

Package ‘WrightMap’

June 18, 2022

Type Package

Title IRT Item-Person Map with 'ConQuest' Integration

Version 1.3

Date 2022-06-18

Author David Torres Irribarra & Rebecca Freund

Maintainer David Torres Irribarra <david@torresirribarra.me>

Description A powerful yet simple graphical tool available in the field of psychometrics is the Wright Map (also known as item maps or item-person maps), which presents the location of both respondents and items on the same scale. Wright Maps are commonly used to present the results of dichotomous or polytomous item response models. The 'WrightMap' package provides functions to create these plots from item parameters and person estimates stored as R objects. Although the package can be used in conjunction with any software used to estimate the IRT model (e.g. 'TAM', 'mirt', 'eRm' or 'IRT-Toys' in 'R', or 'Stata', 'Mplus', etc.), 'WrightMap' features special integration with 'ConQuest' to facilitate reading and plotting its output directly. The 'wrightMap' function creates Wright Maps based on person estimates and item parameters produced by an item response analysis. The 'CQmodel' function reads output files created using 'ConQuest' software and creates a set of data frames for easy data manipulation, bundled in a 'CQmodel' object. The 'wrightMap' function can take a 'CQmodel' object as input or it can be used to create Wright Maps directly from data frames of person and item parameters.

License BSD_2_clause + file LICENSE

Depends R (>= 3.0.0)

Imports methods, RColorBrewer

Encoding UTF-8

NeedsCompilation no

Repository CRAN

Date/Publication 2022-06-18 20:10:02 UTC

R topics documented:

CCCfit 2

CQmodel	4
extract.deltas	7
fitgraph	9
ICCfit	10
item.person.data	13
item.side	15
make.deltas	20
make.thresholds	22
person.side	24
plotCI	26
ppPlot	28
WrightMap	30
wrightMap	32
Index	36

CCCfit	<i>Empirical category characteristic curve plot for the Partial Credit Model</i>
--------	--

Description

The CCCfit function is intended for contrasting a Rasch model's expected category characteristic curve against the empirical data from observed responses. The CCCfit function displays the expected probability associated with all response categories and plots the observed response proportions for all non-zero response categories.

Usage

```
CCCfit(itemNumber, observedResponses, personEstimates,
       itemParameters, xlim = c(-4, 4), method = "Quantile", NQtiles = 10)
```

Arguments

itemNumber The position of the item in the test. This position is used to select the column of observed responses and the item difficulty among the item parameters.

observedResponses Data frame or matrix with observed responses. The data frame or matrix should be of size $N * I$, where N is the number of respondents and I is the number of items in the model.

personEstimates A vector of length N containing the model based person estimates or predictions.

itemParameters A data frame or matrix with I rows (one for each item) and M columns, where M is equal to the maximum number of item scores minus 1. This matrix contains the model based estimates for the step parameters (deltas), where column 1 contains the parameter associated with the step between category 0 versus category 1, column 2 the step parameters of category 1 versus category 2, and so on.

xlim	Vector with two values indicating the minimum and maximum values to be used when plotting the item characteristic curve.
method	Selects the Quantile method to group the respondents (see ‘Details’).
NQtiles	This value controls how many grouping will be used: 4 groups cases groups respondents by quartiles, 5 by quintiles, 10 by deciles, etc.

Details

The function uses the step difficulty parameters to generate the model based curve. The observed responses are then grouped using the Quantile method in order to contrast the model predicted response probability with the observed proportion (this is the only method implemented so far). By default the function uses deciles to generate the respondent groups.

Author(s)

David Torres Iribarra

See Also

[ICCfit](#)

Examples

```
##---- Should be DIRECTLY executable !! ----
##-- ==> Define data, use random,
##--or do help(data=index) for the standard data sets.

## The function is currently defined as
function (itemNumber, observedResponses, personEstimates, itemParameters,
  xlim = c(-4, 4), method = "Quantile", NQtiles = 10)
{
  curve.cols <- paste(RColorBrewer::brewer.pal(n = 8, name = "Dark2"),
    "40", sep = "")
  points.cols <- RColorBrewer::brewer.pal(n = 8, name = "Dark2")
  deltas <- itemParameters[itemNumber, ]
  deltas <- deltas[!is.na(deltas)]
  maxCat <- length(deltas)
  probbCCC <- function(theta, deltas) {
    original.length <- length(deltas) + 1
    deltas <- deltas[!is.na(deltas)]
    deltas <- c(0, deltas)
    1N <- length(deltas)
    M <- matrix(rep(NA, 1N), ncol = 1N)
    CM <- matrix(rep(NA, 1N), ncol = 1N)
    M[, 1] <- 0
    CM[, 1] <- 1
    for (k in 2:1N) {
      M[, k] <- M[, (k - 1)] + theta - deltas[k]
      CM[, k] <- CM[, (k - 1)] + exp(M[, k])
    }
  }
  output <- exp(M)/CM[, k]
```

```

        length(output) <- original.length
        output
    }
    categoryProbs <- sapply(seq(xlim[1], xlim[2], length = 100),
        probCCC, deltas = deltas)
    plot(seq(xlim[1], xlim[2], length = 100), categoryProbs[1,
        ], type = "n", axes = FALSE, xlab = "Proficiency", ylab = "Proportion",
        ylim = c(0, 1))
    axis(2, las = 1)
    axis(1)
    lines(seq(xlim[1], xlim[2], length = 100), categoryProbs[1,
        ], type = "l", lwd = 3, lty = 1, col = "grey80")
    nCats <- length(deltas) + 1
    for (i in 2:nCats) {
        lines(seq(xlim[1], xlim[2], length = 100), categoryProbs[i,
            ], lwd = 3, col = curve.cols[i - 1])
    }
    if (method == "Quantile") {
        agg.data <- list()
        size.data <- list()
        for (i in 1:maxCat) {
            recodedResponses <- observedResponses == i
            cutPoints <- quantile(personEstimates, seq(0, 1,
                length = NQtiles + 1))
            agg.data[[i]] <- aggregate(recodedResponses, by = list(cut(personEstimates,
                cutPoints)), FUN = mean, na.rm = TRUE)
            breakMeans <- aggregate(personEstimates, by = list(cut(personEstimates,
                cutPoints)), FUN = mean, na.rm = TRUE)
            agg.data[[i]][, 1] <- breakMeans[, 2]
            agg.data[[i]][, -1][agg.data[[i]][, -1] == 1] <- 0.999
            agg.data[[i]][, -1][agg.data[[i]][, -1] == 0] <- 0.001
            size.data[[i]] <- aggregate(is.na(recodedResponses),
                by = list(cut(personEstimates, cutPoints)), FUN = length)
            size.data[[i]][, 1] <- breakMeans[, 2]
            points(agg.data[[i]][, 1], agg.data[[i]][, itemNumber +
                1], type = "b", pch = i, cex = 0.75, col = points.cols[i],
                lwd = 2)
        }
    }
    legend("right", horiz = FALSE, legend = paste("Cat", seq(1:maxCat)),
        col = points.cols[1:maxCat], pch = 1:maxCat, cex = 0.8,
        bty = "n")
    title(paste("Item", itemNumber))
}

```

Description

The CQmodel function reads ConQuest item parameter and person parameter output files and converts them into a list of data frames for more convenient data processing.

Usage

```
CQmodel(p.est = NULL, show = NULL, p.type = NULL, equation = NULL)
## S3 method for class 'CQmodel'
print(x,...)
## S3 method for class 'SOE'
print(x,...)
```

Arguments

p.est	Conquest person parameters file (EAPs, MLEs, etc.).
show	ConQuest show file.
p.type	Type of person parameter estimate (EAP, MLE or WLE). If not specified, will try to determine from the extension of the p.est file.
equation	String giving the model equation, if the Summary of Estimation table was not included in the show file.
x	Object that determines which function to call.
...	Additional arguments.

