DELTA SCORING FUNCTION REFERENCE

D. Atanasov

October 24, 2021

1 Introduction

This document describes the set of functions and the way of their ussage for obtaining the DELTA SCORING approach for test evaluations.

2 Instalation

To install the package download it from
`https://github.com/amitko/matlab-delta-scoring.git`.
Place the folder in the MATLAB path and rename it to `+deltaScoring`.

3 Ussage

Here is an example of a simple ussage of the package.

Suppose the raw dichotomous item response is placed in variable `itemScore`.

To estimate the expected item difficulties (”deltas”) the bootstrapping procedure

```
[ItemDelta, estimatedDeltaSE] = ...
        deltaScoring.estimate.itemDeltaBootstrap(
           itemScore);
```

is called. The resulted item deltas and the corresponding standard error of estimate are returned in variables `ItemDelta` and `estimatedDeltaSE`.

The classical person D-scores are calculated using the response paterns in itemScore and already calculated item deltas.

```
personDscores = ...
        deltaScoring.scoring.dScore(ItemDelta,
           itemScore,opt.Dscore_method);
```

Here `opt` is a structure containing the options for the considered delta scoring model. It can be generated by

```
opt = deltaScoring.scoring.Options;
```

Here and after the default will be the model RFM2. If a RFM3 model is required this can be stated in

```
opt.model = 3;
```

and the corresonding options should be passed to the functions.

The item parameters location b and shape s can be obtained by

```
[params, CI, ~, Results] = ...
deltaScoring.estimate.logitDeltaFit(itemScore,
   personDscores,opt);
```

where `params` contains the matrix with corresponding parameters for any item in the test $[b, s]$. The first column corresponds to the location paramater b while the second represents the shape s. If the model is RMF3, the guessing parameter is in the third column.

The matrix `CI` contains the 95% confidence interval of the estimated values. `Results` contains additional fitting parameters (for example MAD is available in `Results.MAD`).

The persons true scores can be calculated by

```
personTrueScores = ...
deltaScoring.scoring.trueScore(ItemDelta,
   ItemParameters,personDscores,opt);
```

and the SE

```
personTrueScoresSE = ...
deltaScoring.scoring.trueScoreSE(ItemDelta,
   ItemParameters,personDscores,opt);
```

A latent verssion of the location and shape parameters (together with their SE) can be obtained by

```
[LatentParams,LatentSE] =  deltaScoring.estimate.
   ML_RFM_params( itemScore, personDscores, opt);
```

The corresponding MAD is obtained by

```
LatentMAD = deltaScoring.item.MAD(LatentParams,
   Results.observedLogitDelta ,opt);
```

where `Results.observedLogitDelta` is calculated with `deltaScoring.estimate.logitDeltaFit` above and contains the proportion of observed correct answers for the values on the D-score scale.

The corresponding latent values of the person D-scores (and SE) can be obtained by

```
[latentScore, ~, latentScoreSE]= ...
deltaScoring.estimate.ML_RFM_scores( itemScore,
   LatentParams, opt)
```

The estimations of latent item parameters and person D-scores can be iterated untill a convergence is reached

```
OldLatentParams = zeros(size(LatentParams));
while max(abs(OldLatentParams - LatentParams)) > eps
        OldLatentParams = LatentParams;
        [LatentParams,LatentSE] =  ...
                deltaScoring.estimate.ML_RFM_params(
                    itemScore, personDscores, opt);
        [latentScore, ~, latentScoreSE]= ...
                deltaScoring.estimate.ML_RFM_scores(
                    itemScore,LatentParams, opt)
end
```

Equating of different tests can be performed by functions located in `deltaScoring.equating`. Here an example of equating of classical (non-latent) parameters of the test will be presented. The equating is based of caclulation of two constants A and B which represent the change of scale for the test equating. These constants can be calculated on the base of item deltas of the target test `targetDeltas`, item deltas (allready colculated above) and few common items between the two tests (stated in the variable `CommonItems`)

```
[A,B] = deltaScoring.equating.constants(
   targetDeltas, ItemDelta, CommonItems);
```

```
equatedItemDeltas = deltaScoring.equating.rescale
   (ItemDelta,A,B);
equatedDscores = deltaScoring.scoring.dScore(
   ItemDelta,itemScore,opt.Dscore_method);
```

