PCM calibration with the "eRm" package

```
install.packages("eRm") #3.1_1
library(eRm) #3.1_2
setwd("c:/mystudy/RIRT/") #3.1_3
ddd<-read.fwf("c3_pcm.dat", width=c(rep(1,5))) #3.1_4
names(ddd)<-paste("I", 1:5, sep="") #3.1_5
dim(ddd) #3.1_6
head(ddd) #3.1_7
summary(ddd) #3.1_8
apply(ddd, 2, table) #3.1_9
```

#3.1_1. The "eRm" package is installed onto the computer using "`install.packages()`." This only needs to be done once on a computer system. See #2.1_1.

#3.1_2. The "eRm" package must be loaded into the current R session using the "`library()`" function. See #2.1_2.

#3.1_3. The working directory where data and R output files are stored is established using "`setwd()`." For the PCM, the "`c3_pcm.dat`" file should be saved into the folder established as the working directory. See #2.1_3.

#3.1_4. Read the item response data file "`c3_pcm.dat`" into the R session using "`read.fwf()`" since the item responses are arranged with a fixed format style having no space between columns. The data are saved it as an R object called "ddd." Each of the five columns represents an item and each row represents a person. See #2.2_1.

#3.1_5. Replace the default variable, or item, names by new names ("I1" through "I5"). See #2.1_8.

#3.1_6. The dimensions of "ddd" are providing using the "`dim()`" function. The number of rows and the number of columns are shown; in this application, 900 test-takers and 5 items. See #2.1_5.

#3.1_7. The first six rows of the data are displayed using "`head()`." These data are polytomous with four category response options: 0, 1, 2, or 3.

#3.1_8. The "`summary()`" command produces descriptive statistics (the first quartile, second quartile [median], third quartile, minimum, maximum, and mean) for each item's responses. The value of the mean for each item is the item easiness, or the item difficulty in classical test theory. For example, in a Likert style psychological test with response options, 0 = *Strongly Disagree*, 1 = *Disagree*, 2 = *Agree*, and 3 = *Strongly Agree*, the higher the item mean, or item easiness, the easier it was for test-takers to respond in a higher category to endorse, or agree with, the item statement.

```
> summary(ddd)
       I1              I2              I3              I4              I5
 Min.   :0.000   Min.   :0.000   Min.   :0.000   Min.   :0.000   Min.   :0.000
 1st Qu.:0.000   1st Qu.:1.000   1st Qu.:1.000   1st Qu.:0.000   1st Qu.:0.000
 Median :1.000   Median :1.000   Median :2.000   Median :1.000   Median :2.000
 Mean   :1.024   Mean   :1.484   Mean   :1.943   Mean   :1.397   Mean   :1.529
 3rd Qu.:2.000   3rd Qu.:2.000   3rd Qu.:3.000   3rd Qu.:2.000   3rd Qu.:3.000
 Max.   :3.000   Max.   :3.000   Max.   :3.000   Max.   :3.000   Max.   :3.000
```


Unfortunately, the "`summary()`" function in R does not include the standard deviation of the item responses. If the means and the standard deviations of items are desired, the "`apply()`" function can be useful for applying a specific function across rows or columns of a data set. The third argument is the operation, "`mean`" or "`sd`," for the mean or standard deviation of the data, respectively. The second argument indicates if the operation is applied across the rows, "`1`," or across the columns "`2`."

```
apply(ddd, 2, mean) # For the mean calculation across columns
apply(ddd, 2, sd) # For the standard deviation calculation across columns
```

#3.1_9. To check that all categories were utilized for each item, i.e., checking items for any unused category (null category), the "`apply()`" command can be used. The first argument is the data set, the second argument is "`2`" to request the operation across columns, and the third argument applies the "`table`" function to each column of the data. If a table for a single item is desired, "`table(ddd$I1)`" can be used, here to request the table by the item name, I1.

```
> apply(ddd, 2, table)
    V1  V2  V3  V4  V5
0  353 181  92 242 236
1  274 281 171 224 209
2  171 259 333 269 198
3  102 179 304 165 257
```

If any item's output does not display a frequency for all four categories, for this four-category example, the item and category may be problematic in the IRT analysis. All five items have no zero-frequency category in this application.