Value

CQmodel returns an object of type CQmodel. Usually contains: Tables:

RMP	A list of data frames containing the response model parameter estimates. One data frame is created for each table in the output. Each data frame contains parameter estimates, errors, and fit information.
GIN	A matrix containing the item thresholds (if included in the ConQuest output). The rows are items and the columns are steps.
p.est	A data frame containing the person parameter estimates

Summary of estimation:

SOE	A list of various parameters related to the estimation
-----	--

Items that may be in the SOE list include:

method	Estimation method
distribution	Assumed population distribution
constraint	Constraint
format	Specified format of the datafile
equation	A character string containing the item model (e.g. "item+item*step")
participants	Sample size

deviance	Final deviance of the model
parameters	Total number of estimated parameters
iterations	Number of iterations
seed	Random number generation seed
PV.nodes	Number of nodes used when drawing PVs
fit.nodes	Number of nodes used when computing fit
n.plausible.values	Number of plausible values drawn
max.iterations.no.improvement	Maximum number of iterations without a deviance improvement
max.steps	Maximum number of Newton steps in M-step
zero.perfect.value	Value for obtaining finite MLEs for zero/perfects
termination.reason	Reason for iteration termination
max.iterations	
parameter.change	
deviance.change	

Run details:

run.details A list of details of the run

Items that may be included in the run.details list include:

date	The date of the ConQuest run
data.file	The name of the datafile used
format	The specified format of the datafile
names	Names of items and/or dimensions

Additional items:

deviance	The deviance of the model
equation	A character string containing the model specification (e.g. "item+item*step")
participants	The number of participants
parameters	The number of parameters
title	The run title
reg.coef	Regression coefficients
rel.coef	Reliability coefficients
variances	
nDim	Number of dimensions
dimensions	Dimension names
p.est.type	

Author(s)

Rebecca Freund and David Torres Irribarra

Examples

```
fpath <- system.file("extdata", package="WrightMap")

# Partial credit model
model1 <- CQmodel(p.est = file.path(fpath,"ex2.eap"),
  show = file.path(fpath,"ex2.shw"))
model1 #Shows what tables are available

model1$SOE #Summary of estimation
model1$equation # Model specification
model1$reg.coef # Regression coefficients
model1$rel.coef # Reliability coefficients
model1$variances # Variances

names(model1$RMP) # Names of parameter tables
head(model1$RMP$item) #Item parameters
head(model1$RMP$"item*step") #Item by step parameters

# Complex model
model2 <- CQmodel(file.path(fpath,"ex4a.mle"),
  file.path(fpath,"ex4a.shw"))
model2$equation # Model specification
names(model2$RMP) # Names of parameter tables
head(model2$RMP$"rater*topic*criteria*step") #An interaction table

model1$GIN #Item thresholds
model2$GIN #Item thresholds

head(model1$p.est) ##EAPs
head(model2$p.est) ##MLEs
```

extract.deltas

Extract Master's Delta parameters from a TAM model.

Description

This function takes as its input a TAM object. It adds reads the TAM item parameters and organizes them into a matrix that can be used as input in the [CCCfit](#) function.

Usage

```
extract.deltas(tamObject)
```

Arguments

tamObject TAM object containing the results of a a Rasch model or Partial Credit model.

Details

This function organizes the item parameter results into a matrix where each row is contains the parameters associated with an item and each columns is contains the parameters associated with a specific step (score 0 vs score 1, score 1 vs score 2, etc.). The resulting matrix will have as many rows as items and as many columns as the maximum number of steps among the items.

Value

A matrix in which each row is an item and each column is a step

Author(s)

David Torres Iribarra

References

Masters, G. N. (1982). A Rasch model for partial credit scoring. *Psychometrika*, 47(2), 149-174.

See Also

[CCCfit](#) [make.thresholds](#)

Examples

```
##---- Should be DIRECTLY executable !! ----
##-- ==> Define data, use random,
##--or do help(data=index) for the standard data sets.

## The function is currently defined as
function (tamObject)
{
  delta.long <- tamObject$xsi
  n.deltas <- apply(tamObject$B, 1, max)
  delta.mat <- matrix(NA, nrow = length(n.deltas), ncol = max(n.deltas))
  matCoords.row <- rep(1:length(n.deltas), n.deltas)
  matCoords.col <- c()
  for (i in 1:length(n.deltas)) {
    for (j in 1:n.deltas[i]) {
      matCoords.col <- c(matCoords.col, j)
    }
  }
  delta.long$matCoords.row <- matCoords.row
  delta.long$matCoords.col <- matCoords.col
  for (k in 1:nrow(delta.long)) {
    delta.mat[delta.long$matCoords.row[k], delta.long$matCoords.col[k]] <- delta.long$xsi[k]
  }
  delta.mat
}
```



```
}

```

fitgraph

Item Fit Graphs

Description

This function creates a graphical summary of the item fit information.

Usage

```
## Default S3 method:
fitgraph(fitEst, fitLB, fitUB, itemLabels, mainTitle = "Fit Plot",
pch = 18, fitColours = c("gray70", "gray60", "gray50", "gray40", "gray0"),
xlab = "Items", cex = 1.25, ...)
## S3 method for class 'numeric'
fitgraph(fitEst, fitLB, fitUB, itemLabels, mainTitle = "Fit Plot",
pch = 18, fitColours = c("gray70", "gray60", "gray50", "gray40", "gray0"),
xlab = "Items", cex = 1.25, ...)
## S3 method for class 'CQmodel'
fitgraph(fitEst, table = NULL, fit.type = "W", itemLabels = NULL, ...)
## S3 method for class 'character'
fitgraph(fitEst, ...)
```

Arguments

fitgraph arguments:

	vector of item fit estimates. Could also be a CQmodel object or name of a ConQuest show file.
fitEst	vector of lower bounds for critical intervals for each item.
fitUB	vector of upper bounds for critical intervals for each item.
itemLabels	vector of item labels.
mainTitle	string containing the title of the plot.
pch	number or vector indicating the type of symbols to be used for each item.
fitColours	Color that will be used to shade the critical interval area.
xlab	Label of the x-axis. The default is 'items'.
cex	Size of the x-axis label.
...	Additional parameters.
	Argument to use when passing a CQmodel object:
table	Name of the RMP table that for which the fit will be plotted. By default fitgraph will plot the first RMP table of the CQmodel object, this argument overrides this default.

`fit.type` Type of fit estimate that will be used, it can be W for Weighted Fit (i.e. Infit, the default), or U for Unweighted Fit (i.e. Outfit). Called `type` in previous versions; use of that parameter is deprecated to avoid collision with the `type` parameter in the `link{plot}` function.

Author(s)

David Torres Irribarra and Rebecca Freund.

References

Wilson, M. (2005). Constructing measures: An item response modeling approach.

Examples

```
# Generating mock data
sampleLabels <- paste('item',1:10)

fitBounds <- (abs(rnorm(10, mean = 0, sd = .05)) * 2)
fitEst <- rnorm(10, mean = 1, sd = .1)

fitLB <- 1 - fitBounds
fitUB <- 1 + fitBounds
par("mar")
# running fitgraph
fitgraph(fitEst,fitLB,fitUB,itemLabels=sampleLabels)

#From ConQuest output:

fpath <- system.file("extdata", package="WrightMap")

fitgraph(file.path(fpath,"ex2.shw"))
```

ICCfit

Empirical item characteristic curve plot for the Rasch Model~

Description

The `ICCfit` function is intended for contrasting a Rasch model's expected item characteristic curve against the empirical data from dichotomous responses. The `ICCfit` function displays a confidence interval for the model based curve and plots the confidence interval for the empirical proportions.

Usage

```
ICCfit(itemNumber, observedResponses, personEstimates,
       itemParameters, xlim = c(-4, 4), method = "Quantile", NQtiles = 10)
```

Arguments

itemNumber	The position of the item in the test. This position is used to select the column of observed responses and the item difficulty among the item parameters.
observedResponses	Data frame or matrix with observed responses. The data frame or matrix should be of size $N * I$, where N is the number of respondents and I is the number of items in the model.
personEstimates	A vector of length N containing the model based person estimates or predictions.
itemParameters	A data frame or matrix of dimensions $I * 2$ containing the model based item difficulty estimates in the first column and the parameter standard error in the second column.
xlim	Vector with two values indicating the minimum and maximum values to be used when plotting the item characteristic curve.
method	Selects the method used to group the respondents: <code>Quantile</code> (default), <code>ByPersonEstimate</code> , and <code>Histogram</code> (see 'Details').
NQtiles	When using the <code>Quantile</code> method this value controls how many grouping will be used: 4 groups cases groups respondents by quartiles, 5 by quintiles, 10 by deciles, etc.