4 Function reference

deltaScoring.assembly.multipleTest

```
multipleTest(nOfTest, nOfItems, itemDeltas,
   varargin)

Returnt the item idexes from the itemParams
which compose a number of tests test.

nOfTest
nOfItems - number of items in teh test.
itemDeltas - list (column) of estimated item deltas

Optional parameters: ['Name',value] pairs
The approach is based on Linear Optimal Test Design
Uses singleTest.
```

deltaScoring.assembly.singleTest

```

Returnt the item idexes from the itemParams
which compose a test.

nOfItems - number of items in teh test.
itemDeltas - list (column) of estimated item deltas

Optional parameters: ['Name',value] pairs
The approach is based on Linear Optimal Test Design
```

deltaScoring.dif.conditionalDIF

```
conditionalDIF(focal_params,reference_params,o)

probabilities for correct item performance between
focal and reference group, based on parameters of
the items for the two groups.

o - deltaScoring.scoring.Options
```

deltaScoring.dif.ESonDIF

```
[onFocal, onReference] = ESonDIF(focal_params,
   reference_params,o)
Calculates the effect size on focal and reference
   group for DIF

INPUT:
               focal_params     - item parameters,
                  estimated on focal group
               reference_params - item parameters,
                  estimated on reverence group
      o                - options
```

deltaScoring.dif.functioning

```
Calculates different characteristics, corresponding
    to the DIF
```

deltaScoring.dif.Mantel_Haenszel

```
[a_MH,log_a_MH_SE,da_MH,z,p,MH,type,against]=
   Mantel_Haenszel(response,score,groups,reference)
Calculates Mantel-Haenszel statistics
```

deltaScoring.dif.MPDonDIF

```
[onFocal, onReference] = MPDonDIF(dif)
dif = pcr_f - pcr_r;
```

deltaScoring.dif.paramsForGroups

```
[focal_params,reference_params,deltasF,deltasR] =
   paramsForGroups(itemResponse,group,latent,o)
Calcultes the item parameters for the focal and
   reference group

group 0 - reference, 1 - focal
params by default are latent
```

deltaScoring.dif.plotConditionalDIF

```
plotConditionalDIF(dif,opt,visible)
Plots conditional DIF for the Focal and reference
   group
```

deltaScoring.dif.plotICC

```
plotICC(focal_params,reference_params,o,visible)
Plots ICC for the Focal and reference group
```

deltaScoring.dif.plotTSC

```
plotTSC(focal_params,reference_params,o,visible)
Plots test score for the Focal and reference group
```

deltaScoring.dif.testing

```
[DIF, DIFT,  HA, HB, HAT, HBT, Results] = testing(
   focal_params, reference_params, o)
Tests DIF having estimated item parameters for
   focal
and reference group.

DIF - indicator for a specific item
      0 - no DIF
      1 - against focal
      2 - against reference
DIFT - DIF on test level
HA   - Hypothesis nder approach A. Cell array of
   structures
HB   - Hypothesis nder approach B. Cell array of
   structures
HAT on test level
HBT on test level
Results - structure with detailed results
```

deltaScoring.equating.constants

```
[A,B] = constants(Base_test_deltas,New_test_deltas,
   common_items)
             Y        X
Calculates the rescaling constants, based on common
    items
between two test.

INPUT:
       Base_test_deltas - item deltas of the base
          test
       New_test_deltas  - item deltas of the new
          test
  common_items     - twoo columns
                               [base_test_item_id
                                  new_test_item_id]
```

```
OUTPUT:
               A and B
```

