Package: ata (via r-universe)

August 21, 2024

Title Automated Test Assembly

Version 1.1.1

Date 2020-11-06

Author Gulsah Gurkan [aut], Michael Chajewski [aut, cre], Sam Buttrey [cph]

Maintainer Michael Chajewski <mchajewski@hotmail.com>

Description Provides a collection of psychometric methods to process item metadata and use target assessment and measurement blueprint constraints to assemble a test form. Currently two automatic test assembly (ata) approaches are enabled. For example, the weighted (positive) deviations method, wdm(), proposed by Swanson and Stocking (1993) <doi:10.1177/014662169301700205> was implemented in its full specification allowing for both item selection as well as test form refinement. The linear constraint programming approach, atalp(), uses the linear equation solver by Berkelaar et. al (2014) <http://lpsolve.sourceforge.net/5.5/> to enable a variety of approaches to select items.

Depends R (>= 3.0.0), lpSolve

Imports stats

License LGPL-2

Copyright file COPYRIGHTS

Encoding UTF-8

LazyData true

RoxygenNote 7.1.1

NeedsCompilation no

Date/Publication 2020-11-10 15:20:08 UTC

Repository https://chajewski.r-universe.dev

RemoteUrl https://github.com/cran/ata

RemoteRef HEAD

RemoteSha b7174e37964be135ecb7c1b482031049a6bff6a6

Contents

`atalp` *Automated Test Assembly via Linear Constrained Programming*

Description

Ingests item metadata jointly with target test form constraints, and can be parametarized to uses either Boolean (0-1) Integer Linear Programming (ILP) or Mixed Integer Linear Programming (MILP) to construct a test form based on the desired objectives. When MILP is desired the selection of the objective function type should be changed.

Usage

```
atalp( ipool,
       id,
       constraints,
       refine = FALSE,
       permutate = FALSE,
       sorttimes = 999,
       tieselect = -1,
       type = "const",
       verbose = TRUE,
       apriorіadd = NA,
       posthocadd = NA )
```

Arguments

`ipool`	Item by characteristic (property) metadata pool.
`id`	Name of unique item identifier.
`constraints`	Complex list object identifying the constraints to be applied in the ATA (see `makeconstobj` for guided process).
`refine`	Creates a final test form from permutated solutions, refined to attempt a deviation balance between the observed form and the constraints. Option only effective if `permutate` is `TRUE` and `type = const` in which the constraint weights have meaning; Default is `FALSE`.

permutate	Requests the test form to be assembled by resorting (sorttimes) the metadata and selecting the most frequently occurring item combination satisfying the constraints. Relevant only for type = const; Default is FALSE.
sorttimes	Number of how often the original input metadata should be resorted. Only functional if permutate is TRUE and type = const; default 999, so that sorttimes + main analysis account for a total of 1,000 selection versions.
tieselect	How should tied items be resolved: -1 (default) - do not manipulate items (which allows for identically functioning items to be included), 1 - select the first item in the list of candidates (sensitive to data sorting); not applicable for situations with all categorical constraints only, 0 - randomly select candidate; not applicable for situations with all categorical constraints only.
type	Type of objective function: const - constraint based only (default), parmin - constraint + minimum non-categorical parameter combination, parmax - constraint + maximum non-categorical parameter combination.
verbose	Should progress be printed to the console? Default TRUE.
apriorideadd	Force item addition (via IDs) to test form before ATA, which affects item selection and constraint attainment success (currently not available).
posthocadd	Force item addition (via IDs) to test form after ATA, which affects final form specifications (currently not available).

Value

A complex list object with test assembly specific estimates:

objective	Constrained objective function value.
items_removed	Removed items from item pool when tieselect is not -1.
excluded	Items from pool excluded.
excluded_set	Item sets excluded. Only included if input constobj includes a set_id.
included	Items from pool included in new test form.
included_set	Item sets from pool included in new test form. Only included if input constobj includes a set_id.
final_ids	Final item ids in the test form.
final_setids	Final set ids in the test form. Only included if input constobj includes a set_id.

Author(s)

Michael Chajewski (mchajewski@hotmail.com), Gulsah Gurkan (gurkangulsah@gmail.com)

References

Chen, P. (2017). Should we stop developing heuristics and only rely on mixed integer programming solvers in automated test assembly? Applied Psychological Measurement, 41, 227-240.

Diao, Q., & van der Linden, W. J. (2011). Automated test assembly using lp_Solve Version 5.5 in R. Applied Psychological Measurement, 35, 398-409.

Shao, C., Liu, S., Yang, H., & Tsai, T. (2019). Automated test assembly using SAS operations research software in a medical licensing examination. Applied Psychological Measurement, 00, 1-15.

van der Linden, W. J. (2005). A comparison of item-selection methods for adaptive tests with content constraints. Journal of Educational Measurement, 42, 283-302.