Details

The function uses the item difficulty parameter to generate the model based curve and the item difficulty parameter standard error to plot a confidence interval around the curve. The observed responses are then grouped using the selected method in order to contrast the model predicted response probability with the observed proportion. By default the function uses deciles to generate the respondent groups. The function allows the method `ByPersonEstimate` in order to make a different group for each observed person estimate (potentially useful when analyzing test data with large numbers with no missing data), and the `Histogram` method, which uses the Freedman-Diaconis algorithm to select the width of the bands used for grouping.

Author(s)

David Torres Iribarra

See Also

[CCCfit](#)

Examples

```
##---- Should be DIRECTLY executable !! ----
##-- ==> Define data, use random,
##--or do help(data=index) for the standard data sets.

## The function is currently defined as
function (itemNumber, observedResponses, personEstimates, itemParameters,
        xlim = c(-4, 4), method = "Quantile", NQtiles = 10)
```

```

{
  propCI <- function(propVector, nVector, alpha = 0.05) {
    propSE <- sqrt(propVector * (1 - propVector)/nVector)
    propLB <- propVector - (propSE * qnorm(1 - (alpha/2)))
    propUB <- propVector + (propSE * qnorm(1 - (alpha/2)))
    data.frame(propSE = propSE, propLB = propLB, propUB = propUB)
  }
  plotICC <- function(difficulty, range = xlim) {
    invlogit <- function(x) {
      1/(1 + exp(-x))
    }
    params <- data.frame(ability = seq(-4, 4, length.out = 1000),
      difficulty = difficulty)
    probs <- invlogit(params[, 1] - params[, 2])
    lines(params[, 1], probs)
  }
  plotICCErrors <- function(difficulty, dSE, range = xlim) {
    invlogit <- function(x) {
      1/(1 + exp(-x))
    }
    params <- data.frame(ability = seq(-4, 4, length.out = 1000),
      lb = difficulty - 1.96 * dSE, ub = difficulty + 1.96 *
      dSE)
    probslb <- invlogit(params[, 1] - params[, 2])
    probsub <- invlogit(params[, 1] - params[, 3])
    xCoords <- c(ability = seq(-4, 4, length.out = 1000),
      ability = seq(4, -4, length.out = 1000))
    yCoords <- c(probslb, rev(probsub))
    polygon(xCoords, yCoords, col = "grey75", border = NA)
  }
  if (method == "ByPersonEstimate") {
    aggdata <- aggregate(observedResponses, by = list(round(personEstimates,
      2)), FUN = mean, na.rm = TRUE)
    aggdata[, -1][aggdata[, -1] == 1] <- 0.999
    aggdata[, -1][aggdata[, -1] == 0] <- 0.001
    sampleSizeAggdata <- aggregate(is.na(observedResponses),
      by = list(round(personEstimates, 2)), FUN = length)
  }
  if (method == "Quantile") {
    cutPoints <- quantile(personEstimates, seq(0, 1, length = NQtiles +
      1))
    aggdata <- aggregate(observedResponses, by = list(cut(personEstimates,
      cutPoints)), FUN = mean, na.rm = TRUE)
    breakMeans <- aggregate(personEstimates, by = list(cut(personEstimates,
      cutPoints)), FUN = mean, na.rm = TRUE)
    aggdata[, 1] <- breakMeans[, 2]
    aggdata[, -1][aggdata[, -1] == 1] <- 0.999
    aggdata[, -1][aggdata[, -1] == 0] <- 0.001
    sampleSizeAggdata <- aggregate(is.na(observedResponses),
      by = list(cut(personEstimates, cutPoints)), FUN = length)
    sampleSizeAggdata[, 1] <- breakMeans[, 2]
  }
  if (method == "Histogram") {

```

```

histData <- hist(personEstimates, breaks = "FD", plot = FALSE)
cutPoints <- histData$breaks
aggdata <- aggregate(observedResponses, by = list(cut(personEstimates,
  cutPoints)), FUN = mean, na.rm = TRUE, drop = FALSE)
breakMeans <- aggregate(personEstimates, by = list(cut(personEstimates,
  cutPoints)), FUN = mean, na.rm = TRUE, drop = FALSE)
aggdata[, 1] <- histData$mids
aggdata[, -1][aggdata[, -1] == 1] <- 0.999
aggdata[, -1][aggdata[, -1] == 0] <- 0.001
sampleSizeAggdata <- aggregate(is.na(observedResponses),
  by = list(cut(personEstimates, cutPoints)), FUN = length,
  drop = FALSE)
sampleSizeAggdata[, 1] <- histData$mids
}
plot(aggdata[, 1], aggdata[, itemNumber + 1], type = "n",
  axes = FALSE, , ylab = "Proportion", xlab = "Proficiency",
  ylim = c(0, 1), xlim = xlim)
plotICCErrors(itemParameters[itemNumber, 1], itemParameters[itemNumber,
  2])
plotICC(itemParameters[itemNumber, 1])
cbind(aggdata[, 1], propCI(aggdata[, itemNumber + 1], sampleSizeAggdata[,
  itemNumber + 1]))
apply(cbind(aggdata[, 1], propCI(aggdata[, itemNumber + 1],
  sampleSizeAggdata[, itemNumber + 1])), 1, function(x) segments(x0 = x[1],
  y0 = x[3], x1 = x[1], y1 = x[4], col = "#31333450"))
points(aggdata[, 1], aggdata[, itemNumber + 1], pch = 18,
  cex = 0.75, col = "#31333450")
axis(2, las = 1)
axis(1)
title(paste("Item", rownames(itemParameters)[itemNumber]))
print(sampleSizeAggdata)
}

```

item.person.data

Processing CQmodel objects for plotting

Description

The `itemData` and `personData` functions take CQmodel objects (or ConQuest output files) as inputs and return a vector or matrix. They were originally developed for use by [wrightMap](#), but are separated out here to allow the outputs to be sent to other plotting functions.

Usage

```

itemData(thresholds, ...)
## Default S3 method:
itemData(thresholds, item.type = "deltas",...)
## S3 method for class 'character'
itemData(thresholds, p.type = NULL, equation = NULL, ...)
## S3 method for class 'CQmodel'

```

```
itemData(thresholds, item.table = NULL, interactions = NULL,
step.table = NULL, item.type = "default", throld = 0.5, ...)
```

```
personData(thetas,...)
## Default S3 method:
personData(thetas,...)
## S3 method for class 'character'
personData(thetas, p.type = NULL,...)
## S3 method for class 'CQmodel'
personData(thetas,...)
```

Arguments

	itemData arguments:
	Usually, a CQmodel object or the name of a ConQuest show file. Will also accept a matrix, but this is only really for use within other functions. In general make.thresholds should be used instead.
thesholds	Indicates whether to use thresholds or deltas.
equation	string giving the model equation, if the Summary of Estimation table was not included in the show file.
item.table	Name of RMP table to use for the main effect of the item parameters.
interactions	Name of RMP interaction table to use in addition to item.table.
step.table	Name of RMP table to use in addition to item.table.
throld	The probability level to be used for calculating thresholds.
...	Additional parameters to pass to make.thresholds .
	personData arguments:
thetas	a CQModel object or the name of the Conquest person parameters file (EAPs, MLEs, etc.)
p.type	Type of person parameter estimate (EAP, MLE or WLE).

Details

The itemData and personData functions are usually called by [wrightMap](#). They can also be called directly.

For the itemData function, note that the item.table, interactions, and step.table parameters must be the exact name of specific RMP tables. You cannot specify an interactions table or a step table without also specifying an item table (although JUST an item table is fine). If your model equation is more complicated, you will have to either use a GIN table or specify in the function call which tables to use for what. A model of the form item + item * step + booklet, for example, will not run unless there is a GIN table or you have defined at least the item.table.

Value

The itemData functions return a vector of item parameters, or a matrix in which the rows are items and the columns are steps. The personData functions return a vector of person parameters, or a matrix in which the rows are persons and the columns are dimensions.

