Whats' beyond Concerto: An introduction to the R package *catR*

Session 5:

CAT application(s) with an item bank of polytomously scored items

The Psychometrics Centre, Cambridge, June 10th, 2014

Outline:

1. Polytomous item bank
2. CAT settings
3. *catR* application
4. Output
5. Yours to play...

1. Polytomous item bank

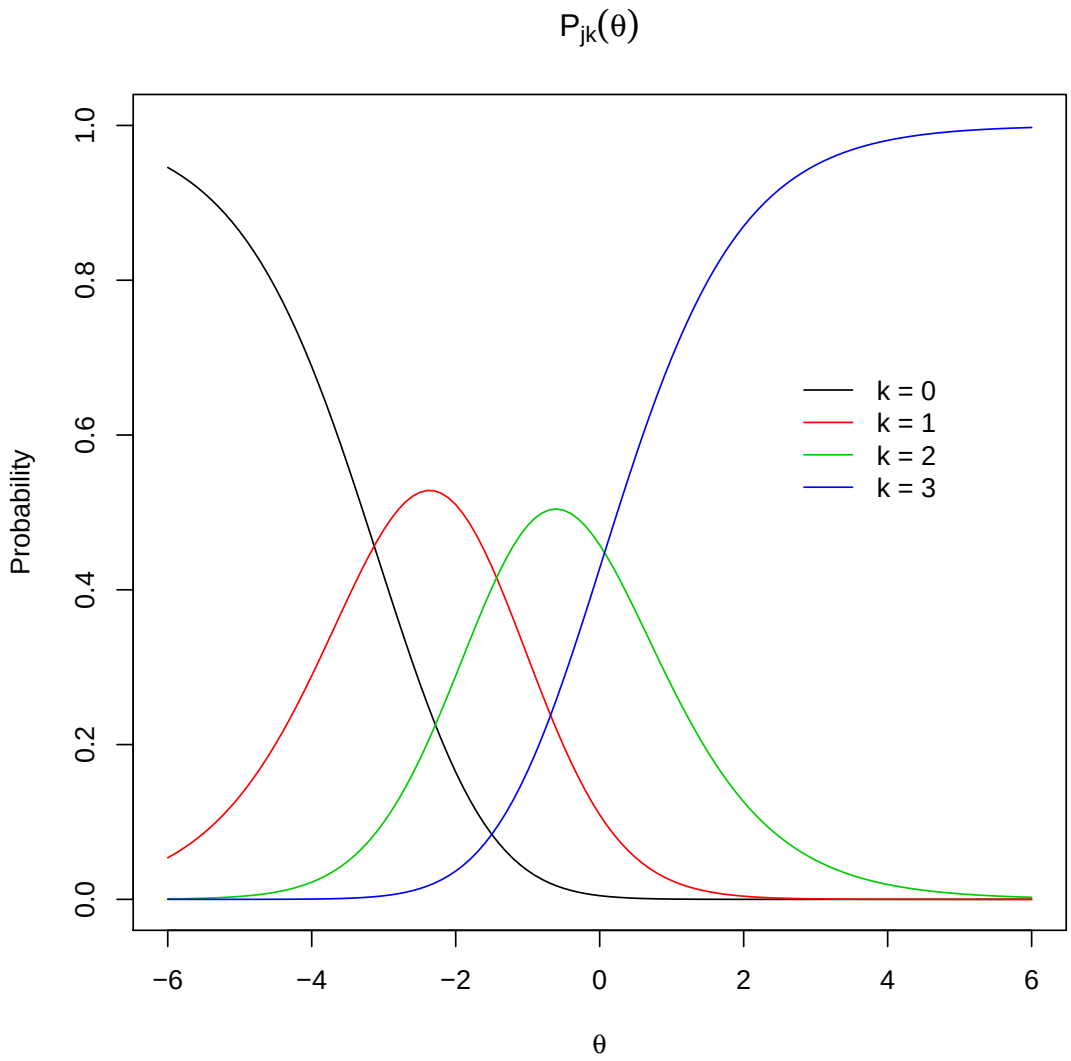
catR holds functions to generate item banks: `genDichoMatrix` and `genPolyMatrix`

Useful for simulation studies (generation of large item banks with pre-specified parent distributions for item parameters)

