Package 'airt'

October 12, 2022

Type Package

Title Evaluation of Algorithm Collections Using Item Response Theory

Version 0.2.1

Maintainer Sevvandi Kandanaarachchi <sevvandik@gmail.com>

Description An evaluation framework for algorithm portfolios using Item Response Theory (IRT). We use continuous and polytomous IRT models to evaluate algorithms and introduce algorithm characteristics such as stability, effectiveness and anomalousness (Kandanaarachchi, Smith-Miles 2020) <doi:10.13140/RG.2.2.11363.09760>.

License GPL-3

Encoding UTF-8

LazyData true

RoxygenNote 7.1.1

Depends R (>= 3.4.0)

Imports pracma, mirt, tidyr, EstCRM, rlang, dplyr, magrittr, tibble

Suggests knitr, rmarkdown, ggplot2, gridExtra, scales

VignetteBuilder knitr

URL https://sevvandi.github.io/airt/

NeedsCompilation no

Author Sevvandi Kandanaarachchi [aut, cre] (<https://orcid.org/0000-0002-0337-0395>)

Repository CRAN

Date/Publication 2021-12-17 05:10:02 UTC

R topics documented:

algo_effectiveness_crm 2
algo_effectiveness_poly 3
cirtmodel 4
classification_cts 5

classification_poly 5
effectiveness_crm 6
effectiveness_poly 7
latent_trait_analysis 8
make_polyIRT_data 9
model_goodness_crm 10
model_goodness_for_algo_crm 10
model_goodness_for_algo_poly 11
model_goodness_poly 12
pirtmodel 13
prepare_for_plots_crm 14
prepare_for_plots_poly 14

Index **16**

`algo_effectiveness_crm` *Computes the actual and predicted effectiveness of a given algorithm.*

Description

This function computes the actual and predicted effectiveness of a given algorithm for different tolerance values.

Usage

```
algo_effectiveness_crm(mod, num = 1)
```

Arguments

`mod`	A fitted `mirt` model using the function `irtmodel` or `R` package `mirt`.
`num`	The algorithm number, for which the goodness of the IRT model is computed.

Value

A list with the following components:

`effective`	The `x,y` coodinates for the actual and predicted effectiveness curves for algorithm `num`.
`predictedEff`	The area under the predicted effectiveness curve.
`actualEff`	The area under the actual effectiveness curve.

Examples

```
set.seed(1)
x1 <- runif(100)
x2 <- runif(100)
x3 <- runif(100)
X <- cbind.data.frame(x1, x2, x3)
max_item <- rep(1,3)
min_item <- rep(0,3)
mod <- cirtmodel(X, max.item=max_item, min.item=min_item)
out <- algo_effectiveness_crm(mod$model, num=1)
out
```

`algo_effectiveness_poly` *Computes the actual and predicted effectiveness of a given algorithm.*

Description

This function computes the actual and predicted effectiveness of a given algorithm for different tolerance values.

Usage

```
algo_effectiveness_poly(mod, num = 1)
```

Arguments

`mod`	A fitted `mirt` model using the function `irtmodel` or `R` package `mirt`.
`num`	The algorithm number

Value

A list with the following components:

`effective`	The `x,y` coodinates for the actual and predicted effectiveness curves for algorithm `num`.
`predictedEff`	The area under the predicted effectiveness curve.
`actualEff`	The area under the actual effectiveness curve.

#'@examples set.seed(1) x1 <- sample(1:5, 100, replace = TRUE) x2 <- sample(1:5, 100, replace = TRUE) x3 <- sample(1:5, 100, replace = TRUE) X <- cbind.data.frame(x1, x2, x3) mod <- pirtmodel(X) out <- algo_effectiveness_poly(mod$model, num=1) out

`cirtmodel` *Fits a continuous IRT model.*

Description

This function fits a continuous Item Response Theory (IRT) model to the algorithm performance data. The function EstCRMitem in the R package EstCRM is updated to accommodate negative discrimination.

Usage

```
cirtmodel(df, max.item = NULL, min.item = NULL)
```

Arguments

`df`	The performance data in a matrix or dataframe.
`max.item`	A vector with the maximum performance value for each algorithm.
`min.item`	A vector with the minimum performance value for each algorithm.

