-------------	--------------------

---

**Description**

Function to calculate the Steiger statistic. The Steiger statistic is a test of independence of the standardised residuals  $((O-E)/\sqrt{\text{Var}(E)})$ , where  $\text{Var}(E) = p(x)/(1-p(x))$ .

**Usage**

```
steigerStat(myDat, q3Adj = TRUE, fisher = TRUE, dfAdj = FALSE, tpm)
```

**Arguments**

myDat	A data frame or matrix containing standardised residuals.
q3Adj	A bool indicating whether the Q3 correction should be applied.
fisher	A bool indicating whether the Fisher Transform should be applied.
dfAdj	A bool indicating whether the df should be adjusted for sample size, L, and targeting. If dfAdj is TRUE, then you must pass in the optional argument 'tmp' (test-person match)
tpm	A number indicating the test-person match, where 0 indicates that mean item difficulty is equal to mean person ability, and -1 indicates that mean item difficulty is 1 logit below mean person ability.

**Value**

A list of class "steigerStat" with the Steiger Statistic, correlation matrix, and chi square test.

---

summariseCqChain	<i>summariseCqChain</i>
------------------	-------------------------

---

**Description**

takes a data frame created by getCqChain and returns a list reporting the mean and variaince for each parameter

**Usage**

```
summariseCqChain(myChain)
```

**Arguments**

myChain            A data frame returned from getCqChain.

**Value**

A list.

**Examples**

```
## Not run:
summariseCqChain(getCqChain(ConQuestSys()))

## End(Not run)
```

---

sysToBMatrixDf	<i>sysToBMatrixDf</i>
----------------	-----------------------

---

**Description**

Read an R object of class ConQuestSys and create a labelled representation of the B matrix (scoring matrix). This maps item response catagories to items and dimensions. Returns long data frame, where items are duplicated if they are in many dimnsions.

**Usage**

```
sysToBMatrixDf(mySys, applyLabels = TRUE)
```

**Arguments**

mySys            An R object of class ConQuestSys, returned by the function conquestr::ConQuestSys  
 applyLabels    A bool indicating whether labels (e.g., dimension labels) should be appended.

**Value**

A data frame containing R the labelled B matrix.

**Examples**

```
myBMatrix<- sysToBMatrixDf(ConQuestSys())
## Not run:
# if you run the above example you will have the B Matrix from the example system file.
str(myBMatrix)

## End(Not run)
```

---

 sysToItemDifDf

*sysToItemDifDf*


---

**Description**

Creates a data frame that includes the common item parameter estimates from two (or more) system files (e.g., a DIF analysis).

**Usage**

```
sysToItemDifDf(listOfSysFiles, mySuffixes, myDims = "all")
```

**Arguments**

`listOfSysFiles` A list of system files returned from `conquestr::ConQuestSys`

`mySuffixes` a vector of strings specifying the names for the two groups being analysed, e.g., if the two system files are an analysis of boys and girls, the vector may be `'c("_male", "_female")'`.

`myDims` A string specifying if all or specific dimensions should be included. The default is "all", Specific dimensions are specified by the label "D1" for dimensions 1 etc.

**Value**

A data frame object.

**See Also**

`conquestr::plotDif()`

---

testInfoAtTheta	<i>testInfoAtTheta</i>
-----------------	------------------------

---

**Description**

Calculates test information at a value of theta given a list of many sets of item parameters.

**Usage**

```
testInfoAtTheta(myItems, theta)
```

**Arguments**

myItems	A list of matrices of item parameters of the structure used in ‘simplef’
theta	a number.

**Examples**

```
anItem<- matrix(c(0,0,0,1,1,0), nrow = 2, byrow = TRUE)
testInfoAtTheta(list(anItem), 0)
```

---

testInfoOverTheta	<i>testInfoOverTheta</i>
-------------------	--------------------------

---

**Description**

Calculates test information over a range of theta given a list of many sets of item parameters. Returns a data frame with item information at a discrete set of values of theta - useful for plotting item information functions.

**Usage**

```
testInfoOverTheta(myItems, minTheta = -6, maxTheta = 6, stepTheta = 0.1)
```

**Arguments**

myItems	a list of item parameters of the structure used in ‘simplef’
minTheta	The smallest value of ability PDF to plot.
maxTheta	The largest value of ability PDF to plot.
stepTheta	The increment to iterate over the ability PDF. Defaults to 0.01.