Author(s)

Rebecca Freund and David Torres Irribarra

See Also

[item.side](#) [person.side](#) [make.thresholds](#) [make.deltas](#) [wrightMap](#)

Examples

```
#As a call from wrightMap:

fpath <- system.file("extdata", package="WrightMap")

model1 <- CQmodel(file.path(fpath,"ex2a.eap"), file.path(fpath,"ex2a.shw"))
# Making thresholds if there are no GIN tables (partial credit model)
wrightMap(model1, type = "thresholds")

#Complex model:

model2 <- CQmodel(file.path(fpath,"ex4a.mle"), file.path(fpath,"ex4a.shw"))
wrightMap(model2, item.table = "rater")
wrightMap(model2, item.table = "rater", interactions = "rater*topic",
  step.table = "topic")

# Plotting item results

fpath <- system.file("extdata", package="WrightMap")
model3 <- CQmodel(file.path(fpath,"ex2a.eap"), file.path(fpath,"ex2a.shw"))
m3.item <- itemData(model3)

dev.new(width=10, height=10)

#control of oma allows us to give more space to longer item names
itemModern(m3.item, label.items.srt= 90, oma = c(2,0,0,2))
itemClassic(m3.item)
itemHist(m3.item)

m3.person <- personData(model3)
personHist(m3.person)
personDens(m3.person)
```

item.side

Wright Map item sides

Description

Draw the item side of a Wright Map in a variety of styles. Intended to be primarily called by [wrightMap](#), but also available for use on their own.

Usage

```
itemModern(thr, yRange = NULL, axis.items = "Items", show.thr.sym = TRUE
, thr.sym.cex = 0.8, thr.sym.lwd = 1, thr.sym.pch = 23
, thr.sym.col.fg = rgb(0, 0, 0, 0.3), thr.sym.col.bg = rgb(0, 0, 0, 0.3)
, show.thr.lab = TRUE, thr.lab.pos = c(2, 4), thr.lab.text = NULL
, thr.lab.col = "black", thr.lab.cex = 0.5, thr.lab.font = 2, label.items.rows = 1
, label.items.srt = 0, label.items = NULL, label.items.cex = 0.6
, label.items.ticks = TRUE, axis.logits = "Logits", show.axis.logits = "R"
, oma = c(0, 0, 0, 3), cutpoints = NULL, vertLines = FALSE, ...)
```

```
itemClassic(thr, yRange = NULL, axis.items = "Items", axis.logits = "Logits"
, show.axis.logits = "R", oma = c(0, 0, 0, 3), cutpoints = NULL, ...)
```

```
itemHist(thr, yRange = NULL, axis.items = "Items", axis.logits = "Logits"
, show.axis.logits = "R", oma = c(0, 0, 0, 3), cutpoints = NULL, ...)
```

Arguments

General arguments:

vector or matrix of threshold parameters. If a matrix, items should be in the rows and steps in the columns.

yRange vector with 2 elements specifying the lower and upper limits of the plot's y-axis.

axis.items title of the x-axis.

axis.logits title of the y-axis.

show.axis.logits

if equal to "R" or "L", draws a logit axis on the right or left. Will also draw an axis on the right if the value is codeTRUE. If any other value, the axis is not drawn.

oma values to use for the oma parameter (see [par](#))

cutpoints values at which to draw horizontal lines (see [cutLines](#))

... additional arguments to [cutLines](#)

itemModern arguments:

show.thr.sym logical. If TRUE (default), the plot will show symbols for the item thresholds.

thr.sym.cex an integer, vector or matrix of numerical values giving the amount by which the threshold symbols should be magnified relative to the default.

thr.sym.lwd an integer, vector or matrix of positive numbers specifying the width of the lines used in the threshold symbols.

thr.sym.pch an integer, vector or matrix of integers specifying a symbol or a single character to be used to represent the item thresholds.

thr.sym.col.fg an integer, vector or matrix of numerical values indicating the foreground color to be used in the thresholds labels.

thr.sym.col.bg an integer, vector or matrix of numerical values indicating the background color to be used in the thresholds labels.

<code>show.thr.lab</code>	logical. If TRUE (default), the plot will show labels for the item thresholds.
<code>thr.lab.pos</code>	an integer, vector or matrix containing the position in which to display the label for each threshold label. Values of 1, 2, 3 and 4, respectively indicate positions below, to the left of, above and to the right of the specified coordinates.
<code>thr.lab.text</code>	a matrix containing the labels to display for each threshold. In the matrix each row represents an item and each column represents a level.
<code>thr.lab.col</code>	a matrix containing the color to display for each threshold label. In the matrix each row represents an item and each column represents a level.
<code>thr.lab.cex</code>	an integer, vector or matrix of numerical values giving the amount by which the threshold labels should be magnified relative to the default.
<code>thr.lab.font</code>	an integer, vector or matrix which specifies which font to use for threshold labels. 1 corresponds to plain text, 2 to bold face (the default), 3 to italic and 4 to bold italic.
<code>label.items.rows</code>	an integer indicating the number of rows used to display the item labels. Can take values 1 (default), 2 and 3. Useful when item labels are overlapping.
<code>label.items.srt</code>	angle of rotation for item labels. It only works if <code>label.item.rows</code> is set to 1.
<code>label.items</code>	a vector of strings containing the labels identifying the items.
<code>label.items.cex</code>	an integer, vector or matrix of numerical values giving the amount by which the threshold labels should be magnified relative to the default.
<code>label.items.ticks</code>	logical. If TRUE (default), ticks are drawn in the x-axis of the item side.
<code>vertLines</code>	logical. If TRUE, light gray vertical lines are drawn to mark all symbols associated with an item or feature in the x-axis.

Details

These functions are designed as helper functions for `wrightMap` to draw the item side of a map. When called outside of that function, they can be used to create more customized maps. Possible uses include:

- draw an item map on its own
- compare two item maps in a single figure
- draw a Wright Map with the item side on the left and the person side on the right
- etc.

The `itemClassic` style draws a stacked plot, similar to the Wright Maps available in ConQuest text output files. The `itemModern` style is the default style for `wrightMap` which plots each item as a column of difficulty parameters. The `itemHist` style plots a histogram.

Note

When combining with a `person.side` function, note that those functions use `split.screen`, which are incompatible with `layout` and some other plotting functions. Note also that all graphs on a single plot should usually have their `yRange` explicitly specified to ensure that values are comparable across plots. To plot data from ConQuest output, use `itemData` first to extract the data table.

Author(s)

Rebecca Freund and David Torres Irribarra

See Also

[person.side](#) [itemData](#) [wrightMap](#)

