Package 'CCTpack'

October 10, 2017

Type Package

Title Consensus Analysis, Model-Based Clustering, and Cultural Consensus Theory Applications

Version 1.5.2

Date 2017-10-10

Author Royce Anders <andersr@uci.edu>

Maintainer Royce Anders <andersr@uci.edu>

Description

Consensus analysis, model-based clustering, and cultural consensus theory applications to response data (e.g. questionnaires). The models are applied using hierarchical Bayesian inference. The current package version supports binary, ordinal, and continuous data formats.

Depends R2jags (>= 0.04-03)

Imports tcltk, rjags, psych, mvtnorm, polycor, MASS, methods

License GPL (>= 2)

NeedsCompilation no

Repository CRAN

Date/Publication 2017-10-10 11:09:58 UTC

R topics documented:

`CCTpack-package` *CCTpack: Consensus Analysis, Model-Based Clustering, and Cultural Consensus Theory Applications*

Description

An R package for advanced model-based analyses of questionnaire data. These models can detect the consensus answers of the respondents, and perform a model-based clustering of the respondents. The methods can detect latent subgroups in the data, their differing consensuses, the expertise of each respondent, their response biases, and the difficulty of each question. The ability to parse the variance of the data, by these factors, results in sophisticated analyses of the consensus answers, the clusters, and the respondent knowledge/expertise. These are advanced models that are fit using hierarchical Bayesian inferential methods. Bayesian sampling routines are currently handled with JAGS (please install beforehand: mcmc-jags.sourceforge.net/). The package can currently handle data from questionnaires of binary responses, ordered categorical, or continuous responses. If not already in this format, your data might be appropriately transformed to one of these scales to be fit by these models. The models are based on mathematical publications in the domain of Cultural Consensus Theory (CCT): the General Condorcet Model (GCM), the Latent Truth Rater Model (LTRM), and the Continuous Response Model (CRM). See the relevant literature listed in the reference manual for more information about these CCT models and methods. Respectively, these models are applicable to dichotomous/binary (0,1), ordinal (1, 2, ...), and continuous data. In addition, there is functionality for component analyses/scree plots of the data, automatic plots of the model results, diagnostics for quality of fit, and exporting these plots/results.

Details

Package:	CCTpack
Type:	Package
Version:	1.5.2

Date: 2017-10-10
License: GPL (>= 2)

1. Make sure you have JAGS installed
2. To install CCTpack use command: install.packages("CCTpack",dependencies=TRUE)
3. To load CCTpack use command: library(CCTpack)
4. Type ?cctapply to get a walkthrough with example data 5. For more information on the models and methods used, see the References section.

Author(s)

Royce Anders

Maintainer: Royce Anders <andersr@uci.edu>

References

Anders, R., Oravecz, Z., & Batchelder, W. H. (2014). Cultural consensus theory for continuous responses: A latent appraisal model for information pooling. Journal of Mathematical Psychology, 61, 1-13.

Anders, R., & Batchelder, W. H. (2015). Cultural consensus theory for the ordinal data case. Psychometrika, 1-31.

Anders, R., & Batchelder, W. H. (2012). Cultural consensus theory for multiple consensus truths. Journal for Mathematical Psychology, 56, 452-469.

Batchelder, W. H., & Anders, R. (2012). Cultural consensus theory: comparing different concepts of cultural truth. Journal of Mathematical Psychology, 56, 316-332.

Oravecz, Z., Anders, R. & Batchelder, W. H. (2015). Test theory without an answer key in a Bayesian hierarchical modeling framework. Psychometrika, 1-24.

See Also

To install JAGS, see: mcmc-jags.sourceforge.net/

`cct-class` *Class* `"cct"`

Description

Virtual class that contains "rjags" class, used to create plot, summary, and screeplot methods for the cctfit objects from package CCTpack.

Objects from the Class

Objects can be created by calls of the form `new("cct", ...)`.

Slots

`model`: Object of class `"jags"` ~~

`BUGSoutput`: Object of class `"bugs"` ~~

Extends

Class `"rjags"`, directly.

Methods

plot `signature(x = "cct", y = "rjags")`: ...