Examples

```
# Specifying constraints
constin <- list(
  nI = 5,                                      # Number of items on the future test
  nC = 4,                                      # Number of constraints to be satisfied
 nameC = c("Content_A","Content_B","p","iSx"), # Name of constraint; must be numeric and must
  # reflect variable name in input
  lowerC = c(2, 3, 3.00, 0.50),            # Lower bound total constraint value on ATA form
  upperC = c(2, 3, 3.50, 0.60),            # Upper bound total constraint value on ATA form
  wC = c(1, 1, 1, 1),                         # Constraint weight used for weighted sum of
                                               # (positive) deviations St
  set_id = NA                                  # Aggregation ID for units / sets
)

# Running atalp
testLP <- atalp(ipool = metadata_example,
                id = "Item",
                constraints = constin)

# Summary of results
summary(testLP)
```

`makeconstobj` *User-guided Function to Create a Constraints Input for ATA Form Creation*

Description

Guides the user to create a complex list object identifying the constraints to be applied in automated test assembly functions from the `ata` package.

Usage

```
makeconstobj( ipool,
 id,
 empty = FALSE)
```

Arguments

ipool	Item by characteristic (or property) metadata.
id	Name (not actual codes) of unique item identifier (variable).
empty	Should the function return an empty list to be filled in manually. Default is FALSE.

Value

A list object with "nC" "nameC" "lowerC" "upperC" "wC" "nI" "set_id"

Author(s)

Gulsah Gurkan (gurkangulsah@gmail.com), Michael Chajewski (mchajewski@hotmail.com)

References

Parshall, C. G., et al. (2002). Automated test assembly for online administration. In C. G. Parshall, J. A. Spray, J. C. Kalohn, & T. Davey, Practical considerations in computer based testing (pp.106-125). New York, NY: Springer-Verlag New York, Inc.

---

metadata_example *ATA Package Example Item Metadata*

Description

Sample data based on data from Parshall et al. (2002) used for the demonstration of the Weighted (positive) Deviations Method (WDM).

Usage

```
metadata_example
```

Format

A data frame with 10 rows and 10 variables:

Item Unique item identifier, integer.

Content Content label, as factor identifying content "A" and "B".

Content_A Dummy code for content "A", 0 and 1 indicators.

Content_B Dummy code for content "B", 0 and 1 indicators.

p Item proportion correct responding, rounded decimal.

rpbis Item-total point biserial correlation, rounded decimal correlation in range -1.00 to 1.00.

iSx Item contribution to total composite standard deviation, double precision numeric.

Time Expected item response time, in minutes.

Parent0 Item set ID–initial, unique item set name.

Parent1 Item set ID–modified, unique item set name.

References

Parshall, C. G., et al. (2002). Automated test assembly for online administration. In C. G. Parshall, J. A. Spray, J. C. Kalohn, & T. Davey, Practical considerations in computer based testing (pp.106-125). New York, NY: Springer-Verlag New York, Inc.

`metadata_large_example`

ATA Package Large Example Item Metadata

Description

Sample data used to demonstrate automated test assembly.

Usage

```
metadata_large_example
```

Format

A data frame with 1096 rows and 44 variables:

Item Unique item identifier, alpha-numeric index code.

Content Content label, as factor identifying content "A" and "B".

Content_A Dummy code for content "A", 0 and 1 indicators.

Content_B Dummy code for content "B", 0 and 1 indicators.

Content_C Dummy code for content "C", 0 and 1 indicators.

Content_D Dummy code for content "D", 0 and 1 indicators.

Content_E Dummy code for content "E", 0 and 1 indicators.

Content_F Dummy code for content "F", 0 and 1 indicators.

Content_G Dummy code for content "G", 0 and 1 indicators.

Content_H Dummy code for content "H", 0 and 1 indicators.

Content_I Dummy code for content "I", 0 and 1 indicators.

Content_J Dummy code for content "J", 0 and 1 indicators.

Content_K Dummy code for content "K", 0 and 1 indicators.

Content_L Dummy code for content "L", 0 and 1 indicators.

Content_M Dummy code for content "M", 0 and 1 indicators.

Content_N Dummy code for content "N", 0 and 1 indicators.

Content_O Dummy code for content "O", 0 and 1 indicators.

Content_P Dummy code for content "P", 0 and 1 indicators.

Content_Q Dummy code for content "Q", 0 and 1 indicators.

Content_R Dummy code for content "R", 0 and 1 indicators.

Content_S Dummy code for content "S", 0 and 1 indicators.

Content_T Dummy code for content "T", 0 and 1 indicators.

Content_U Dummy code for content "U", 0 and 1 indicators.