Before estimating an IRT model in R, we recommend that users rescore, or downcode, the data if null categories are found by removing the null category from the sequence of categories. Take, for example, an item that theoretically has five response categories: 0, 1, 2, 3, and 4. If categories 0 and 2 are never used, i.e., they are null categories, then the non-null categories of 1, 3, and 4 are rescored as 0, 1, and 2. Although this is a straightforward treatment of the null categories, this may not always be the most satisfactory solution. See Wilson and Masters (1993) for their proposal on handling null categories using a modified item parameterization for the PCM. Currently, no universally accepted approach for all polytomous response IRT models exists for the null categories. It is recommended that readers pay full attention in the early developments of a test to the item development, the optimal number of response options, or categories, and category descriptions for an item to mitigate the chance to observe this "null" category issue. When the null category is observed, readers may also be prompted to more closely evaluate the item and its categories for a better understanding of why there were no responses within a category.

Item parameter estimation

```
mod.pcm<-PCM(ddd, sum0 = F) #3.1_10
mod.pcm$conv #3.1_11
thresholds(mod.pcm)$threshtable$'1' #3.1_12
cbind(thresholds(mod.pcm)$threshpar, thresholds(mod.
 pcm)$se.thresh) #3.1_13
```

#3.1_10. The PCM is fit to the "ddd" data, and item parameters are estimated using the "`PCM()`" function. The first threshold of the first item ($b_{1(k=1)}$) is constrained to be 0 for the model identification constraint by specifying the option "`sum0 = F`." The estimation result is saved as an object called "`mod.pcm`."

The "eRm" uses the conditional maximum likelihood (CML) estimation. A popular model constraint of the Rash family model when the CML estimation method is to impose a constraint on the item parameters. As in the simple Rasch model, either the average of item parameters or the first item parameter of the first item is set to equal 0. For a straightforward understanding of the model identification imposed to the item parameters and the output, the authors prefer to use the constraint of the first threshold equal to 0 in the PCM application when using the "eRm" package. This allows the rest of the item parameters and person ability parameters to be estimated in reference to $b_{1(k=1)}$.

#3.1_11. The PCM convergence check result is displayed through "`mod.pcm$conv`." An output value of "`1`" indicates successful convergence, and no specific issue was detected under the default convergence criteria.

#3.1_12. The "`thresholds()`" command produces the parameter estimates of b_{ik}, which are the threshold parameter estimates and the overall item location, or difficulty. Specifying "`$threshtable$'1'`" provides the output as a matrix, rather than a list or other object type.

```
> thresholds(mod.pcm)$threshtable$'1'
       Location Threshold 1  Threshold 2  Threshold 3
I1   0.757580792   0.0000000   0.84016632    1.4325761
I2   0.062194359  -1.0091013   0.16114421    1.0345402
I3  -0.643925513  -1.5291272  -0.89319037    0.4905410
I4   0.241977890  -0.4118716  -0.05097397    1.1887793
I5   0.009683789  -0.4169371   0.10777690    0.3382115
```

The first numeric column is the overall "`Location`," or overall item difficulty; the remaining "`Threshold`" columns are the threshold estimates (e.g., $\hat{b}_{1(k=1)}$, $\hat{b}_{1(k=2)}$, and $\hat{b}_{1(k=3)}$ for item 1). The first threshold of the first item is fixed as 0 for the model identification. The overall item difficulty of an item is defined as the average of all threshold parameters. The first item overall difficulty, or "`Location`," is 0.7576, which is the average of the three threshold estimates (0.0000, 0.8402, and 1.4326).