Author(s)

Royce Anders Maintainer: Royce Anders <andersr@uci.edu>

Examples

```
showClass("cct")
```

---

cctapply

Primary function to perform model-based consensus analysis: loads the data and fits the consensus model. Allows for model-based clustering based on the numbers of "clusters" specified. Options are also available to run diagnostics on the fit, and to export the results to saved files. Based on cultural consensus theory (CCT) models for data matrices of binary, ordered categorical, or continuous response data.

Description

loads the data, fits the appropriate model, runs the posterior predictive checks, and optionally exports results

Usage

```
cctapply(data, clusters = 1, itemdiff = FALSE, biases = TRUE, samples = 10000, chains = 3,
         burnin = 2000, thinning = 1, runchecks = FALSE, exportfilename = "",
         polych = FALSE, parallel = FALSE, seed = NULL,plotr=FALSE)
```

Arguments

data	a 2-dimensional matrix or array, missing values should be input as NA.
clusters	The number of possible clusters (cultures) to use
itemdiff	Whether item difficulty should be measured and accounted for
biases	Whether response biases should be measured and accounted for
samples	The number of samples for the inference
chains	The number of chains for the inference
burnin	The number of burn-in for the inference
thinning	The amount of thinning in the inference
runchecks	If the posterior predictive checks should be calculated after the inference
exportfilename	If you'd like to export: specify a filename and optionally its location. Ex: `exportfilename = "C:/CCTpack/CCTpackdata.Rdata"`
polych	used for ordinal data only, if the polychoric correlations, rather than Pearson correlations, should be used (for the posterior predictive checks) – these take a long time to calculate but are more precise in the ordinal data case.
parallel	Whether the inference should be computed in parallel (1 chain per logical processor)
seed	Set the random number seed here (to reproduce results as before). If not specified, the seed is randomly generated.
plotr	Whether to plot the posterior mean results for each parameter. Note: `runchecks = TRUE` will plot the posterior predictive checks after the posterior mean results. The posterior mean results plot can be later called via `cctresults()`.

Details

This is the main function to fit the consensus models. The function fits the model using hierarchical Bayesian inference. The Bayesian sampling is performed using JAGS.

Value

`cctfit` is returned, which has the structure of a 'jagsfit' object as in Rjags, but has additional data included.

Examples

```
# Load Data (here binary responses, 1/0 for yes/no)
data(hotcold)

# Calculate scree plot to decide how many clusters to run, looks like 2 clusters here
dat <-  cctscree(hotcold)

# Retrieve factors from the scree plot
cctfac(dat)    # dat$factors

# Fit the Model
```

```
# cctfit <- cctapply(data = hotcold, clusters = 2, itemdiff = TRUE, samples = 10000,
#                     chains = 3, burnin = 2000, seed = 1, runchecks = FALSE)

# Calculate Fit Diagnostics (Posterior Predictive Checks)
# cctfit <- cctppc(cctfit)

# Plot Parameter Results
# cctresults(cctfit)

# Tables of Subject and Item Parameter Values and Credible Intervals
# cctsubj(cctfit)      # cctfit$subj
# cctsubjhdi(cctfit)   # cctfit$subjhdi
# cctitem(cctfit)      # cctfit$item
# cctitemhdi(cctfit)   # cctfit$itemhdi

# Show Missing Value Model Estimates if there was missing data
# cctmvest(cctfit)

# Export Results (saves data and plots)
# cctexport(cctfit,filename="CCTpackdata.Rdata")

# Load and Fit Example Data for ordered categorical or continuous responses
# data(raterdata)

# cctfit <- cctapply(data = raterdata, clusters = 1, itemdiff = TRUE, samples = 10000,
#                     chains = 3, burnin = 2000, seed = 1, runchecks = FALSE)

# data(continuousdata)
# cctfit <- cctapply(data = continuousdata, clusters = 1, itemdiff = TRUE, samples = 10000,
#                     chains = 3, burnin = 2000, seed = 1, runchecks = FALSE)

##Note: if an insufficient memory message occurs, you can increase the
##memory allocation by the command 'memory.limit(25000)'  (or as high as needed)
```

`cctcat` *Accessor function for the model-estimated category boundaries (obtained from the model applied to the data). Applicable for the LTRM model only.*

Description

Outputs a table read out of the category boundary parameters of the model inference, as well as their credible intervals (posterior highest density intervals, HDIs), for each cluster.