Value

A list with the following components:

`model`	The IRT model.
`anomalous`	A binary value for each algorithm. It is set to 1 if an algorithm is anomalous. Otherwise it is set to 0.
`stability`	The stability of each algorithm.
`easiness_threshold`	The easiness threshold of each algorithm. A lower threshold indicates that the algorithm finds more test instances easy.

Examples

```
set.seed(1)
x1 <- runif(100)
x2 <- runif(100)
x3 <- runif(100)
X <- cbind.data.frame(x1, x2, x3)
max_item <- rep(1,3)
min_item <- rep(0,3)
mod <- cirtmodel(X, max.item=max_item, min.item=min_item)
```

`classification_cts` *A dataset containing classification algorithm performance data in a continuous format.*

Description

This dataset contains the performance of 10 classification algorithms on 235 datasets discussed in the paper Instance Spaces for Machine Learning Classification by M. A. Munoz, L. Villanova, D. Baatar, and K. A. Smith-Miles .

Usage

```
classification_cts
```

Format

A dataframe of 235 x 10 dimensions.

Dimension 1 Each row contains the algorithm performance of a dataset on 10 classification algorithms.

Dimensions 2 Each column contains the algorithm performance of a single algorithm.

Source

https://katesmithmiles.wixsite.com/home/matilda

`classification_poly` *A dataset containing classification algorithm performance data in a polytomous format.*

Description

This dataset contains the performance of 10 classification algorithms on 235 datasets discussed in the paper Instance Spaces for Machine Learning Classification by M. A. Munoz, L. Villanova, D. Baatar, and K. A. Smith-Miles .

Usage

```
classification_poly
```

Format

A dataframe of 235 x 10 dimensions.

Dimension 1 Each row contains the algorithm performance of a dataset on 10 classification algorithms.

Dimensions 2 Each column contains the algorithm performance of a single algorithm.

Source

https://katesmithmiles.wixsite.com/home/matilda

`effectiveness_crm` *Computes the actual and predicted effectiveness of the collection of algorithms.*

Description

This function computes the actual and predicted effectiveness of the collection of algorithms for different tolerance values.

Usage

```
effectiveness_crm(mod)
```

Arguments

`mod`	A fitted `mirt` model using the function `irtmodel` or `R` package `mirt`.

Value

A list with the following components:

`effectivenessAUC`
: The area under the actual and predicted effectiveness curves.

`actcurves`
: The `x,y` coodinates for the actual effectiveness curves for each algorithm.

#'

`prdcurves`
: The `x,y` coodinates for the predicted effectiveness curves for each algorithm.

Examples

```
set.seed(1)
x1 <- runif(100)
x2 <- runif(100)
x3 <- runif(100)
X <- cbind.data.frame(x1, x2, x3)
max_item <- rep(1,3)
min_item <- rep(0,3)
mod <- cirtmodel(X, max.item=max_item, min.item=min_item)
out <- effectiveness_crm(mod$model)
out
```

`effectiveness_poly` *Computes the actual and predicted effectiveness of the collection of algorithms.*

Description

This function computes the actual and predicted effectiveness of the collection of algorithms for different tolerance values.

Usage

```
effectiveness_poly(mod)
```

Arguments

`mod`	A fitted `mirt` model using the function `irtmodel` or `R` package `mirt`.

Value

A list with the following components:

`effectivenessAUC`	The area under the actual and predicted effectiveness curves.
`actcurves`	The `x,y` coodinates for the actual effectiveness curves for each algorithm.
#'	
`prdcurves`	The `x,y` coodinates for the predicted effectiveness curves for each algorithm.

Examples

```
set.seed(1)
x1 <- sample(1:5, 100, replace = TRUE)
x2 <- sample(1:5, 100, replace = TRUE)
x3 <- sample(1:5, 100, replace = TRUE)
X <- cbind.data.frame(x1, x2, x3)
mod <- pirtmodel(X)
out <- effectiveness_poly(mod$model)
out
```

`latent_trait_analysis` *Performs the latent trait analysis*

Description

This function performs the latent trait analysis of the datasets/problems after fitting a continuous IRT model. It fits a smoothing spline to the points to compute the latent trait.