Examples

```
#As a call from wrightMap:

## Mock results
uni.proficiency <- rnorm(1000, mean = -0.5, sd = 1)

items.loc <- sort( rnorm( 20))
thresholds <- data.frame(
  l1 = items.loc - 0.5 ,
  l2 = items.loc - 0.25,
  l3 = items.loc + 0.25,
  l4 = items.loc + 0.5)

## Setting up labels, colors and symbols

thresholds.labels <- data.frame(
  l1 = paste('Lev',rep(1,20),sep = ''),
  l2 = paste('Lev',rep(2,20),sep = ''),
  l3 = paste('Lev',rep(3,20),sep = ''),
  l4 = paste('Lev',rep(4,20),sep = ''))

thresholds.colors <- data.frame(
  l1 = rep( 'green',20),
  l2 = rep( 'red',20),
  l3 = rep( 'yellow',20),
  l4 = rep( 'blue',20))

thresholds.symbols <- data.frame(
  l1 = rep( 15,20),
  l2 = rep( 16,20),
  l3 = rep( 17,20),
  l4 = rep( 18,20))

wrightMap( uni.proficiency, thresholds
  , thr.lab.text = thresholds.labels
  , thr.lab.col = thresholds.colors
  , thr.sym.pch = thresholds.symbols
  )

#As direct call:

## Plotting results of a unidimensional Rating Scale Model
```

```

items.loc <- sort( rnorm( 20))
thresholds <- data.frame(
  l1 = items.loc - 0.5 ,
  l2 = items.loc - 0.25,
  l3 = items.loc + 0.25,
  l4 = items.loc + 0.5)

itemModern(thresholds)
itemClassic(thresholds)
itemHist(thresholds)

## Plotting ConQuest results

fpath <- system.file("extdata", package="WrightMap")
model1 <- CQmodel(file.path(fpath,"ex2a.eap"), file.path(fpath,"ex2a.shw"))
m1.item <- itemData(model1)

#control of oma allows us to give more space to longer item names
itemModern(m1.item, label.items.srt= 90, oma = c(3,0,0,3))
itemClassic(m1.item)
itemHist(m1.item)

## Creating a Wright Map with item side on the left

multi.proficiency <- data.frame(
  d1 = rnorm(1000, mean = -0.5, sd = 1),
  d2 = rnorm(1000, mean = 0.0, sd = 1),
  d3 = rnorm(1000, mean = +0.5, sd = 1))

# split.screen: Set up a split screen with the left side 80 percent of the screen
# yRange = c(-3,4): Set the yRange to be the same for both sides
# axis.logits.side = "L": Move the item logit axis to the left
# oma = c(0,0,0,2): Adjust the spacing between the graphs
# mtext("Wright Map", side = 3, font = 2, line = 1): add a title
# screen(2): Start drawing on the second screen

split.screen(figs = matrix(c(0,.8,0,1,.8,1,0,1),ncol = 4, byrow = TRUE))
itemModern(thresholds, yRange = c(-3,4), show.axis.logits = "L", oma = c(0,0,0,2))
mtext("Wright Map", side = 3, font = 2, line = 1)
screen(2)
personHist(multi.proficiency, axis.persons = "",yRange = c(-3,4)
, axis.logits = "Persons", show.axis.logits = FALSE)

## Creating a multidimensional Wright Map with each dimension separate

## Mock results

d1 = rnorm(1000, mean = -0.5, sd = 1)
d2 = rnorm(1000, mean = 0.0, sd = 1)

dim1.diff <- rnorm(5)
dim2.diff <- rnorm(5)

```

```

dev.new(width=10, height=10)
split.screen(figs = matrix(c(0,.1,0,1,
  .12,.6,0,1,
  .5,.6,0,1,
  .5,1,0,1),ncol = 4,byrow = TRUE))

personDens(d1,yRange = c(-3,3),show.axis.logits = FALSE,axis.logits = "")
screen(2)
itemModern(dim1.diff,yRange = c(-3,3),show.axis.logits = FALSE)
mtext("Wright Map", side = 3, font = 2, line = 1)
screen(3)
personDens(d2,yRange = c(-3,3),show.axis.logits = FALSE,axis.logits = ""
,axis.persons = "",dim.names = "Dim2")
screen(4)
itemModern(dim2.diff,yRange = c(-3,3),show.axis.logits = FALSE
,label.items = paste("Item",6:10))

```

make.deltas

Calculate Master's Delta parameters.

Description

This function takes as its input a CQmodel object or the name of a ConQuest show file. It adds together the parameters as specified by the user, or if no tables are specified it reads the model equation to determine the appropriate tables to sum. This function is used by wrightMap to draw the item side of the map when a CQmodel is passed to wrightMap.

Usage

```

make.deltas(item.params, ...)
## S3 method for class 'character'
make.deltas(item.params, ...)
## S3 method for class 'CQmodel'
make.deltas(item.params, item.table = NULL, interactions = NULL,
step.table = NULL, item.sign = NULL, inter.sign = NULL,
step.sign = NULL, ...)
## Default S3 method:
make.deltas(item.params, cross.params = 0, step.params = 0,
item.sign = 1, step.sign = 1, inter.sign = 1, ...)

```

Arguments

<code>item.params</code>	The item parameters. Can either be a vector, a CQmodel object, or a path to a ConQuest show file
<code>item.table</code>	If <code>item.params</code> is a CQmodel object or a path to a ConQuest show file, <code>item.table</code> is the name of the items table. Commonly "item" but can be any string representing the name of a table in the ConQuest show file. This identifies what variable will form the rows of the delta matrix. If not specified, will be the first variable mentioned in the model equation.
<code>interactions</code>	If <code>item.params</code> is a CQmodel object or a path to a ConQuest show file, <code>item.table</code> is the name of the table with the interactions (if present). Commonly "item*step" but can be any string containing "*" that is the name of a table in the ConQuest show file. Should be the product of the <code>item.table</code> variable and the <code>step.table</code> variable (if present). If not specified, will be the product term of the model equation.
<code>step.table</code>	If <code>item.params</code> is a CQmodel object or a path to a ConQuest show file, <code>step.table</code> is the name of the steps table (if present). Commonly "step" but can be any string representing the name of a table in the ConQuest show file. This identifies what variable will form the columns of the delta matrix. If not specified, will be the second variable mentioned in the model equation.
<code>item.sign</code>	Can be 1 or -1. Indicates whether the item parameters should be added or subtracted.
<code>inter.sign</code>	Can be 1 or -1. Indicates whether the interaction parameters should be added or subtracted.
<code>step.sign</code>	Can be 1 or -1. Indicates whether the step parameters should be added or subtracted.
<code>cross.params</code>	If <code>item.params</code> is a vector, use this parameter to pass a matrix of interaction parameters.
<code>step.params</code>	If <code>item.params</code> is a vector, use this parameter to pass a matrix of step parameters.
<code>...</code>	Additional parameters

Details

This function reshapes the tables in the ConQuest show file and adds the step parameters to the appropriate items. The vector version of this is rarely called by the user.

Value

A matrix in which each row is an item and each column is a step

Author(s)

Rebecca Freund & David Torres Irribarra

See Also

[make.thresholds](#) [CQmodel](#) [wrightMap](#)

Examples

```
fpath <- system.file("extdata", package="WrightMap")

# Partial credit model
model1 <- CQmodel(file.path(fpath,"ex2a.eap"), file.path(fpath,"ex2a.shw"))
make.deltas(model1)

# Rating scale model
model2 <- CQmodel(file.path(fpath,"ex2b.eap"), file.path(fpath,"ex2b-2.shw"))
make.deltas(model2)

# Raters, criteria, topics
model3 <- CQmodel(file.path(fpath,"ex4a.mle"), file.path(fpath,"ex4a.shw"))
make.deltas(model3, item.table = "rater")
make.deltas(model3, item.table = "rater", interactions = "rater*topic", step.table = "topic")
```

make.thresholds

Calculate Thurstonian thresholds.

Description

This function accepts a matrix of delta parameters and converts them to thresholds (using a threshold of .5). It can also take as input a CQmodel object or a filename of a ConQuest show file.

Usage

```
make.thresholds(item.params, ...)
## S3 method for class 'character'
make.thresholds(item.params, design.matrix = "normal",...)
## S3 method for class 'CQmodel'
make.thresholds(item.params,item.table = NULL, interactions = NULL
,step.table = NULL, design.matrix = "normal", throld = 0.5, alpha = 1,...)
## Default S3 method:
make.thresholds(item.params, design.matrix = "normal"
, make.from = "deltas", theta.interval = c(-10, 10), throld = 0.5, alpha = 1
, c.params = 0,...)
## S3 method for class 'matrix'
make.thresholds(item.params, design.matrix = "normal"
, make.from = "deltas", theta.interval = c(-10, 10), throld = 0.5
, alpha = 1, c.params = 0,...)
```

Arguments

item.params The item parameters. Can either be a matrix, a CQmodel object, or a path to a ConQuest show file

design.matrix Can be "normal" or "ConQuest". Note that for a CQmodel object or ConQuest file, should be normal, NOT ConQuest.