Usage

```
cctcat(cctfit)
```

Arguments

`cctfit`	The `cctfit` object as obtained from the `cctapply()` function.

Examples

```
data(raterdata)
# cctfit <- cctapply(data = raterdata, clusters = 1, itemdiff = TRUE, samples = 10000,
#                    chains = 3, burnin = 2000, runchecks = FALSE)
# cctcat(cctfit)
```

---

cctexport *Export the results (the cctfit object)*

Description

Exports the `cctfit` object as an .Rdata file, as well as .jpeg and .eps files of the relevant plots

Usage

```
cctexport(cctfit, filename = "CCTpackdata.Rdata")
```

Arguments

`cctfit`	The `cctfit` object as obtained from the `cctapply()` function.
`filename`	The filename and location you would like to use. If no location is specified, it is saved to the current R working directory (see `getwd()`).

Details

Saves the `cctfit` object as an .Rdata file, as well as .jpeg and .eps files of the relevant plots, which include: the scree plot, results plot, and posterior predictive check plots.

Examples

```
data(hotcold)
# cctfit <- cctapply(data = hotcold, clusters = 2, itemdiff = TRUE, samples = 10000,
#                    chains = 3, burnin = 2000, runchecks = TRUE)
# cctexport(cctfit, filename = "C:/CCTpack/CCTpackdata.Rdata")
```

---

cctfac *Accessor function for the factor coefficients after* `dat <- cctscree(dat)` *has been run.*

Description

Outputs a vector of the factor coefficients obtained from a principal components analysis of the respondent by respondent correlation matrix of the data.

Usage

```
cctfac(dat)
```

Arguments

dat	The object as obtained from the `dat <- cctscree(dat)` function.

Examples

```
data(hotcold)
dat <- cctscree(hotcold)
cctfac(dat)
```

---

cctgui *The CCT GUI*

Description

There is a graphical user interfance available, though it is deprecated / being phased out – it is not compatible with all operating systems.

Instructions how to use the GUI: 1. Type 'cctgui()' in the R command prompt to start the CCT GUI
2. Click "Load data" (data must be in csv format as a .csv or .txt file)
data format: respondents (rows) by items (columns), csv format, no row/column names necessary
the models can handle missing data, missing data should be input as NA
3. Verify that the number of respondents, items, and data type the GUI detects is correct.
4. Check the number of significant factors in the scree plot
5. Use this as "the number of cultures to assume," and select an item difficulty option
6. Click "Apply CCT Model"
7. Check inference results (if the number of Rhats is appropriate);
if not, one could try running more samples, or a different number of cultures
8. Click "Run Checks" to run the posterior predictive checks, and verify if they are satisfied.
Note: the CCT method suggests that the fewest number of cultures to pass the checks should be the model used.
9. Click "Plot Results" to see the posterior results
10. Click "Export Results" to save the model fit and plots.

Usage

```
cctgui()
```

Value

All information will be saved in the 'cctfit' object. This is in the same structure as the 'jagsfit' object returned by the function jags(), which utilizes the 'rjags' and 'R2jags' packages.

Author(s)

