Tutorial for Chapter 7: Pendulum Interview Task

Tara L. Valladares & Moritz Heene

July 22, 2020

# Preliminary steps before you can run the pendulum interview analysis

1. Create a folder called “Pendulum\_analysis” on your desktop, for example. Download the data set ARMChapter7.csv and the R code Chapter\_7\_eRm.R from the website and save both in *that* folder. This folder will serve as your working directory containing all files you need to conduct the analysis and to store optional output (i.e., code, data, and figures). If the R code and the .csv files are not in the same folder, you will not be able to load the data using the code below.
2. Open the file Chapter\_7\_eRm.R in RStudio by clicking on the file. This will open the file in RStudio.
3. Go to “Session” “Set Working Directory” “To Source File Location”. This defines the folder you named above as your working directory in which you are currently working and where R expects all data sets to be.

You are now ready to run the analysis of the Pendulum data used in ARM4 Chapter 7. Please use the following instructions and explanations of the R code.

# Rasch Partial Credit Analysis: Data Preparation

First, we will load the necessary packages we will need for this analysis. If you have not yet, you will need to install the packages using the install.packages() function below. This will only need to be done once. Then, we will load the packages into R using library(). This will need to be done each time you open R.

# Installing the required packages for this analysis.   
# This will only need to be done once.  
# Uncomment the code below to run the install.packages function.  
# install.packages(c("eRm", "dplyr", "ggplot2", "pairwise", "TAM", "psych"))  
  
# Load the packages required for the analysis  
library(eRm)

Now we will read in the Pendulum data

# Reading in the comma-seperated data set  
df <- read.csv("ARMChapter7.csv", header = TRUE, sep = ";")

This data consists of the ratings of 58 interviews about solving the Piaget’s Pendulum problem expressed in terms of 18 qualitative items. The scoring criteria are given in ARM4 Table 7.2.The rating scale for the data is 0, 1, 2, 3. The four scoring categories for the students’ audio-recorded responses can be loosely described as:

1. “Not” = nothing relevant to solving the problem
2. “Action” = relevant action
3. “Logic” = relevant logic
4. “Correct” = totally correct solution

Like always, we will need to preprocess and format our data. We will create an object called pend\_items containing the Pendulum items we need for the analysis.

# Select the 18 columns of df where the Pendulum items are at  
pend\_items <- df[,2:19]  
# Looking at the first five rows of the Pendulum dataframe  
head(pend\_items)

## Order.lengths Order.weights Order.push Length.freq Vary.factors Infer  
## 1 1 1 1 1 1 2  
## 2 1 0 0 1 1 1  
## 3 1 1 1 1 1 1  
## 4 1 1 1 1 1 1  
## 5 1 1 1 1 1 1  
## 6 1 1 1 1 1 2  
## Tests.lengths Tests.weights Tests.push Length.role Weight.role Push.role  
## 1 2 2 1 1 1 0  
## 2 2 2 2 2 0 0  
## 3 2 2 2 2 2 2  
## 4 1 2 1 0 1 1  
## 5 1 2 1 2 2 0  
## 6 2 2 2 2 0 0  
## Length.combine Weight.combine Push.combine Systematic Exclude.weight  
## 1 1 2 2 0 0  
## 2 2 3 3 0 0  
## 3 1 3 2 0 0  
## 4 2 3 2 0 0  
## 5 1 1 0 0 0  
## 6 1 1 0 0 0  
## Exclude.push  
## 1 0  
## 2 0  
## 3 0  
## 4 0  
## 5 0  
## 6 0

Let’s take a look at what the names of the items are. This is stored in the column names of the dataset.

# Look at column names of the data  
colnames(pend\_items)

## [1] "Order.lengths" "Order.weights" "Order.push" "Length.freq"   
## [5] "Vary.factors" "Infer" "Tests.lengths" "Tests.weights"   
## [9] "Tests.push" "Length.role" "Weight.role" "Push.role"   
## [13] "Length.combine" "Weight.combine" "Push.combine" "Systematic"   
## [17] "Exclude.weight" "Exclude.push"

We are now ready to run the actual Partial Credit model analyses. In accordance with the corresponding Winsteps analysis, we will

* Fit a Rasch Partial Credit Model for the Pendulum data
* Examine the item thresholds
* Look at pathway and Wright map plots

# Fitting a Partial Credit Model

The Pendulum data is not dichotomous (either a 0 or a 1). Since it has multiple categories, this kind of data is called polytomous data. We cannot use the RM function for polytomous data, instead we will use the PCM function which will fit a Rasch Partial Credit Model.