<code>make.from</code>	Specifies whether the <code>item.params</code> matrix contains threshold or delta parameters.
<code>item.table</code>	If <code>item.params</code> is a CQmodel object or a path to a ConQuest show file, <code>item.table</code> is the name of the items table. Commonly "item" but can be any string representing the name of a table in the ConQuest show file. This identifies what variable will form the rows of the thresholds matrix. If not specified, will be the first variable mentioned in the model equation.
<code>interactions</code>	If <code>item.params</code> is a CQmodel object or a path to a ConQuest show file, <code>item.table</code> is the name of the table with the interactions (if present). Commonly "item*step" but can be any string containing "*" that is the name of a table in the ConQuest show file. Should be the product of the <code>item.table</code> variable and the <code>step.table</code> variable (if present). If not specified, will be the product term of the model equation.
<code>step.table</code>	If <code>item.params</code> is a CQmodel object or a path to a ConQuest show file, <code>step.table</code> is the name of the steps table (if present). Commonly "step" but can be any string representing the name of a table in the ConQuest show file. This identifies what variable will form the columns of the thresholds matrix. If not specified, will be the second variable mentioned in the model equation.
<code>theta.interval</code>	If <code>item.params</code> is a matrix, <code>theta.interval</code> specifies over what interval to search for the parameters.
<code>throlld</code>	The probability level to use for calculating the thresholds.
<code>alpha</code>	A vector or single value for the slope parameter or parameters.
<code>c.params</code>	A vector or single value for the guessing parameter or parameters.
<code>...</code>	Additional parameters.

Value

A matrix of threshold parameters.

Author(s)

Daniel Coulter Furr, Rebecca Freund, & David Torres Irribarra

See Also

[make.deltas](#) [itemData](#) [CQmodel](#) [wrightMap](#)

Examples

```
fpath <- system.file("extdata", package="WrightMap")

# Partial credit model
model1 <- CQmodel(file.path(fpath, "ex2a.eap"), file.path(fpath, "ex2a.shw"))
deltas <- make.deltas(model1)
make.thresholds(deltas)
make.thresholds(model1)
```

person.side *Wright Map person sides*

Description

Draw the person side of a Wright Map in a variety of styles. Intended to be primarily called by `wrightMap`, but also available for use on their own.

Usage

```
personHist(thetas, yRange = NULL, breaks = "FD", dim.lab.cex = 0.6, dim.lab.side = 3
, dim.lab.adj = 0.5, dim.names = NULL, dim.color = "white", person.points = NULL
, person.range = NULL, p.point.col = "gray45", p.range.col = "gray75"
,axis.persons = "Respondents", oma = c(0, 5, 0, 5), axis.logits = "Logits"
, show.axis.logits = TRUE,...)
```

```
personDens(thetas, yRange = NULL, dim.lab.cex = 0.6, dim.lab.side = 3, dim.lab.adj = 0.5
,dim.names = NULL,dim.color = "black",person.points = NULL, person.range = NULL
, p.point.col = "black", p.range.col = "gray70",oma = c(0, 5, 0, 5)
, axis.logits = "Logits",show.axis.logits = TRUE, axis.persons = "Respondents",...)
```

Arguments

<code>thetas</code>	vector or matrix of person parameters. If a matrix, persons should be the rows and dimensions the columns.
<code>yRange</code>	vector with 2 elements specifying the lower and upper limits of the plot's y-axis.
<code>dim.lab.cex</code>	An integer specifying the amount the dimension labels should be magnified relative to the default.
<code>dim.lab.side</code>	an integer specifying in which side to plot the dimension names. Values of 1, 2, 3 (default) and 4, respectively indicate positions below, to the left of, above and to the right of the person distributions.
<code>dim.lab.adj</code>	a numerical value adjusting the position of the dimension names.
<code>dim.names</code>	a string or a vector of strings containing the names of each one of the dimensions.
<code>dim.color</code>	a numerical value or vector indicating the colors to be used for representing each dimension.
<code>person.points</code>	a vector of individual values to highlight
<code>person.range</code>	Can be a pair of values, an even-lengthed vector, or a matrix with two rows. Values indicate the start and endpoints of ranges to highlight. If a matrix, the first row should be lower bounds and the second row upper bounds of the ranges. If a vector, the values should alternate: (lower1,upper1,lower2,upper2,...).
<code>p.point.col</code>	a string or vector of strings indicating the color to use for the highlighted points
<code>p.range.col</code>	a string or vector of strings indicating the color to use for the highlighted ranges.
<code>axis.persons</code>	title of the y-axis on the left side.

oma	values to use for the oma parameter (see par)
show.axis.logits	logical indicating whether to show the logit axis
axis.logits	title of the y-axis on the right side
...	Not used.
	For personHist:
breaks	See hist). This argument is passed directly to hist, so it will accept all the options detailed in that function's manual.

Details

These functions are designed as helper functions for [wrightMap](#) and [ppPlot](#) to draw the person side of a map. When called outside of that function, they can be used to create more customized maps. Possible uses include:

- draw a person map on its own
- compare two person maps in a single figure
- draw a Wright Map with the item side on the left and the person side on the right
- etc.

The personHist style, the default, draws the person distribution as a histogram, and is equivalent to the use.hist = TRUE option from previous versions of wrightMap. The personDens style draws a density plot.

The person.points, person.range, p.point.col, and p.range.col parameters are called directly by [ppPlot](#) to show the estimate and standard deviation for a single person. However, they can also be specified without using [ppPlot](#) to highlight arbitrary values or ranges.

Author(s)

Rebecca Freund and David Torres Irribarra

See Also

[item.side](#) [personData](#) [wrightMap](#) [ppPlot](#)

Examples

```
# Creating a Wright Map with item side on the left

multi.proficiency <- data.frame(
  d1 = rnorm(1000, mean = -0.5, sd = 1),
  d2 = rnorm(1000, mean = 0.0, sd = 1),
  d3 = rnorm(1000, mean = +0.5, sd = 1))

items.loc <- sort( rnorm( 20))
thresholds <- data.frame(
```

```

l1 = items.loc - 0.5 ,
l2 = items.loc - 0.25,
l3 = items.loc + 0.25,
l4 = items.loc + 0.5)

# split.screen: Set up a split screen with the left side 80 percent of the screen
# yRange = c(-3,4): Set the yRange to be the same for both sides
# axis.logits.side = "L": Move the item logit axis to the left
# oma = c(0,0,0,2): Adjust the spacing between the graphs
# mtext("Wright Map", side = 3, font = 2, line = 1): add a title
# screen(2): Start drawing on the second screen

split.screen(figs = matrix(c(0,.8,0,1,.8,1,0,1),ncol = 4, byrow = TRUE))
itemModern(thresholds, yRange = c(-3,4), show.axis.logits = "L", oma = c(0,0,0,2))
mtext("Wright Map", side = 3, font = 2, line = 1)
screen(2)
personHist(multi.proficiency, axis.persons = "",yRange = c(-3,4)
, axis.logits = "Persons", show.axis.logits = FALSE)

## Creating a multidimensional Wright Map with each dimension separate

## Mock results

d1 = rnorm(1000, mean = -0.5, sd = 1)
d2 = rnorm(1000, mean = 0.0, sd = 1)

dim1.diff <- rnorm(5)
dim2.diff <- rnorm(5)

split.screen(figs = matrix(c(0,.1,0,1,
.12,.6,0,1,
.5,.6,0,1,
.5,1,0,1),ncol = 4,byrow = TRUE))

personDens(d1,yRange = c(-3,3),show.axis.logits = FALSE
, axis.logits = "")
screen(2)
itemModern(dim1.diff,yRange = c(-3,3),show.axis.logits = FALSE)
mtext("Wright Map", side = 3, font = 2, line = 1)
screen(3)
personDens(d2,yRange = c(-3,3),show.axis.logits = FALSE
, axis.logits = ""
, axis.persons = "",dim.names = "Dim2")
screen(4)
itemModern(dim2.diff,yRange = c(-3,3),show.axis.logits = FALSE
, label.items = paste("Item",6:10))

```

Description

The plotCI function is intended for graphing confidence intervals. The difplot function is a wrapper for plotCI specifically intended for examining Differential Item Functioning from ConQuest output.

Usage

```
plotCI(ests, errors, labels = "", zeroline = TRUE, incol = "gray", outcol = "blue"
, main.title = "Statistical Significance Plot", axes = FALSE, xlab = "", pch = 16, ...)
## Default S3 method:
difplot(data, grouptype = NULL, group = NULL, item.names = NULL
, ylim = c(-1, 1), ylab = NULL, ...)
## S3 method for class 'CQmodel'
difplot(data, table.name = NULL, grouptype = NULL
, group = NULL, ...)
## S3 method for class 'character'
difplot(data, equation, ...)
```

Arguments

	plotCI parameters:
	vector of point estimates.
ests	vector of standard errors.
labels	vector of labels for the items.
zeroline	logical indicating whether to draw a line at zero.
incol	color of intervals containing zero.
outcol	color of intervals not containing zero.
main.title	title of the plot.
axes,xlab,pch	parameters passed to plot .
	difplot parameters:
data	A CQmodel object or the name of a ConQuest show file. Can also be a table of parameters taken from ConQuest output.
table.name	The RMP table to use for parameters. Should be an interactions table.
grouptype	The name of the demographic variable (e.g. "gender").
group	The name of the group to test for DIF (e.g. "male").
item.names	vector of item names.
equation	string specifying the model equation, if the Summary of Estimation table was not included in the show file.
ylim,ylab	more parameters passed to plot .
...	additional parameters to pass to plot.