Usage

```
latent_trait_analysis(df, paras, min_item = 0, max_item = 1, epsilon = 0.01)
```

Arguments

`df`	The performance data in a matrix or dataframe.
`paras`	The parameters from fitting `cirtmodel`.
`min_item`	A vector with the minimum performance value for each algorithm.
`max_item`	A vector with the maximum performance value for each algorithm.
`epsilon`	A value defining good algorithm performance. If `epsilon = 0`, then only the best algorithm is considered. A default

Value

A list with the following components:

`crmtheta`	The problem trait output computed from the R package EstCRM.
`crmtheta`	The problem trait output computed from the R package EstCRM.
`strengths`	The strengths of each algorithm and positions on the latent trait that they performs well.
`dfl`	The dataset in a long format of latent trait occupancy.
`plt`	The ggplot object showing the fitted smoothing splines.
`thetas`	The easiness of the problem set instances.
`weakness`	The weaknesses of each algorithm and positions on the latent trait that they performs poorly.

Examples

```
set.seed(1)
x1 <- runif(100)
x2 <- runif(100)
x3 <- runif(100)
X <- cbind.data.frame(x1, x2, x3)
max_item <- rep(1,3)
min_item <- rep(0,3)
mod <- cirtmodel(X, max.item=max_item, min.item=min_item)
```

```
out <- latent_trait_analysis(X, mod$model$param, min_item= min_item, max_item = max_item)
out
```

`make_polyIRT_data` *Converts continuous performance data to polytomous data with 5 categories.*

Description

This function converts continous performance data to polytomous data with 5 categories

Usage

```
make_polyIRT_data(df, method = 1)
```

Arguments

`df`	The input data in a dataframe or a matrix
`method`	If 1, then the data is an accuracy measure between 0 and 1. If 2, then the performance data is possibly has a bigger range. So we divide it into 5 equal bins to make it polytomous.

Value

The polytomous data frame.

Examples

```
set.seed(1)
x1 <- runif(500)
x2 <- runif(500)
x3 <- runif(500)
x <- cbind(x1, x2, x3)
xout <- make_polyIRT_data(x)
```

model_goodness_crm *Computes the goodness of IRT model for all algorithms.*

Description

This function computes the goodness of the IRT model for all algorithms for different goodness tolerances.

Usage

```
model_goodness_crm(mod)
```

Arguments

mod	A fitted mirt model using the function irtmodel or R package mirt.

Value

A list with the following components:

goodnessAUC	The area under the model goodness curve for each algorithm.
curves	The x,y coodinates for the model goodness curves for each algorithm.

Examples

```
set.seed(1)
x1 <- runif(100)
x2 <- runif(100)
x3 <- runif(100)
X <- cbind.data.frame(x1, x2, x3)
max_item <- rep(1,3)
min_item <- rep(0,3)
mod <- cirtmodel(X, max.item=max_item, min.item=min_item)
out <- model_goodness_crm(mod$model)
out
```

model_goodness_for_algo_crm *Computes the goodness of IRT model for a given algorithm.*

Description

This function computes the goodness of the IRT model for a given algorithm for different goodness tolerances.

Usage

```
model_goodness_for_algo_crm(mod, num = 1)
```

Arguments

mod	A fitted mirt model using the function irtmodel or R package mirt.
num	The algorithm number, for which the goodness of the IRT model is computed.

Value

A list with the following components:

xy	The x values denote the goodness tolerances. The y values denote the model goodness.
auc	The area under the model goodness curve.

Examples

```
set.seed(1)
x1 <- runif(100)
x2 <- runif(100)
x3 <- runif(100)
X <- cbind.data.frame(x1, x2, x3)
max_item <- rep(1,3)
min_item <- rep(0,3)
mod <- cirtmodel(X, max.item=max_item, min.item=min_item)
out <- model_goodness_for_algo_crm(mod$model, num=1)
out
```

---

model_goodness_for_algo_poly

Computes the goodness of the IRT model fit for a given algorithm.

Description

This function computes the goodness of the IRT model fit for a given algorithm using the empirical cumulative distribution function of errors.

Usage

```
model_goodness_for_algo_poly(mod, num = 1)
```

Arguments

mod	A fitted mirt model using the function irtmodel or R package mirt.
num	The algorithm number

Value

A list with the following components:

xy	The x values denote the error tolerances. The y values denotes its empirical cumulative distribution function.
auc	The area under the CDF.
mse	The mean squared error.