# Fit a Partial Credit Model for the BLOT data  
fit\_pcm\_pend <- PCM(pend\_items)

## Warning:   
## The following items have no 0-responses:  
## Order.lengths  
## Responses are shifted such that lowest category is 0.

## Warning:   
## The following items were excluded due to complete 0/full responses:  
## Order.lengths

## Warning:   
## The following items do not have responses on each category:  
## Tests.weights  
## Estimation may not be feasible. Please check data matrix!

We got a few warnings from running the PCM. You should always pay attention to warnings in R and determine if there is a problem. Let’s break each of them down and discuss what to do.

First, “*The following items have no 0-responses: Order.lengths. Responses are shifted such that lowest category is 0.*”. This warning is indicating that the item “Order.lengths” only has responses 1, 2, and 3. No one responded “0” to this item. This tell us two things - 1) The “0” response to this item was too easy for this group, and - 2) When we interpret our results, we need to keep in mind that “Order.lengths” has one less category and threshold than the other items. But, the real solution is in the second warning, “*The following items were excluded due to complete 0/full responses: Order.lengths*”. This warning tell us that everyone had the same response to this item. If everyone has the same response, *eRm* cannot estimate the item parameters. This item was dropped from the output; we will only have 17 items in the model. You can verify this issue by looking directly at the item.

# Look at Order.length's responses  
pend\_items$Order.lengths

## [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1  
## [39] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

Finally, we have the error “*The following items do not have responses on each category: Tests.weights. Estimation may not be feasible. Please check data matrix!*”. “Test.weights” seems to have a similar problem as “Order.lengths” - there are no responses for certain categories. Let’s look directly at the responses to see what is going on, like the warning suggests (“*Please check data matrix*”).

# Look at Test.weight's responses  
pend\_items$Tests.weights

## [1] 2 2 2 2 2 2 2 2 2 2 2 0 2 2 2 2 0 2 0 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2  
## [39] 2 2 2 2 2 2 2 2 2 0 2 2 2 2 2 2 2 2 2 2

# Look at a table of the responses  
table(pend\_items$Tests.weights)

##   
## 0 2   
## 4 54

For “Test.weights”, four people received a “0” and fifty four people received a “2”. Without knowing how the data was collected, we don’t know if this is a typo, if the question is actually dichotomous, or if people really only received a “0” or “2”. In fact, let’s go ahead and check what the rest of the data looks like. We’ll use the sapply function to run the table function on each column of pend\_items. This will give us how many people responded to each category per item.

sapply(X = pend\_items, FUN = table)

## $Order.lengths  
##   
## 1   
## 58   
##   
## $Order.weights  
##   
## 0 1   
## 2 56   
##   
## $Order.push  
##   
## 0 1   
## 2 56   
##   
## $Length.freq  
##   
## 0 1   
## 2 56   
##   
## $Vary.factors  
##   
## 0 1   
## 2 56   
##   
## $Infer  
##   
## 0 1 2   
## 1 18 39   
##   
## $Tests.lengths  
##   
## 0 1 2   
## 5 21 32   
##   
## $Tests.weights  
##   
## 0 2   
## 4 54   
##   
## $Tests.push  
##   
## 0 1 2   
## 9 14 35   
##   
## $Length.role  
##   
## 0 1 2 3   
## 2 27 23 6   
##   
## $Weight.role  
##   
## 0 1 2 3   
## 5 33 8 12   
##   
## $Push.role  
##   
## 0 1 2 3   
## 13 26 13 6   
##   
## $Length.combine  
##   
## 0 1 2   
## 2 31 25   
##   
## $Weight.combine  
##   
## 0 1 2 3   
## 3 25 1 29   
##   
## $Push.combine  
##   
## 0 1 2 3   
## 26 5 15 12   
##   
## $Systematic  
##   
## 0 1   
## 33 25   
##   
## $Exclude.weight  
##   
## 0 1   
## 44 14   
##   
## $Exclude.push  
##   
## 0 1   
## 50 8

We can see that several of our items are dichotomous, and others are polytomous. We should keep the issues with these items in mind as we go forward.