If the first threshold of the first item happens to be the highest, then, because it is set to be 0, the other item thresholds will be estimated with a negative value. From the model-identification point of view, this is not a problem because the origin of the θ scale is arbitrary. However, some may find the negative item parameter estimates in all (or a majority of) items but the first item peculiar. In this case, the parameters could be rescaled by subtracting a constant from all item and person parameter estimates, which is the average of the overall location parameter estimates of all items. This sets the mean of the overall difficulty of items, or, simply speaking, the overall difficulty of the test, to 0.

For example, suppose that there is a three-item test and each item has four categories (0, 1, 2, and 3). Also, suppose that item threshold parameter estimates using the constraint of $b_{i(k=1)} = 0$ are 0, 0.4, 0.8 for the first item, −3.3, −2.3, −1.3 for the second item, and −2.2, −1.1, −0.9 for the third item. Then the overall item location, or difficulty parameter estimates, which are equal to the average of all threshold parameters, are 0.4, −2.3, and −1.4 for the first, second, and third items, respectively. To conduct the rescaling, the grand mean of the three overall item difficulty parameter estimates, −1.1, is subtracted from $\hat{b}_{ik}$. (Also, to place person ability on the same scale, $\hat{\theta}$, −1.1 is subtracted from all estimates.) These steps are illustrated here. This simple rescaling that sets the average of the overall item difficulties to be 0 may appeal more in some applications.

Typing "`thresholds(mod.pcm)`" or "`thresholds(mod.pcm)$threshtable`" will also provide the same output as #3.1_12, which contains the item location and deviation parameters (b_i and d_{ik}, respectively). However, the current command in #3.1_12 produces the results as a matrix, which allows further operations when necessary, while these shorter commands do not. For example, suppose the current output is saved as an R object labeled as "`rrr`" as shown.

```
rrr<-thresholds(mod.pcm)$threshtable$'1'
mean(rrr[,1])
rrr-mean(rrr[,1])
```

If the average difficulty of a test is sought, the location parameter values are extracted ("`rrr[,1]`") and the mean of them is calculated ("`mean(rrr[,1])`"). In this example, the mean is 0.0855. The aforementioned rescaling process that sets the overall difficulty of the test to 0 can be conducted with ease by implementing the third line as well, which subtracts the mean of all overall location parameters from the estimated item location and threshold parameters. The rescaled item parameter estimates are shown here.

```
> rrr-mean(rrr[,1])
       Location  Threshold 1  Threshold 2  Threshold 3
I1   0.67207853  -0.08550226   0.75466405    1.3470738
I2  -0.02330790  -1.09460355   0.07564195    0.9490379
I3  -0.72942778  -1.61462948  -0.97869263    0.4050388
I4   0.15647563  -0.49737391  -0.13647623    1.1032770
I5  -0.07581847  -0.50243933   0.02227463    0.2527093
```

#3.1_13. The output for this command shows both item threshold parameter estimates (in the first numeric column) and their standard errors (in the second numeric column).

```
> cbind(thresholds(mod.pcm)$threshpar, thresholds(mod.
  pcm)$se.thresh)
                          [,1]       [,2]
thresh beta I1.c1  0.00000000 0.0000000
thresh beta I1.c2  0.84016632 0.1576391
thresh beta I1.c3  1.43257606 0.1671750
thresh beta I2.c1 -1.00910129 0.1351391
thresh beta I2.c2  0.16114421 0.1264066
thresh beta I2.c3  1.03454016 0.1412999
...
thresh beta I5.c1 -0.41693707 0.1335477
thresh beta I5.c2  0.10777690 0.1356705
thresh beta I5.c3  0.33821154 0.1374057
```