Examples

```
set.seed(1)
x1 <- sample(1:5, 100, replace = TRUE)
x2 <- sample(1:5, 100, replace = TRUE)
x3 <- sample(1:5, 100, replace = TRUE)
X <- cbind.data.frame(x1, x2, x3)
mod <- pirtmodel(X)
out <- model_goodness_for_algo_poly(mod$model, num=1)
out
```

---

model_goodness_poly *Computes the goodness of IRT model for all algorithms.*

Description

This function computes the goodness of the IRT model for all algorithms using the empirical cumulative distribution function of errors.

Usage

```
model_goodness_poly(mod)
```

Arguments

mod	A fitted mirt model using the function irtmodel or R package mirt.

Value

A list with the following components:

goodnessAUC	The area under the model goodness curve for each algorithm.
mse	The mean squared error.
curves	The x,y coodinates for the model goodness curves for each algorithm.

Examples

```
set.seed(1)
x1 <- sample(1:5, 100, replace = TRUE)
x2 <- sample(1:5, 100, replace = TRUE)
x3 <- sample(1:5, 100, replace = TRUE)
X <- cbind.data.frame(x1, x2, x3)
mod <- pirtmodel(X)
out <- model_goodness_poly(mod$model)
out
```

---

pirtmodel *Fits a polytomous IRT model.*

Description

This function fits a polytomous Item Response Theory (IRT) model to the algorithm performance data.

Usage

```
pirtmodel(dat, ncycle = NULL, vpara = TRUE)
```

Arguments

dat	The performance data in a matrix or dataframe.
ncycle	The number of cycles for mirt. The default is 500.
vpara	It TRUE the verbose parameter for the mirt would be set to true.

Value

A list with the following components:

model	The IRT model using the R package mirt.
anomalous	A binary value for each algorithm. It is set to 1 if an algorithm is anomalous. Otherwise it is set to 0.
stability	The stability of each algorithm.
easiness_threshold	The easiness thresholds for each algorithm. Lower thresholds indicates that the algorithm finds more test instances easy.

Examples

```
set.seed(1)
x1 <- sample(1:5, 100, replace = TRUE)
x2 <- sample(1:5, 100, replace = TRUE)
x3 <- sample(1:5, 100, replace = TRUE)
X <- cbind.data.frame(x1, x2, x3)
mod <- pirtmodel(X)
```

prepare_for_plots_crm *Utility function to make a dataframe from the continuous IRTmodel*

Description

This is a utility function to make a dataframe from the continuous IRTmodel, which makes it easier to plot the surfaces

Usage

```
prepare_for_plots_crm(mod, thetarange = c(-6, 6))
```

Arguments

mod	IRT model, either from function cirtmodel or the R package EstCRM.
thetarange	The range for theta, default from -6 to 6.

Value

Dataframe with output probabilities from the IRT model for all algorithms.

Examples

```
data(classification_cts)
mod <- cirtmodel(classification_cts)
dat <- prepare_for_plots_crm(mod$model)
head(dat)
```

prepare_for_plots_poly *Utility function to make a dataframe from the polytomous IRTmodel*

Description

This is a utility function to make a dataframe from the polytomous IRTmodel, which makes it easier to plot trace lines

Usage

```
prepare_for_plots_poly(mod)
```

Arguments

mod	IRT model, either from function pirtmodel or the R package mirt.

Value

Dataframe with output probabilities from the IRT model for all algorithms.

Examples

```
data(classification_poly)
mod <- pirtmodel(classification_poly)
dat <- prepare_for_plots_poly(mod$model)
head(dat)
```

Index

∗ **datasets**
    classification_cts, 5
    classification_poly, 5

algo_effectiveness_crm, 2
algo_effectiveness_poly, 3

cirtmodel, 4
classification_cts, 5
classification_poly, 5

effectiveness_crm, 6
effectiveness_poly, 7

latent_trait_analysis, 8

make_polyIRT_data, 9
model_goodness_crm, 10
model_goodness_for_algo_crm, 10
model_goodness_for_algo_poly, 11
model_goodness_poly, 12

pirtmodel, 13
prepare_for_plots_crm, 14
prepare_for_plots_poly, 14