---

deltaScoring.equating.dscore_common

```
NOT IN USE
```

---

deltaScoring.equating.dscore_rfm

```
[Dscore_equated] = dscore_rfm(X_params, X_rescaled,
    Dscores, type, o)
Calculates equated latent D-Score based on the
   latent parameters

INPUT:
               X_params   - latent parameters of
                  the test
      X_rescaled - rescaled parameters of the test
         after equating
               Dscores    - persons D-score
               type       - default value is m1
               o              -  options

OUTPUT:
               Dscore_equated - equated D-score
```

---

deltaScoring.equating.dscoreOnSubtest

```
NOT IN USE
```

---

deltaScoring.equating.rescale

```
res = rescale(deltas,A,B)
Rescale the item deltas of a test according
to rescaling constants A and B.
```

deltaScoring.equating.rescale_rfm

```
[X_params_rescaled, opts] = rescale_rfm(X_params,
  Y_params,common_items,method_type,o)
Calculates the rescaled latent item parameters of
  test X on the scale of the base test Y.

INPUT:
           X_params - parameters of test X
           Y_params - parametres of test Y
           common_items - common items;  twoo
             columns
                       [base_test_item_id
                         new_test_item_id]
           method_type - Rescaling of the shape
              parameter s has two options
                                   direct

                                     [
                                     default
                                     ]

                                     |

                                     trough_a


OUTPUT:
           X_params_rescaled
           opts - Structure eith
                       bA,bB,sA,sB
```

deltaScoring.estimate.EM_RFM

```
 Function [pars,ability] = irt.
   ItemParametersEstimate_EM_3PL( data,o)
```

```
     estimates the parameters of the item
        characreristic
     curves under the IRT model usen the EM
        algorith.

 Input:
     data - Dihotomous item response
     o    - scoring.Options (optional)
 Output:
     pars - Item parapeters
          [difficulty, discrimination, guessing]
```

deltaScoring.estimate.itemDeltaBootstrap

deltaScoring.estimate.latentLklh

```
latentLklh(xi,itemResponse,deltaScores,o)
Calculates person likelihood on a specific test
   with a specific item response
For internal use in estimations
```

deltaScoring.estimate.logitDeltaFit

deltaScoring.estimate.logitDeltaPlot

```
h = logitDeltaPlot(GF,observedLogitDelta,o)
Plots the fit and the estimated logistics curve
Returns the figure object

INPUT:
   GF - output from logitDeltaFit
   observedLogitDelta - from Results of
      logitDeltaFit
   o - options
          dScale
```

deltaScoring.estimate.ML_RFM_params

```
[pars,se] = ML_RFM_params( itemResponse,
   deltaScores, o)
Estimates the latent parameters of the items base
   on
RFM model, using JML approach.

INPUT:
               itemResponse - dichotomous item
                  response
               deltaScores  - person D-scores
               o            - oprions

OUTPUT:
               pars         - estimated parameter
                  values
               se           - standard errors of
                  the estimates
```

deltaScoring.estimate.ML_RFM_scores

```
[scores,se, see]=ML_RFM_scores( itemResponse,
   itemParams, o)
Estimates the latent parameters for person
   abilities, based on
RFM model, using JML approach.

INPUT:
               itemResponse - dichotomous item
                  response
               itemParams  - person D-scores
               o            - oprions
```

```
OUTPUT:
               scores       - person latent D-
                  scores
               se           - standard errors of
                  the estimates
      see          - analitical solution for se
```

deltaScoring.generate.guttman

```
guttman(NofPersons,itemParams,options,reverse)
Generates a item response according Guttman concept
```

deltaScoring.generate.itemResponse

```
[res, out] = itemResponse(Persons,itemParams,
   options,env)
Generates an item response Persons over a set
of items, defined by their item Parameters.

env is a structure containing additional
information about cheating and guessing.
```

deltaScoring.item.characteristicsFromParameters

```
res = characteristicsFromParameters(item_params,o)
Calculates the item characteristics from item
   parameters

INPUT:
       item_params - item parameters
       o           - options

OUTPUT
       res - [location, discrimination]
```