Details

The `plotCI` function takes point estimates and standard errors as inputs and plots 95 percent confidence intervals in relation to a zero-line. By default, it colors the intervals gray if they include zero, and blue if they do not. The `difplot` function is a wrapper for `plotCI` specifically intended for examining Differential Item Functioning from ConQuest output and expects tables formatted exactly like ConQuest output to work correctly. For plotting DIF from other statistical packages, it is recommended to use `plotCI` directly.

Author(s)

David Torres Iribarra and Rebecca Freund

Examples

```
#Plotting confidence intervals

ests <- rnorm(10,sd = .5)
errors <- runif(10,min = .1,max = .5)
plotCI(ests,errors,ylim = c(-3,3))

#DIF plot:

fpath <- system.file("extdata", package="WrightMap")

# equation must be specified because there is no summary of estimation
# table in this example
difplot(file.path(fpath,"ex6a.shw"), equation = "item-gender+item*gender")
```

ppPlot

Person probability plots

Description

Plots a Wright Map for a single person (similar to a kidmap). On the person side, highlights their estimated ability and a range of one standard error. On the item side, draws lines representing item difficulties at which they are expected to have a 20%, 40%, 50%, 60%, and 80% chance of success.

Usage

```
ppPlot(thetas, thresholds, est, SE, main.title = "Person Probability Plot"
, cut.left = 0, cut.right = .94, cut.lab.adj = c(1,.5),...)

cutLines(cutpoints = NULL,cut.left = 0, cut.right = 1, cut.lab.text = NULL
, cut.lab.adj = c(0,1),...)
```

Arguments

thetas	a vector, matrix or data frame of person parameter estimates. Can also be a character string specifying a ConQuest output file of person parameter estimates, or a CQmodel object. Will be sent to the function personData .
thresholds	matrix or data frame of item parameter estimates. Can also be a character string specifying a ConQuest show file. Will be sent to the function itemData .
est	estimated ability of the person
SE	standard error of the estimate
main.title	title of the Person Probability Plot.
cut.left	value between 0 and 1 describing where to place the lefthand side of the cut-points, as a fraction of the item plot.
cut.right	value between 0 and 1 describing where to place the righthand side of the cut-points, as a fraction of the item plot.
cut.lab.adj	similar to the adj parameter in text , describes where to place the text for the cutpoints as a pair of values between 0 and 1 in terms of left-right and up-down alignment. Left-right alignment is 0 for the left side of the item plot and 1 for the right side, while up-down alignment is 0 for below the line and 1 for above the line.
cutpoints	argument to cutLines when called through wrightMap or one of the item.side functions. Specifies locations of cutlines. When cutLines is called through ppPlot, the cutpoints are calculated rather than specified.
cut.lab.text	argument to cutLines when called through wrightMap or one of the item.side functions. Specifies text to appear for each cut line. When cutLines is called through ppPlot, the text is always the percent chance of success given the estimated ability level and difficulty location.
...	additional arguments to pass to wrightMap or its associated functions.

Details

The ppPlot function is a wrapper for [wrightMap](#) that is specifically designed for person probability plots, and as such has access to all the parameters of [wrightMap](#) and its associated functions. It uses the `person.points`, `person.range`, `p.point.col`, and `p.range.col` parameters on the [person.side](#) function to draw a range of one standard error around the estimated ability level. On the item side, it calculates at what item difficulty the respondent is expected to have a 20%, 40%, 50%, 60%, and 80% chance of success and then uses the `cutLines` function to illustrate these cut-points. The `cutLines` function should not be called on its own and may be hidden in future versions. It is included here to show the available parameters, which can be included in a call to [wrightMap](#) or any of the [item.side](#) functions.

Author(s)

David Torres Iribarra and Rebecca Freund

See Also

[wrightMap](#)

Examples

```
fpath <- system.file("extdata", package="WrightMap")
model1 <- CQmodel(p.est = file.path(fpath,"ex2.eap"), show = file.path(fpath,"ex2.shw"))

#Person histogram, modern item
ppPlot(model1,est = 0, SE = 1)

#Person density, classic item
ppPlot(model1,est = 0, SE = 1, person.side = personDens,item.side = itemClassic)
```

WrightMap

Wright Map: IRT Item-Person Map

Description

This package allows the easy generation of ‘Wright Maps’ (named after Ben Wright), also known as item-person maps to display unidimensional and multidimensional assessment results. These maps represent simultaneously the proficiency distribution of respondents and the item difficulty parameters as estimated by a model of the Rasch Family. The package contains several other functions for graphing common IRT statistics.

Additionally, the package contains the CQmodel function, which reads output files created using ConQuest software and creates a set of data frames for easy data manipulation, bundled in a CQmodel object. The wrightMap function can take a CQmodel object as input or it can be used to create Wright Maps directly from data frames of person and item parameters.

Details

```
Package: WrightMap
Type: Package
Version: 1.0
Date: 2014-03-02
License: BSD_3_clause | LICENSE
```

The wrightMap function relies on two main inputs: (a) *thetas*: a vector or matrix of respondent proficiencies, and (b) *thresholds*: a vector or matrix of item thresholds. In the simplest case, say for a unidimensional Rasch model, *thetas* can be a vector of person proficiencies and *thresholds* a vector of item difficulties.

To plot multiple dimensions of person proficiency, simply provide them as a matrix where the results for each dimension is stored in a different column, such that for a 3-dimensional model with 1,000 persons, *theta* is a matrix of 1000 rows and 3 columns.

To plot polytomous items, the thresholds for each level must be passed to the functions through the *thresholds* matrix, where each row represents an item and each column represents a level.

For instance, if the results of a Rating Scale model with 5 response categories and 10 items is being plotted, the thresholds matrix will have 10 rows and 4 columns (column one represents the thresholds between the 1 and 2 response category, column 2 the threshold between categories 2 and 3, etc.).

Alternatively, wrightMap can read directly the .shw and .eap/.mle/.wle output files from a Conquest analysis, and will automatically generate the thetas and thresholds matrices.

Author(s)

David Torres Iribarra and Rebecca Freund

Maintainer: David Torres Iribarra <dti@berkeley.edu> and Rebecca Freund <rifreund@berkeley.edu>

References

Andrich, D. (1978). A rating formulation for ordered response categories. *Psychometrika*, 43(4), 561–573. Masters, G. N. (1982). A Rasch model for partial credit scoring. *Psychometrika*, 47(2), 149–174. Wilson, M. (2005). Constructing measures: An item response modeling approach. Wright, B. D. and Stone, M. H. (1979). *Best test design*. Chicago: Mesa Press.