Royce Anders

Examples

```
#First visit  'mcmc-jags.sourceforge.net/' to install JAGS (Plummer, 2003) if it's not installed

####################
#Instructions for Using the GUI
####################

#1) Convert your data into a .csv file
### Or use our example data such as with the following commands:

data(hotcold);
write.csv(x=hotcold, file="hotcold.csv",row.names=FALSE)

#2) Invoke the GUI

cctgui()

#Click "Load Data" then find and select "hotcold.csv"
#Note that 23 respondents, 27 items are detected,
#      and that it is Dichotomous (binary) data, which the GCM is applicable for
#      the GUI detects that there are 14 missing data values in the matrix
#Click "Scree Plot"  Note that there are 2 apparent significant factors,
#      thus we assume 2 cultures in the data, click "yes" to estimate item difficulty
#Click "Apply CCT Model" and wait for the inference to finish
#Take note of the Number of Rhats above 1.1 (if too many, perhaps run more samples)
#Click "Run Checks" to run the posterior predictive checks, wait for the checks to complete
#Note that the model satisfies both checks in the plots shown
#Proceed to inspect the inference results by clicking "Plot Results"
#type 'cctfit$MVest' to view the model estimates of the 14 missing data values
#Click "Export Results" to save the plot and the inference results
#The Inference results are included in the object 'cctfit'
#Type 'cctfit' in the R prompt to see the summary,
#   type 'str(cctfit)' to see what the object contains

####################
#2) Instructions for Using the Command Prompt
####################

#The corollary to the GUI instructions above, for the command line, is below

data(hotcold);   #for an ordinal data example, use data(raterdata)

#Loads data and provides the Scree Plot
# cctscree(data = hotcold)

#Loads data and Runs the Inference
# cctfit <- cctapply(data = hotcold,clusters=2,itemdiff=TRUE,samples=10000,
#                     chains=3,burnin=2000,runchecks=FALSE)
```

```
#Calculates and Plots Posterior Predictive Checks
# cctfit <- cctppc(cctfit)

#Show Missing Value Model Estimates if there was missing data
# cctfit$MVest

#Plots Posterior Results
# cctresults(cctfit)

#Exports Results
# cctexport(cctfit,filename="CCTpackdata.Rdata")

##Note: if an insufficient memory message occurs, you can increase the
##memory allocation by the command 'memory.limit(25000)'  (or as high as needed)
```

`cctitem` *Accessor function for the item parameters (obtained from the model applied to the data)*

Description

Outputs a table read out of the item parameters of the model inference (for example: item #, estimated consensus answer for each cluster, and item difficulty for each cluster)

Usage

```
cctitem(cctfit)
```

Arguments

`cctfit`	The `cctfit` object as obtained from the `cctapply()` function.

Examples

```
data(hotcold)
# cctfit <- cctapply(data = hotcold, clusters = 2, itemdiff = TRUE, samples = 10000,
#                    chains = 3, burnin = 2000, runchecks = FALSE)
# cctitem(cctfit)
```

cctitemhdi *Accessor function for the item parameter credible intervals (obtained from the model applied to the data)*

Description

Outputs a table read out of the credible intervals (posterior highest density intervals, HDIs) of the item parameters of the model inference (for example, the lower and upper bounds of estimated consensus answers and item difficulties for each cluster)

Usage

```
cctitemhdi(cctfit)
```

Arguments

cctfit	The cctfit object as obtained from the cctapply() function.

Examples

```
data(hotcold)
# cctfit <- cctapply(data = hotcold, clusters = 2, itemdiff = TRUE, samples = 10000,
#                     chains = 3, burnin = 2000, runchecks = FALSE)
# cctitemhdi(cctfit)
```

cctmemb *Accessor function for the cluster memberships of the respondents.*

Description

Outputs the cluster (cultural) assignment of each respondent, as determined by the CCT model fit to the data .

Usage

```
cctmemb(cctfit)
```

Arguments

cctfit	The cctfit object as obtained from the cctapply() function.

Examples

```
data(hotcold)
# cctfit <- cctapply(data = hotcold, clusters = 2, itemdiff = TRUE, samples = 10000,
#                     chains = 3, burnin = 2000, runchecks = FALSE)
# cctmemb(cctfit)
```

cctmvest *Accessor function for the missing value estimates of the data, based on the cctfit (if there were missing values in the data).*

Description

Outputs an N by 3 matrix, where N is the number of missing values estimated by the model, column 1 is the person index, column 2 the item index, column 3 is the value estimate.

Usage

```
cctmvest(cctfit)
```

Arguments

cctfit	The cctfit object as obtained from the cctapply() function.