deltaScoring.item.icc

```
res = icc(itemParameters,o)
Plots ICC curves under given item parameters

INPUT:
       itemParameters
       o - options

OUTPUT:
       res  - figure handle
```

deltaScoring.item.MAD

```
res = MAD(params,observedLogitDelta,o)
Calculates the Mean Absolute Difference between
opserved probability for correct response and
predicted probability obtained under the
selected RFM model.

INPUT:
       params             - item parameters
       observedLogitDelta - observed PCR
       o                  - options

OUTPUT:
       res - MAD values
```

deltaScoring.item.parametersFromCharacteristics

```
res = parametersFromCharacteristics(location,
   discrimination,o)
Calculates the item parameters from item
   characteristics
```

```
 INPUT:
        location
        discrimination
        o           - options

 OUTPUT
        res - [b,s]
```

deltaScoring.person.aberrant

```
 res = aberrant(itemParams,itemDeltas,Dscores,
    itemResponse,options)
 Find aberrant person behaviour according
 to the "quantile method" according to
 An Examination of Different Methods of Setting
    Cutoff Values in Person Fit Research
 Amin Mousavi, Ying Cui & Todd Rogers

 INPUT:
        itemParams   - item parameters
        itemDeltas   - item deltas
        Dscores      - persons D-scores
        itemResponse - dichotomous item response

 OUTPUT:
        res - aberrant indicator 0/1
```

deltaScoring.person.fitHStatistics

```
 fitHStatistics(item_response)
 Calculates the H statistics for the
 dichotomous item response
```

deltaScoring.person.fitIndexZ

```
fitIndexZ(lklh,Elklh,Vlklk)
 Inputs are from person.likelihood

based on
D. Dimitrov, R. Smith. Adjusted Rasch Person-Fit
   Statistics. J. of
Applied measurement. 2006
```

deltaScoring.person.fitMSE

```
[Outfit, Infit] = fitMSE(item_response,
   expected_item_score)
Calculates the Outfit and Infit of the MSE fit
```

deltaScoring.person.fitU

```
fitU( params, Dscore, responses, o)
Calculates U statistics
```

deltaScoring.person.fitUD1

```
fitUD1(deltas, Dscore, responses)
Calculates UD1 Statistics
```

deltaScoring.person.fitUD2

```
fitUD2( deltas, Dscore, responses, params, o)
Calculates UD2 statistics
```

deltaScoring.person.likelihood

```
[res, expected, variance] = likelihood(dScores,
   item_parameters,item_response,o)
```

```
Calculates the likelihood for a specific response
   pattern in item_response from a
person with ability in dScores, over a set of items
    in item_parameters.


based on
D. Dimitrov, R. Smith. Adjusted Rasch Person-Fit
   Statistics. J. of
Applied measurement. 2006
```

deltaScoring.person.plotConditionalSE

```
plotConditionalSE(dScores,SE)
Plots conditional standard eroor
```

deltaScoring.person.responeseProbability

```
res = responeseProbability(delta,item_parameters,o)
Calculates the probability for correct response
   from a
person with ability delta over the set of items
   with
parameters, defined in item_parameters.

INPUT
  delta - Person ability
  item_parameters - Item parameters [b,s,c]
  o - Delta Scoring Options
```

deltaScoring.poly.ccr

```
ccr(itemParameters,observedLogitDelta,o)
Plots Cathegory Characteristic Curves for the
Polytomous items
```

```
 INPUT:
   itemParameters - row vector of values of the
      difficulty parameter
                          for item grades.
         observedLogitDelta    - person ability value
         o             - options
```

---

deltaScoring.poly.itemPerformance

```
 res = itemPerformance(itemParameters,delta,o)
Calculates the probability for correct performance
   for the polytomous item

 INPUT:
   itemParameters - row vector of values of the
      difficulty parameter
                          for item grades.
         delta             - person ability value
         o             - options

 OUTPUT:
        res - probability for correct performance
```