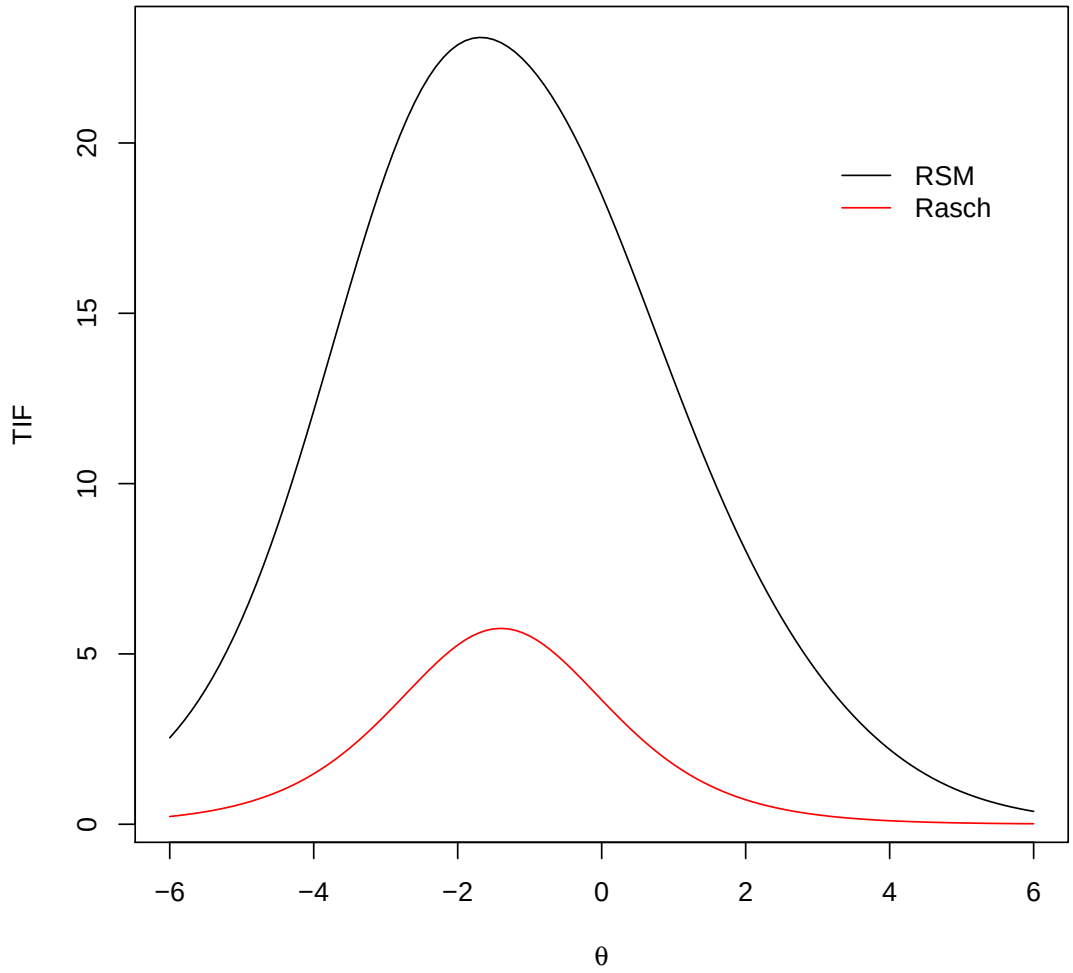
Here however, I embedded the dichotomous item bank in a polytomous version by:

- considering the RSM,
- keeping the original item difficulties as λ_j parameters,
- assuming four response categories,
- fixing category thresholds $(\delta_1, \delta_2, \delta_3)$ equal to (-1.4, 0.3, 1.8)

1. Polytomous item bank



1. Polytomous item bank



1. Polytomous item bank

To load the `polyIQ` bank in R:

1. Load *catR* package in R:

```
R> require(catR)
```

2. Set a working directory wherein all files are stored:

```
R> setwd("C:/Users/David/Desktop/")
```

3. Load the item bank:

```
R> bank <- read.table("polyIQ.txt", header =
TRUE)
```

4. Convert the bank in a matrix:

```
R > bank <- as.matrix(bank)
```

1. Item bank

To see the item bank as stored in R:

```
R > bank
```

which returns

```
      lambdaj delta1 delta2 delta3
 [1,]  -1.732   -1.4    0.3    1.8
 [2,]  -1.020   -1.4    0.3    1.8
 [3,]  -2.100   -1.4    0.3    1.8
 [4,]  -2.535   -1.4    0.3    1.8
 [5,]  -1.258   -1.4    0.3    1.8
          .       .      .      .
          .       .      .      .
          .       .      .      .
```