Examples

```
data(hotcold)
# cctfit <- cctapply(data = hotcold, clusters = 2, itemdiff = TRUE, samples = 10000,
#                      chains = 3, burnin = 2000, runchecks = FALSE)
# cctmvest(cctfit)
```

cctppc *Calculate or Plot the Posterior Predictive Model Checks*

Description

Plots (and calculates if not calculated already), the posterior predictive model checks for the cctfit object

Usage

```
cctppc(cctfit, polych = FALSE, doplot = TRUE)
```

Arguments

cctfit	The cctfit object as obtained from the cctapply() function.
polych	used for ordinal data only, if the polychoric correlations, rather than Pearson correlations, should be used (for the posterior predictive checks) – these take a long time to calculate but are more precise in the ordinal data case.
doplot	If the diagnostics should be plotted.

Details

Generates 500 posterior predictive data sets that are randomly sampled from the posterior predictive data; it uses these to calculate 2 posterior predictive checks that respectively pertain to fitting the consensus structure of the data (the number of latent cultures), and if heterogeneous item difficulty should be used.

Value

returns the cctfit object with the posterior predictive data and checks saved.

Examples

```
data(hotcold)
# cctfit <- cctapply(data = hotcold, clusters = 2, itemdiff = TRUE, samples = 10000,
#                     chains = 3, burnin = 2000, runchecks = FALSE)
# cctfit <- cctppc(cctfit)
```

---

cctresults *Plot the posterior results from the model inference*

Description

Plot the posterior results from the model inference, a specialized display is produced depending on the model that is applied.

Usage

```
cctresults(cctfit)
```

Arguments

`cctfit`	The `cctfit` object as obtained from the `cctapply()` function.

Examples

```
data(hotcold)
# cctfit <- cctapply(data = hotcold, clusters = 2, itemdiff = TRUE, samples = 10000,
#                     chains = 3, burnin = 2000, runchecks = FALSE)
# cctresults(cctfit)
```

cctscree *Generate a scree plot*

Description

Creates a screeplot of the data, providing the first 8 factors and their magnitudes.

Usage

```
cctscree(data, polych = FALSE)
```

Arguments

data	a 2-dimensional matrix or array, missing values should be input as NA.
polych	used for ordinal data only, if the polychoric correlations, rather than Pearson correlations, should be used

Details

The scree plot is generated from the respondent by respondent correlation (Pearson) matrix.

Examples

```
data(hotcold)
cctscree(data = hotcold)
## With this example data, a scree plot with 2 substantial factors
##              (suggesting two cultures) is produced.
```

cctsubj *Accessor function for the subject parameters (obtained from the model applied to the data)*

Description

Outputs a table read out of the subject parameters of the model inference (for example: participant #, cluster membership, competency, response biases)

Usage

```
cctsubj(cctfit)
```

Arguments

cctfit	The `cctfit` object as obtained from the `cctapply()` function.

Examples

```
data(hotcold)
# cctfit <- cctapply(data = hotcold, clusters = 2, itemdiff = TRUE, samples = 10000,
#                    chains = 3, burnin = 2000, runchecks = FALSE)
# cctsubj(cctfit)
```

`cctsubjhdi` *Accessor function for the subject parameter credible intervals (obtained from the model applied to the data)*

Description

Outputs a table read out of the credible intervals (posterior highest density intervals, HDIs) of the subject parameters of the model inference (for example, the lower and upper bounds of estimated participant knowledge competencies and response biases)

Usage

```
cctsubjhdi(cctfit)
```

Arguments

`cctfit`	The `cctfit` object as obtained from the `cctapply()` function.

Examples

```
data(hotcold)
# cctfit <- cctapply(data = hotcold, clusters = 2, itemdiff = TRUE, samples = 10000,
#                    chains = 3, burnin = 2000, runchecks = FALSE)
# cctsubjhdi(cctfit)
```

`continuousdata.csv` *continuousdata*

Description

Example 2 culture continuous data with item difficulty for cctgui()
This data should load as 40 respondents by 40 items, and as continuous data
It is an example of 2 culture data, with 16 missing data values

Note

csv or text data files need not use header or row names
Though respondents should be by the rows, and items by the columns

dtraceplot *Traceplots for discrete parameters of a cctfit object*

Description

Produces all of the traceplots the discrete parameters of a cctfit object, in a 3x3 design and multiple plot windows, via traceplot from **R2jags**.