---

deltaScoring.poly.logitDeltaPlot

```
 h = logitDeltaPlot(GF,observedLogitDelta,o)
 Plots the fit and the estimated logistics curve
 Returns the figure object

 INPUT:
    GF - output from logitDeltaFit
    observedLogitDelta - from Results of
       logitDeltaFit
    o - options
           dScale
```

```
OUTPUT:
       h - figure habdle
```

deltaScoring.scoring.dScore

```
Returns the so called d-Score for a person
with a given response vector over a set
of items with precalculated deltas;

INPUT:
       itemDeltas - item delta values
  response   - 0/1 item response
  t          - type
                         total / relative_to_n /
           relative_to_d [default]
```

deltaScoring.scoring.dScoreSE_IRT

```
Returns the so called d-Score SE for a person
with a given a given ability theta (on a logit
   scale)
over a set of items with IRT parameters [b,a,c];

left here only for convenience.
```

deltaScoring.scoring.itemPersonMap

deltaScoring.scoring.observedLogitDelta

```
observedLogitDelta(ItemResponse, Dscore,o)
Calculates the proportions of the correct scores
for different falues of the dScale
```

```
INPUT:
  ItemResponse - dichotomous item response 0/1
  Dscore       - estimated person's dScore
  o            - options

OUTPUT:
       res - proportion of correct answers on
          dScale
```

deltaScoring.scoring.Options

```
option = Options(varargin)
Defines the options for DELTA SCORING
Default Values

        NofSamplesForBootstrapping: 1000
  sampleProportionForBootstrapping: 0.1000
           estTypeForBootstrapping: 'mode'
                            dScale: [21x1 double]
                            Models: {1x3 cell}
                        ModelNames: {'RFM1'  'RFM2'
                             'RFM3'}
                  ModelFixedParams: [1x1 struct]
                Model_coefficients: {'b'  's'  'c'
                    'd'}
                             model: 2
                              type: 'raw'
                skipObservedOnPlot: 2
                  aberrantQuantile: 0.7000
                                EM: [1x1 struct]
                     StartingPoint: [0.5000 1
                        0.1000]
                             Lower: [0.0100 0.2000
                                0]
                             Upper: [0.9900 5
                                0.5000]
```

```
                 RFM_params_method: 'constrained'
```

deltaScoring.scoring.PCR

```
res = PCR(params,delta,o)
Probability for correct response

  INPUT:
    params - logistics parameres
    delta  - delta values
    o      - options
              mmodel
```

deltaScoring.scoring.poly2dih

```
[DIHscores,Poly,Org] = poly2dih(Response)
Convert polytomous to dihotomous item respone

INPUT:
       Response - polytomous item response

OUTPUT:
       DIHscores - dihotomous ite response
       Poly      - indicator for polytomous items
       Org       - Labels, etc. for poly items
```

deltaScoring.scoring.scaledScore

```
res=scaledScore(scores,t)
Scales the score according to type t
t = 'range' scales in range 0..100
```

deltaScoring.scoring.trueScore

```
res = trueScore(itemDeltas, parameters, dScore, o)
Calculates the true-score measure for person's
   dScode
on a set of items with delta scores in itemDeltas,
logistic parameters of the items and the persons
   dScore

 INPUT:
  itemDeltas - item's delta scores
  parameters - item's logistics parameters
  dScore     - persons dScore
  o          - Options (defaults scoring.Options)
```

deltaScoring.scoring.trueScoreSE

```
res = trueScoreSE(itemDeltas, parameters, dScore, o
   )
Calculates the true-score SE measure for person's
   dScode
on a set of items with delta scores in itemDeltas,
logistic parameters of the items and the persons
   dScore

 INPUT:
  itemDeltas - item's delta scores
  parameters - item's logistics parameters
  dScore     - persons dScore
  o          - Options (defaults scoring.Options)
```