Content_V Dummy code for content "V", 0 and 1 indicators.

Content_W Dummy code for content "W", 0 and 1 indicators.

Content_X Dummy code for content "X", 0 and 1 indicators.

Content_Y Dummy code for content "Y", 0 and 1 indicators.

Content_Z Dummy code for content "Z", 0 and 1 indicators.

p Item proportion correct responding.

rpbis Item-total point biserial correlation, rounded decimal correlation in range -1.00 to 1.00

iSx Item contribution to total composite standard deviation.

Time Observed median item response time, in seconds.

Choices Number of response choices in the multiple-choice question.

Answer Correct answer key. In the multiple-choice questions answer 1 = A, 2 = B, and so on.

Skill Formative insight skill classification.

Skill_1 Formative skill insight dummy code for skill 1 (S1): Interpretive.

Skill_2 Formative skill insight dummy code for skill 2 (S2): Factual.

Skill_3 Formative skill insight dummy code for skill 3 (S3): Evaluative.

IIF_m2 Item Response Theory (IRT) item information function value at a theta = -2.0.

IIF_m1 Item Response Theory (IRT) item information function value at a theta = -1.0.

IIF_0 Item Response Theory (IRT) item information function value at a theta = 0.0.

IIF_1 Item Response Theory (IRT) item information function value at a theta = 1.0.

IIF_2 Item Response Theory (IRT) item information function value at a theta = 2.0.

Parent0 Passage based item set parent identification.

`metadata_withreplic_example`

ATA Package Example Item Metadata with Item Replications

Description

Sample data based on data from Parshall et al. (2002) used for the demonstration of the Weighted (positive) Deviations Method (WDM).

Usage

```
metadata_withreplic_example
```

Format

A data frame with 14 rows and 10 variables:

Item Unique item identifier, integer.

Item2 Unique item identifier, character letter.

Content Content label, as factor identifying content "A" and "B".

Content_A Dummy code for content "A", 0 and 1 indicators.

Content_B Dummy code for content "B", 0 and 1 indicators.

p Item proportion correct responding, rounded decimal.

rpbis Item-total point biserial correlation, rounded decimal correlation in range -1.00 to 1.00.

iSx Item contribution to total composite standard deviation, double precision numeric.

Time Expected item response time, in minutes.

Orig_Item Original item copy, corresponding "Item" column ID.

References

Parshall, C. G., et al. (2002). Automated test assembly for online administration. In C. G. Parshall, J. A. Spray, J. C. Kalohn, & T. Davey, Practical considerations in computer based testing (pp.106-125). New York, NY: Springer-Verlag New York, Inc.

`plot.ata` *Generic Plot Function for Class* `ata`

Description

Default plotting function for output objects of class ata. The function detects the object's method and renders the appropriate visualizations.

Usage

```
\method{plot}{ata}(x,
      conditem=NA,
      useconst=TRUE,
      itemorder=NA,
      itemlab=NA,
      useitemlab=FALSE,
      together=FALSE,
      ...)
```

Arguments

x	An output object of class ata generated by either wdm() or atalp() from the ata package.
conditem	Provides a conditional or secondary item classification (i.e. content label). If provided, it must be given in the order of final_ids in the ata output object.
useconst	Indicator whether all constraints from the test assembly process should be visualized or whether only a selection is desired. If a selection is desired, the name of the constraint as given by the constobj should be provided. Default is TRUE.
itemorder	Identifies the item order with which to visualize constraints. If NA the observed order in the ata object will be used. If provided, the new order for the order of final_ids in the ata output object must be given.
itemlab	Identifies item labels. Default is NA. If NA, then the item ids in the final_ids vector of the ata output object will be used. If provided, ids must be given in the order of final_ids in the ata output object.
useitemlab	Identifies if items should be labeled. Default is FALSE. If FALSE then the item order in the final_ids vector of the ata output object will be used as labels. If TRUE, but itemlab is not provided, then the ids from the final_ids vector will be used.
together	Should progress plots be stacked together in one plot? Default is FALSE. Not advisable for situations with more than 5 constraints.
...	Arguments to be passed to methods.

Value

The function returns plots of the test form constraints and a cumulative additive constraint list for each constraint if assigned to an object.

plots	For each constraint in the test form two visualizations are considered: 1) A cumulative additive progressive plot showing the change in the constraint total value per selected item, and 2) a plot of the constraint item specific value for each selected item.
cumulative	If plot.ata is assigned to an object, the object will inherit a list of length equal to the number of constraints each element containing the cumulative constraint value after each selected item.

Author(s)

Michael Chajewski (mchajewski@hotmail.com)

summary.ata *Generic Summary Function for Class* ata

Description

Default summary function for output objects of class ata. The function provides a brief summary of the ATA form in text, and provides a binary table of constraint success.