Usage

```
dtraceplot(cctfit,ask = FALSE)
```

Arguments

cctfit	The cctfit object as obtained from the cctapply() function.
ask	logical; if TRUE, the user is asked before each plot, to proceed to the next. See par(ask=.)

Examples

```
data(hotcold)
# cctfit <- cctapply(data = hotcold, clusters = 2, itemdiff = TRUE, samples = 10000,
#                      chains = 3, burnin = 2000, runchecks = FALSE)
# dtraceplot(cctfit)
```

hotcold.csv *hotcold*

Description

Example 2 culture binary data with item difficulty for cctgui()
This data should load as 23 respondents by 27 items, and as binary data
It is an example of 2 culture data, with 14 missing data values

Note

csv or text data files need not use header or row names
Though respondents should be by the rows, and items by the columns

`plot` *Plot method for a cctfit object of class 'cct'; equivalent to function cctresults().*

Description

Plots the posterior results from the model inference, a specialized display is produced depending on the model that is applied.

Usage

```
plot(x,y,...)
```

Arguments

`x`	is a `cctfit` object as obtained from the `cctapply()` function, which has class 'cct'.
`y`	NULL
`...`	Additional arguments

Examples

```
data(hotcold)
# cctfit <- cctapply(data = hotcold, clusters = 2, itemdiff = TRUE, samples = 10000,
#                    chains = 3, burnin = 2000, runchecks = FALSE)
# plot(cctfit)
```

`plot-methods` *~~ Methods for Function* `plot` *in Package* **graphics** *~~*

Description

~~ Methods for function `plot` in package **graphics** ~~

Methods

`signature(x = "ANY")`	Default plot method.
`signature(x = "cct")`	CCTpack plot method for cctfit objects from cctapply.

raterdata.csv *raterdata*

Description

Example 1 culture ordinal data with item difficulty for cctgui()
This data should load as 25 respondents by 40 items, and as ordinal data
It is an example of 1 culture data, with 9 missing data values

Note

csv or text data files need not use header or row names
Though respondents should be by the rows, and items by the columns

screeplot *Generate a scree plot*

Description

Screeplot method for CCTpack data, or a cctfit object of class 'cct'; equivalent to function cctscree().

Usage

```
screeplot(x, ...)
```

Arguments

x	a 2-dimensional matrix or dataframe, missing values should be input as NA, or a cctfit object of class 'cct'.
...	Additional arguments: polych=T (for ordinal data only), to use the polychoric correlations rather than Pearson correlations.

Details

The scree plot is generated from the respondent by respondent correlation (Pearson) matrix.

Examples

```
data(hotcold)
screeplot(hotcold)
## With this example data, a scree plot with 2 substantial factors
##              (suggesting two cultures) is produced.
```

screeplot-methods *~~ Methods for Function* screeplot *in Package* **stats** *~~*

Description

~~ Methods for function screeplot in package **stats** ~~

Methods

signature(x = "ANY") A matrix or data.frame.

signature(x = "cct") An cctfit object from cctapply.

signature(x = "data.frame")

signature(x = "matrix")

summary *Summary of the cctfit object of class 'cct'.*

Description

Summary method for a cctfit object of class 'cct'. Information about the data, the fit, posterior predictive checks.

Usage

```
summary(object, ...)
```

Arguments

object	The cctfit object as obtained from the cctapply() function, which has class 'cct'.
...	Additional arguments

Examples

```
data(hotcold)
# cctfit <- cctapply(data = hotcold, clusters = 2, itemdiff = TRUE, samples = 10000,
#                     chains = 3, burnin = 2000, runchecks = FALSE)
# summary(cctfit)
```

summary-methods ~~ *Methods for Function* summary *in Package* **base** ~~

Description

~~ Methods for function summary in package **base** ~~

Methods

signature(object = "ANY") Default method.

signature(object = "cct") CCTpack plot method for cctfit objects from cctapply.

---

testdat.csv *testdat*

Description

Example test data for cctgui()
This data should load as 20 respondents by 25 items, and as binary data
It is an example of 1 culture data

Note

csv or text data files need not use header or row names
Though respondents should be by the rows, and items by the columns

Index