2. CAT settings

In a first step, the same design will be used as previously with the dichotomous IQ bank:

- Random generation of one CAT response pattern for true proficiency level $\theta = 0$
- Two starting items, selected as being most informative for proficiency levels $\theta = -1$ and $\theta = 1$
- Next item selection by maximum (Fisher) information (MFI)
- Ad-interim proficiency estimation by EAP with standard normal prior (default)
- Stopping rule: after 10 items
- Final proficiency estimation by ML

3. *catR* application

Main strength of *catR*: implementation of each step is identical to the dichotomous case!

- All options of each step (starting, test, stopping, final) must be provided as lists
- Elements of a list have specific names and allowed values
- All elements have by-default values (so specify only interesting elements)

Only difference: `randomCAT` holds the option `model` to specify the type of polytomous IRT model in use

If `model` is not specified, then a dichotomous IRT model is considered (cfr session 3)

3. *catR* application

The starting list is set as follows:

```
R > startList <- list(nrItems = 2, theta = 0,
                      halfRange = 1)
```

Explanations:

- `nrItems` sets the number of initial items (by default 1)
- `theta` sets the center of the range of starting proficiency levels
- `halfRange` sets the half-range of the interval of starting proficiency levels
- As another example, setting `nrItems = 3`, `theta = 1` and `halfRange = 0.5` yields the starting proficiencies (0.5, 1, 1.5)

3. *catR* application

The testing list is set as follows:

```
R > testList <- list(method = "EAP", itemSelect
                     = "MFI")
```

Explanations:

- `method` sets the ad-interim proficiency estimator (by default "BM")
- `itemSelect` sets the method for next item selection
- `"MFI"` is the acronym for `maximum Fisher information` (default method)

3. *catR* application

The stopping list is set as follows:

```
R > stopList <- list(rule = "length", thr = 10)
```

Explanations:

- `rule` sets the stopping rule, `"length"` is the default value
- `thr` sets the numerical value related to the stopping rule

3. *catR* application

The final list is set as follows:

```
R > finalList <- list(method = "ML")
```

Explanations:

- Basically, only the final proficiency estimator is required
- Specified through `method` argument

Note: `startList`, `testList`, `stopList`, `finalList` are just names of variables in R!

3. *catR* application

Now, to set the CAT with the polyIQ, make use of the `randomCAT` function and add `model="RSM"` in it:

```
R > res <- randomCAT(theta = 0, itemBank = bank,
model="RSM", start = startList, test = testList,
stop = stopList, final = finalList)
```

Explanations:

- `model` sets the type of item bank
- by default, `model = NULL` and a dichotomous item bank is assumed
- possible values are: `"GRM"`, `"MGRM"`, `"PCM"`, `"GPCM"`, `"RSM"` and `"NRM"`

4. Output

Let's have a look at the R session and output...

5. Yours to play...

To end up this session, I propose to let you try *catR* by yourself

Design:

- Generation of a response pattern for true proficiency level $\theta = -1$
- Item bank: polyIQ
- Starting step: 3 items selected as most informative at proficiency levels -1, 0 and 1
- Test step: next item selection by Kullback-Leibler criterion, ad-interim proficiency estimation by maximum a posteriori with standard normal prior
- Stopping step: end item administration once the ad-interim SE is smaller than 0.4

5. Yours to play...

To end up this session, I propose to let you try *catR* by yourself

Design:

- Final step: final proficiency estimation by weighted likelihood estimation
- In addition: item exposure control with the randomesque method and the selection of 3 randomesque items

Now it's yours to play :-)

5. Yours to play...

Solution:

- Starting step:

```
R > startList <- list(nrItems = 3, theta = 0,
halfRange = 1)
```

- Test step:

```
R > testList <- list(itemSelect = "KL",
method = "BM", randomesque = 3)
```

- Stopping step:

```
R > stopList <- list(rule = "precision",
thr = 0.4)
```

- Final step:

```
R > finalList <- list(method = "WL")
```

5. Yours to play...

Solution:

- CAT generation:

```
R> res <- randomCAT(trueTheta = -1,
itemBank = bank, model="RSM", start = startList,
test = testList, stop = stopList,
final = finalList)
```

- Display of results:

```
R> res
```