Examples

```
# Plotting results of a unidimensional Rasch Model

## Mock results
uni.proficiency <- rnorm(1000, mean = -0.5, sd = 1)
difficulties <- sort( rnorm( 20))

## Default map
wrightMap( uni.proficiency, difficulties)

## Density version
wrightMap( uni.proficiency, difficulties, person.side = personDens)

# Plotting results of a multidimensional Rasch Model

## Mock results
multi.proficiency <- data.frame(
  d1 = rnorm(1000, mean = -0.5, sd = 1),
  d2 = rnorm(1000, mean = 0.0, sd = 1),
  d3 = rnorm(1000, mean = +0.5, sd = 1))

difficulties <- sort( rnorm( 20))

dev.new(width=10, height=10)
wrightMap( multi.proficiency, difficulties)

# Plotting results of a unidimensional Rating Scale Model

## Mock results
uni.proficiency <- rnorm(1000, mean = -0.5, sd = 1)
```

```
items.loc <- sort( rnorm( 20))
thresholds <- data.frame(
  l1 = items.loc - 0.5,
  l2 = items.loc - 0.25,
  l3 = items.loc + 0.25,
  l4 = items.loc + 0.5)

dev.new(width=10, height=10)
wrightMap( uni.proficiency, thresholds)

## Setting up labels, colors and symbols

thresholds.labels <- data.frame(
  l1 = paste('Lev',rep(1,20),sep = ''),
  l2 = paste('Lev',rep(2,20),sep = ''),
  l3 = paste('Lev',rep(3,20),sep = ''),
  l4 = paste('Lev',rep(4,20),sep = ''))

thresholds.colors <- data.frame(
  l1 = rep( 'green',20),
  l2 = rep(  'red',20),
  l3 = rep( 'yellow',20),
  l4 = rep(  'blue',20))

thresholds.symbols <- data.frame(
  l1 = rep( 15,20),
  l2 = rep( 16,20),
  l3 = rep( 17,20),
  l4 = rep( 18,20))

dev.new(width=10, height=10)
wrightMap( uni.proficiency, thresholds
, thr.lab.text = thresholds.labels
, thr.lab.col = thresholds.colors
, thr.sym.pch = thresholds.symbols
)
```

wrightMap

wrightMap: item-person maps

Description

This function allows the easy generation of ‘Wright Maps’ (named after Ben Wright), also known as item-person maps. They are used to display unidimensional and multidimensional assessment results. These maps represent simultaneously the proficiency distribution of respondents and the item difficulty parameters as estimated by a model of the Rasch family.

Usage

```
wrightMap(thetas, thresholds = NULL, item.side = itemModern, person.side = personHist
, main.title = "Wright Map", min.logit.pad = 0.25, max.logit.pad = 0.25, min.l = NULL
, max.l = NULL, item.prop = 0.8, return.thresholds = TRUE, new.quartz = FALSE
, use.hist = NULL,...)
## S3 method for class 'CQmodel'
plot(x, ...)
```

Arguments

The parameters documented here do not include many of the options included in the Wright Map family of functions. For graphical parameters, see [item.side](#) and [person.side](#). For data handling, see [item.person.data](#) and [CQmodel](#).
wrightMap parameters:

	a vector, matrix or data frame of person parameter estimates. Can also be a character string specifying a ConQuest output file of person parameter estimates, or a CQmodel object. Will be sent to the function personData .
<code>thresholds</code>	matrix or data frame of item parameter estimates. Can also be a character string specifying a ConQuest show file. Will be sent to the function itemData .
<code>item.side</code>	function to use to draw the item side of the map. Currently included options are <code>itemModern</code> (default), <code>itemClassic</code> (for ConQuest-style Wright Maps) and <code>itemHist</code> . See item.side for details.
<code>person.side</code>	function to use to draw the person side of the map. Currently included options are <code>personHist</code> (default), to draw the person distribution as a histogram, and <code>personDens</code> , which draws a density plot. See person.side for details.
<code>main.title</code>	title of the Wright Map.
<code>min.logit.pad</code>	numeric value indicating how much of the lower end of the logit scale should be included in the plot.
<code>max.logit.pad</code>	numeric value indicating how much of the upper end of the logit scale should be included in the plot.
<code>min.l</code>	numeric value for fixing the lower end of the logit scale. It overrides the automatic detection of the range and the <code>min.logit.pad</code> correction.
<code>max.l</code>	numeric value for fixing the upper end of the logit scale. It overrides the automatic detection of the range and the <code>max.logit.pad</code> correction.
<code>item.prop</code>	numeric value greater than 0 and smaller than 1 indicating the proportion of the plot to be allocated to the item part of the Wright Map.
<code>return.thresholds</code>	logical. Determines whether to return or not the numeric values used to position the parameters on the item side of the Wright Map. Enabled by default.
<code>new.quartz</code>	logical. Determines whether the wrightMap will be created on a new graphical device or if it will reuse one already open. By default is set to FALSE to avoid creating new devices.
<code>use.hist</code>	deprecated. Use the person.side parameter instead
<code>...</code>	Additional arguments to pass to personData , itemData , person.side , or item.side
	wrightMap can also be called by passing a CQmodel object to plot:
<code>x</code>	CQmodel object to pass to plot

Author(s)

David Torres Iribarra and Rebecca Freund

References

Wilson, M. (2005). *Constructing measures: An item response modeling approach*. Wright, B. D. and Stone, M. H. (1979). *Best test design*. Chicago: Mesa Press.

See Also

[person.side](#) [item.side](#) [personData](#) [itemData](#)

Examples

```
# Plotting results of a unidimensional Rasch Model

## Mock results
uni.proficiency <- rnorm(1000, mean = -0.5, sd = 1)
difficulties <- sort( rnorm( 20))

## Default map
wrightMap( uni.proficiency, difficulties)

## Density version
wrightMap( uni.proficiency, difficulties, person.side = personDens)

# Plotting results of a multidimensional Rasch Model

## Mock results
multi.proficiency <- data.frame(
  d1 = rnorm(1000, mean = -0.5, sd = 1),
  d2 = rnorm(1000, mean = 0.0, sd = 1),
  d3 = rnorm(1000, mean = +0.5, sd = 1))

difficulties <- sort( rnorm( 20))

dev.new(width=10, height=10)
wrightMap( multi.proficiency, difficulties)

# Plotting results of a unidimensional Rating Scale Model

## Mock results
uni.proficiency <- rnorm(1000, mean = -0.5, sd = 1)

items.loc <- sort( rnorm( 20))
thresholds <- data.frame(
  l1 = items.loc - 0.5 ,
  l2 = items.loc - 0.25,
  l3 = items.loc + 0.25,
  l4 = items.loc + 0.5)
```

```
wrightMap( uni.proficiency, thresholds)

#####ConQuest integration###

fpath <- system.file("extdata", package="WrightMap")

#Partial credit model:

model1 <- CQmodel(p.est = file.path(fpath,"ex2.eap"), show = file.path(fpath,"ex2.shw"))
wrightMap(model1)

# Rating scale model:
model2 <- CQmodel(file.path(fpath,"ex2b.eap"), file.path(fpath,"ex2b-2.shw"))
wrightMap(model2, label.items.row = 2)

# Complex model
model3 <- CQmodel(file.path(fpath,"ex4a.mle"), file.path(fpath,"ex4a.shw"))
wrightMap(model3, min.logit.pad = -29, person.side = personDens)

### Skip CQmodel
wrightMap(file.path(fpath,"ex2a.eap"), file.path(fpath,"ex2a.shw"),
  label.items.row = 3)
```

Index

- * **IRT**
 - WrightMap, 30
- * **Item Response Models**
 - WrightMap, 30
- * **Rasch Model**
 - WrightMap, 30
- * **Wright Map**
 - WrightMap, 30
- * **hplot**
 - fitgraph, 9
- CCCfit, 2, 7, 8, 11
- CQmodel, 4, 21, 23, 33
- cutLines, 16
- cutLines (ppPlot), 28
- difplot (plotCI), 26
- extract.deltas, 7
- fitgraph, 9
- hist, 25
- ICCfit, 3, 10
- item.person.data, 13, 33
- item.side, 15, 15, 25, 29, 33, 34
- itemClassic (item.side), 15
- itemData, 17, 18, 23, 29, 33, 34
- itemData (item.person.data), 13
- itemHist (item.side), 15
- itemModern (item.side), 15
- kidmap (ppPlot), 28
- layout, 17
- make.deltas, 15, 20, 23
- make.thresholds, 8, 14, 15, 21, 22
- par, 16, 25
- person.side, 15, 17, 18, 24, 29, 33, 34
- personData, 25, 29, 33, 34
- personData (item.person.data), 13
- personDens (person.side), 24
- personHist (person.side), 24
- plot, 27
- plot.CQmodel (wrightMap), 32
- plotCI, 26
- ppPlot, 25, 28
- print.CQmodel (CQmodel), 4
- print.SOE (CQmodel), 4
- split.screen, 17
- text, 29
- WrightMap, 30
- wrightMap, 13–15, 17, 18, 21, 23–25, 29, 32
- WrightMap-package (WrightMap), 30