In any case, let’s continue on with the analysis. We’ll take a look at the item parameters for the Pendulum test. Similar to before, we’ll first estimate the person parameters using the person.paramter function, then run the output through the itemfit function and examine the item statistics using print.

# Estimating person parameters for each observed raw score.   
pparameters\_pend <- person.parameter(fit\_pcm\_pend)   
# Estimate item parameters  
item\_fit\_pend <- itemfit(pparameters\_pend)

## Warning in cor.smooth(R): Matrix was not positive definite, smoothing was done

## In smc, smcs < 0 were set to .0

print(item\_fit\_pend)

##   
## Itemfit Statistics:   
## Chisq df p-value Outfit MSQ Infit MSQ Outfit t Infit t Discrim  
## Order.weights 58.970 57 0.403 1.017 1.000 0.353 0.205 0.122  
## Order.push 58.969 57 0.403 1.017 1.000 0.353 0.205 0.122  
## Length.freq 20.802 57 1.000 0.359 0.875 -0.525 -0.009 0.349  
## Vary.factors 17.213 57 1.000 0.297 0.835 -0.654 -0.082 0.378  
## Infer 51.445 57 0.683 0.887 0.944 -0.398 -0.252 0.419  
## Tests.lengths 61.553 57 0.316 1.061 1.114 0.347 0.694 0.203  
## Tests.weights 436.769 57 0.000 7.531 1.041 1.971 0.233 0.088  
## Tests.push 48.170 57 0.791 0.831 0.855 -0.568 -0.829 0.488  
## Length.role 45.792 57 0.856 0.790 0.789 -1.202 -1.246 0.607  
## Weight.role 55.980 57 0.513 0.965 0.926 -0.088 -0.395 0.522  
## Push.role 47.276 57 0.817 0.815 0.807 -1.031 -1.147 0.659  
## Length.combine 47.208 57 0.819 0.814 0.836 -1.122 -1.078 0.393  
## Weight.combine 63.205 57 0.267 1.090 1.035 0.346 0.275 0.412  
## Push.combine 83.658 57 0.012 1.442 1.231 1.518 1.348 0.354  
## Systematic 49.563 57 0.747 0.855 0.877 -1.001 -1.323 0.535  
## Exclude.weight 37.860 57 0.976 0.653 0.776 -1.501 -1.425 0.582  
## Exclude.push 33.528 57 0.994 0.578 0.787 -1.127 -0.793 0.516

You can see that even though this is a Partial Credit model with polytomous responses, we still have the same kind of fit statistics for the items. We have misfit based on the chi-square statistic for “Push.combine” and “Tests.weights”. We already know “Tests.weights” is a problem. Several of our items have suspicious outfit statistics, so let’s sort by that using the sort\_by = "outfit\_MSQ" argument of the print function.

print(item\_fit\_pend, sort\_by = "outfit\_MSQ")

##   
## Itemfit Statistics:   
## Chisq df p-value Outfit MSQ Infit MSQ Outfit t Infit t Discrim  
## Vary.factors 17.213 57 1.000 0.297 0.835 -0.654 -0.082 0.378  
## Length.freq 20.802 57 1.000 0.359 0.875 -0.525 -0.009 0.349  
## Exclude.push 33.528 57 0.994 0.578 0.787 -1.127 -0.793 0.516  
## Exclude.weight 37.860 57 0.976 0.653 0.776 -1.501 -1.425 0.582  
## Length.role 45.792 57 0.856 0.790 0.789 -1.202 -1.246 0.607  
## Length.combine 47.208 57 0.819 0.814 0.836 -1.122 -1.078 0.393  
## Push.role 47.276 57 0.817 0.815 0.807 -1.031 -1.147 0.659  
## Tests.push 48.170 57 0.791 0.831 0.855 -0.568 -0.829 0.488  
## Systematic 49.563 57 0.747 0.855 0.877 -1.001 -1.323 0.535  
## Infer 51.445 57 0.683 0.887 0.944 -0.398 -0.252 0.419  
## Weight.role 55.980 57 0.513 0.965 0.926 -0.088 -0.395 0.522  
## Order.weights 58.970 57 0.403 1.017 1.000 0.353 0.205 0.122  
## Order.push 58.969 57 0.403 1.017 1.000 0.353 0.205 0.122  
## Tests.lengths 61.553 57 0.316 1.061 1.114 0.347 0.694 0.203  
## Weight.combine 63.205 57 0.267 1.090 1.035 0.346 0.275 0.412  
## Push.combine 83.658 57 0.012 1.442 1.231 1.518 1.348 0.354  
## Tests.weights 436.769 57 0.000 7.531 1.041 1.971 0.233 0.088

We can see that the items “Vary.factors”, “Length.freq”, “Exclude.push”, and “Exclude.weight” have low outfit MSQ values, while “Test.weights” again has a super high outfit MSQ.