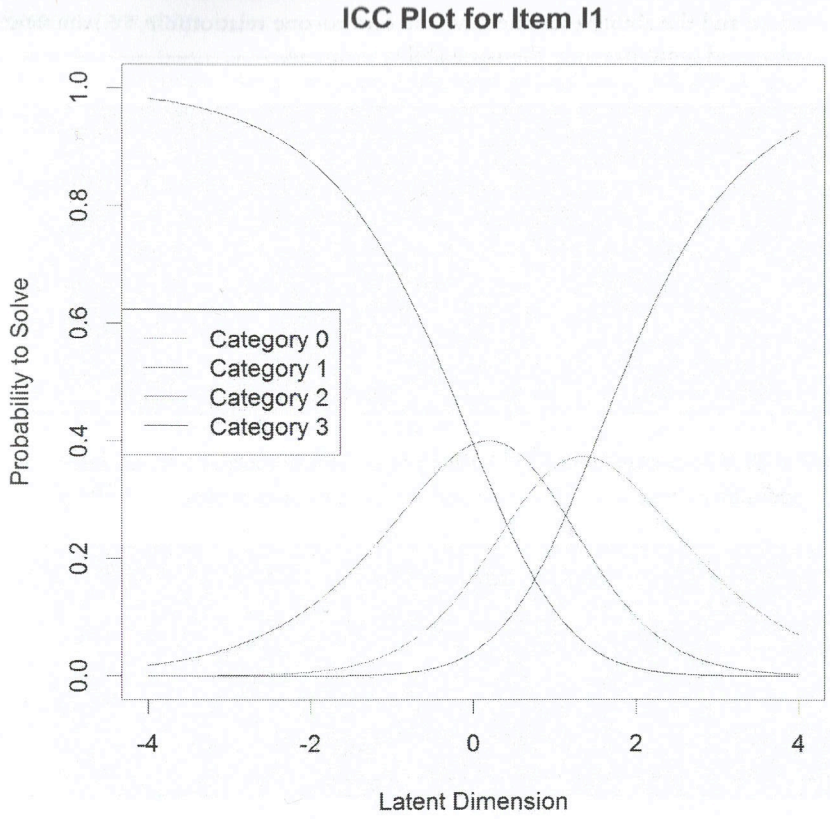
Typing "`summary(mod.pcm)`" also produces the estimation results; however, they are for the original model parameterization used in the estimation in the "eRm" package and do not follow the usual PCM parameterization.

Category characteristic curve (CCC) plot

In the dichotomous response modeling, the IRF was displayed graphically with an ICC. In the polytomous response modeling, the response function for a category is plotted using a category characteristic curve (CCC). Sometimes a CCC is also called a "trace line," although no universally accepted nomenclature exists.

```
plotICC(mod.pcm, 1) #3.1_14
```

#3.1_14. The CCC of a specified item can be requested using "`plotICC()`" with the desired item in the second argument. "`plotICC(mod.pcm)`" will provide a graphical user interface for all item CCCs, where users must press the return key to advance through the displays. A subset of items can be also selected by the command "`plotICC(mod.pcm, 1:3)`," for example, which provides the CCCs for items 1, 2, and 3. Unlike the simple Rasch case in the "eRm" package, there is no provision of empirical category characteristic curves compared with the model-based CCCs in the PCM application (shown in #2.1_18).



Person latent trait, or ability score estimation

```
p.pcm<-person.parameter(mod.pcm) #3.1_15
p.pcm #3.1_16
coef(p.pcm) #3.1_17
round(cbind(p.pcm$thetapar$NAgroup1, p.pcm$se.
  theta$NAgroup1), 3) #3.1_18
plot(p.pcm) #3.1_19
```

#3.1_15 and #3.1_16. The "`person.parameter()`" command provides the maximum likelihood (ML) estimates for person latent trait or ability scores. There were five items and each item response ranged from 0 and 3. The minimum observed sum score is 0 and the maximum observed sum score is 15 (5 × 3). For the perfect score and all zero score, no finite ML estimate is possible. An interpolation method is used to provide finite ability estimates for those zero and perfect scores, and "NA" is reported for the "`Std. Error.`" In the Rasch family model application, the observed