**Examples**

```
anItem<- matrix(c(0,0,0,1,1,0), nrow = 2, byrow = TRUE)
testInfoOverTheta(list(anItem))
```



---

thrstThrsh	<i>thrstThrsh</i>
------------	-------------------

---

### Description

Generates Thurstonian Thresholds (sometimes called `_gammas_`) to an item. Thurstonian thresholds are the location on the trait/scale at which the cumulative probability of being in category  $k$ , or any higher category equals some probability (usually 0.5, the default). Thurstonian thresholds are considered a way of describing the difficulty of polytomously scored items and are usually the value used in visualisations like Wright maps. Thurstonian thresholds can only be calculated for items where response categories are scored such that each category can be placed in an order increasing scores (e.g., no ties as per the Ordered Partition model)

### Usage

```
thrstThrsh(myItem, threshP = 0.5, minTheta = -20, maxTheta = 20, convC = 1e-05)
```

### Arguments

myItem	A matrix of parameters for a single item of the structure used in ‘simplef’ (a matrix of $k$ categories by three (category score, delta dot, tau)).
threshP	The probability at which the threshold is calculated (defaults to the usual value of 0.5)
minTheta	The lower-bound starting value of the split-half search used to find the threshold for the category.
maxTheta	The upper-bound starting value of the split-half search used to find the threshold for the category.
convC	The convergence criteria used to determine when the threshold has been found. The difference between ‘threshP’ and the cumulative probability of the category and any higher category at the current value of theta (the current threshold being tested).

### Value

A  $k-1$  by 1 matrix with Thurstonian thresholds for this item. Values are NA when the threshold cannot be calculated.

### Examples

```
myItem<- matrix(
  c(
    0, -0.58 , 0, # delta+tau  thurst thresh (gamma)
    1, -0.58 , 0.776, # 0.196 -1.14
    2, -0.58 , -0.697, # -1.277 -0.93
    3, -0.58 , -0.629, # -1.209 -0.64
    4, -0.58 , 0.55 # -0.03 0.25
  ), ncol =3, byrow=TRUE)
```

```
)
thrstThrsh(myItem)
```

---

timesTwo	<i>Multiply a number by two</i>
----------	---------------------------------

---

### Description

Multiply a number by two

### Usage

```
timesTwo(x)
```

### Arguments

x	A single integer.
---	-------------------

---

transformPvs	<i>transformPvs</i>
--------------	---------------------

---

### Description

Helper function to Transform PVs onto a new metric (e.g., PISA Mean = 500, SD = 100). Uses the method described in the PISA 2012 technical manual.

### Usage

```
transformPvs(x, mT = 0, sdT = 1, weights, data, addToDf = FALSE, debug = TRUE)
```

### Arguments

x	A concatenated vector of varnames in data, PV1, PV2, ..., PVm.
mT	The desired mean of the PVs
sdT	The desired sd of the PVs
weights	The name of the weight variable in 'data' used to calculate the mean and SD across the PVs
data	The data frame that contains the PVs and weights.
addToDf	A Boolean, if TRUE, the transformed PVs are coerced into the DF, data, with name data\$x_T (not yet implemented).
debug	A temporary flag to spit-out objects to global env for chekcing. Will be removed when pushed to CRAN

### Value

a List of transofrmed PVs with as many elements as PVs were listed in 'x'.

# Index

checkItemRespValid, [3](#)  
checkVars, [4](#)  
ConQuestCall, [4](#)  
ConQuestRout, [5](#)  
ConQuestSys, [5](#)  
createConQuestProject, [6](#)

fisherTrnsfrm, [7](#)  
fmtCqItanal, [7](#)

genResponses, [8](#)  
getCqChain, [9](#)  
getCqData, [10](#)  
getCqDataDf, [10](#)  
getCqFit, [11](#)  
getCqHist, [12](#)  
getCqItanal, [12](#)  
getCqParams, [13](#)  
getCqParams2, [13](#)  
getCqTerms, [14](#)  
getCqVars, [15](#)

infoFunc, [15](#)  
informationWrightMap, [16](#)  
informationWrightMap2, [17](#)  
infoWI, [18](#)  
itemInfoAtTheta, [19](#)  
itemInfoOverTheta, [19](#)

makeItemList, [20](#)

plotCqHist, [21](#)  
plotDif, [22](#)  
plotItemMap, [23](#)  
plotMCC, [24](#)  
plotRout, [25](#)  
pvMeanVar, [26](#)

q3ExpCorrect, [26](#)

ReadSys, [27](#)

recodeResps, [27](#)  
replaceInDataFrame, [28](#)  
replaceInVector, [28](#)

searchConQuestSys, [29](#)  
steigerStat, [29](#)  
summariseCqChain, [30](#)  
sysToBMatrixDf, [30](#)  
sysToItemDifDf, [31](#)

testInfoAtTheta, [32](#)  
testInfoOverTheta, [32](#)  
thrstThrsh, [33](#)  
timesTwo, [34](#)  
transformPvs, [34](#)