Usage

```
\method{summary}{ata}(object, ...)
```

Arguments

object	An output object of class ata generated by either wdm() or atalp() from the package.
...	Additional arguments affecting the summary produced.

Value

The function returns a statement summarizing the evaluation of the assembled test form. Additionally, the function will return a pattern matrix for the test form constraints if assigned to an object.

statement	A summary of items (and/or item sets) in the test form and the overview of constraint success.
pattern	A matrix of constraints by a classification if the additive constraints are below, at or above the constraint specific user provided bounds. This matrix, only returned if summary.ata is assigned to an object, will always demonstrate meeting all criteria for atalp test forms as all criteria have to be met to obtain a feasible solution.

Author(s)

Gulsah Gurkan (gurkangulsah@gmail.com), Michael Chajewski (mchajewski@hotmail.com)

wdm *Automated Test Assembly via Weighted (positive) Deviations Method*

Description

Ingests item metadata jointly with target test form constraints and uses the Weighted (positive) Deviations Method (WDM) to construct a test form based on the desired objectives.

Usage

```
wdm( ipool,
 id,
 constraints,
 first = NA,
 refine = TRUE,
 permutate = FALSE,
 tieselect = 1,
 verbose = TRUE,
 apriorіadd = NA,
 posthocadd = NA )
```

Arguments

ipool	Item by characteristic (property) metadata pool.
id	Name of unique item identifier.
constraints	Complex list object identifying the constraints to be applied in the ATA (see makeconstobj for guided process).
first	How should item selection start: id of the item to be selected first from the pool, NA (default) - select randomly from the pool.
refine	Should the final test form be refined against the remaining item pool? Default is TRUE.
permutate	Assemble test forms starting with each item sequentially (as many forms as items in pool) and define final test form based on eligible constraint compliant solutions; Default is FALSE (currently not available).
tieselect	How should tied items be resolved: 1 (default) - select the first item in the list of candidates (sensitive to data sorting); not applicable for situations with all categorical constraints only, 0 - randomly select candidate; not applicable for situations with all categorical constraints only
verbose	Should progress of wdm() be printed to the console? Default = TRUE.
apriorіadd	Force item addition (via IDs) to test before ATA, which affects item selection and constraint attainment success (currently not available).
posthocadd	Force item addition (via IDs) to test after ATA, which affects final form specifications (currently not available).

Value

A complex list object with test assembly specific estimates:

wde	Estimates of the computational steps deriving the positive weighted deviations and item selection.
evaluation	Final assembled test form additive properties across constraints.
considered	Estimates of the computational steps when refine = TRUE evaluating selected items and selecting replacements.
excluded	Items from pool excluded.

excluded_set	Item sets excluded. Only included if input constobj includes a set_id.
included	Items from pool included in new test form.
included_set	Item sets from pool included in new test form. Only included if input constobj includes a set_id.
initial_ids	Item sets from pool included in new test form.
initial_setids	Item sets from pool included in new test form. Only included if input constobj includes a set_id.
final_ids	Final item ids in the test form.
final_setids	Final set ids in the test form. Only included if input constobj includes a set_id.

Author(s)

Gulsah Gurkan (gurkangulsah@gmail.com), Michael Chajewski (mchajewski@hotmail.com)

References

Parshall, C. G., et al. (2002). Automated test assembly for online administration. In C. G. Parshall, J. A. Spray, J. C. Kalohn, & T. Davey, Practical considerations in computer based testing (pp.106-125). New York, NY: Springer-Verlag New York, Inc.

Sanders, P. F., & Verschoor, A. J. (1998). Parallel test construction using classical item parameters. Applied Psychological Measurement, 22, 212-223.

Swanson, L., & Stocking, M. L. (1993). A Model and heuristic for solving Very large item selection problems. Applied Psychological Measurement, 17, 151-166.

Examples

```
# Specifying constraints
constin <- list(
  nI = 5,                                     # Number of items on the future test
  nC = 4,                                     # Number of constraints to be satisfied
 nameC = c("Content_A","Content_B","p","iSx"), # Name of constraint; must be numeric and must
  # reflect variable name in input
  lowerC = c(2, 3, 3.00, 0.50),            # Lower bound total constraint value on ATA form
  upperC = c(2, 3, 3.50, 0.60),            # Upper bound total constraint value on ATA form
  wC = c(1, 1, 1, 1),                         # Constraint weight used for weighted sum of
                                              # (positive) deviations St
  set_id = NA                                 # Aggregation ID for units / sets
)

# Running WDM example from Parshall et al. (2002)
testWDM <- wdm( ipool = metadata_example,
                id = "Item",
                constraints = constin,
                first = 2)

# Summary of results
summary(testWDM)
```

Index