We can also look at the thresholds for the items. The thresholds are the “Rasch-Thurstone Thresholds”, the points at which there is a 50% probability of being observed in categories below and 50% being observed in categories at or above the category transition point. So for “Length.role”, Threshold 1 is equal to -2.44. A person with an ability score of -2.44 has a 50% chance to receive either “0” or “1” on this item. Threshold 2 for “Length.role” is at 0.75. A person with an ability score of 0.75 has a 50% chance to receive either “1” or “2” on this item.

This table also shows the item locations, which is the overall difficulty of the items, alongside the thresholds. Notice that not all items have three thresholds. The polytomous items have as many thresholds as one minus the number of categories in that item. For the dichotomous items, their Location equals the value of Threshold 1 because they only have two categories. We can see that “Order.weights” was a dichotomous item, and “Infer” was an item with three different response categories. We can also see that “Tests.weights” is a crazy item still - it’s thresholds are at +16 and -18 (those are extreme values!) even though it only has two category responses.

thresholds(fit\_pcm\_pend)

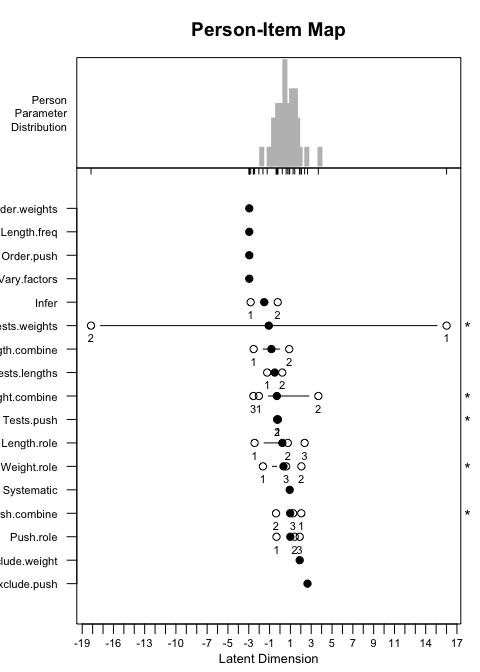
##   
## Design Matrix Block 1:  
## Location Threshold 1 Threshold 2 Threshold 3  
## Order.weights -2.95083 -2.95083 NA NA  
## Order.push -2.95081 -2.95081 NA NA  
## Length.freq -2.95081 -2.95081 NA NA  
## Vary.factors -2.95081 -2.95081 NA NA  
## Infer -1.51682 -2.81479 -0.21886 NA  
## Tests.lengths -0.50645 -1.23162 0.21872 NA  
## Tests.weights -1.07510 16.00460 -18.15481 NA  
## Tests.push -0.23731 -0.20487 -0.26974 NA  
## Length.role 0.22690 -2.43984 0.75230 2.36822  
## Weight.role 0.33636 -1.63846 2.05632 0.59124  
## Push.role 0.99240 -0.33462 1.41829 1.89352  
## Length.combine -0.82282 -2.53058 0.88494 NA  
## Weight.combine -0.30089 -2.02985 3.68540 -2.55822  
## Push.combine 0.97594 2.04541 -0.37807 1.26047  
## Systematic 0.93234 0.93234 NA NA  
## Exclude.weight 1.89161 1.89161 NA NA  
## Exclude.push 2.64189 2.64189 NA NA

# Plots

## Wright Map

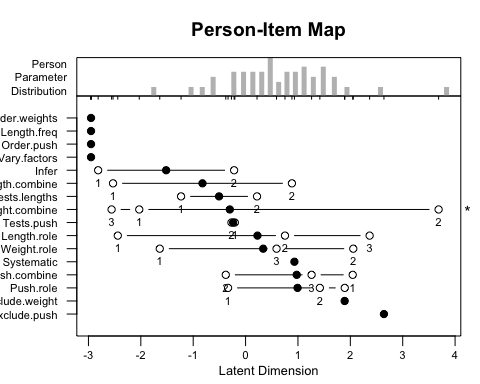
Now let’s look at the Wright map for this model. If you recall for the dichotomous Rasch model, the Wright map plots the difficulties of the items against the distribution of the person abilities. The same is true for the polytomous Partial Credit model, but with some modifications. The distribution on the top is still the distribution of person measures. The x-axis is still the latent dimension, and the y-axis still corresponds to each item. Now, each item now gets a black dot that represents the item location. The white dots represent the location of the thresholds. If an item is dichotomous, it just has the black location dot. Each item’s dots are connected by a line to make the graph easier to read.

# Wright map  
plotPImap(fit\_pcm\_pend, irug = TRUE, sort = TRUE)



The first thing you’ll likely notice is the crazy parameters of “Tests.weights.” That’s a bit distracting, so we’re going to run the map again, but this time we’ll pick which items we want to see using the argument item.subset and excluding column number 7.

# Run the Wright map without "Test.weights"  
plotPImap(fit\_pcm\_pend, irug = TRUE, sort = TRUE,  
 item.subset = c(1:6,8:17))



Now this map is a lot easier to read. We can see that four of our items (“Order.weights”, “Length.freq”, “Order.push”, and “Vary.factors”) are probably too easy for this group of participants. We can also see that the item “Infer”, had 3 category responses because it has two thresholds (white dots). “Weight.combine” has something strange going on - the categories are not ordered how we expected! Threshold 3 (between a response of “2” and “3”) is easier than Threshold 1 (between a response of “0” and “1”). This kind of category ordering can be easier to see on the map rather than by using the previous threshold table.

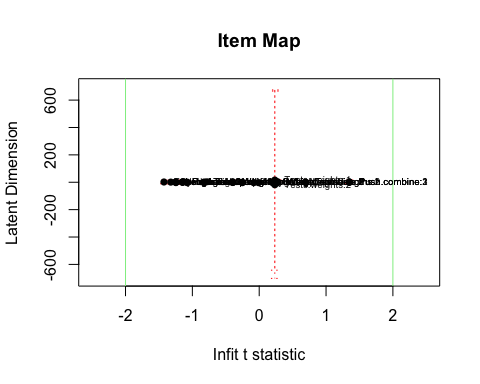
## Pathway Map

Now let’s look at the pathway map for this model.

# Pathway map with 95% confidence intervals for the item parameters  
plotPWmap(fit\_pcm\_pend, itemCI = list(gamma=0.95,col="red",lty="dotted"))

## Warning in cor.smooth(R): Matrix was not positive definite, smoothing was done

## In smc, smcs < 0 were set to .0



Uh oh, that map totally does not look right at all! You’ll notice the warning message, “*Warning message: In cor.smooth(R) : Matrix was not positive definite, smoothing was done*”. This is indicating that something is seriously wrong with our data. It is likely due to the messed up item “Test.weights” that has been giving us so much trouble! Let’s try to do the pathway map again, but this time we are going to re-estimate our model to not include the “Test.weights” item. We’ll then run the pathway map function plotPWmap on this new model.

# Re-estimate the model without the eighth item ("Test.weights")  
new\_fit\_pcm\_pend <- PCM(pend\_items[,-c(8)])

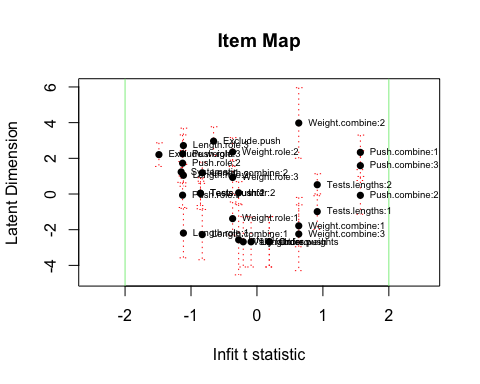
## Warning:   
## The following items have no 0-responses:  
## Order.lengths  
## Responses are shifted such that lowest category is 0.

## Warning:   
## The following items were excluded due to complete 0/full responses:  
## Order.lengths

# Run the pathway map on the new model  
plotPWmap(new\_fit\_pcm\_pend, itemCI = list())

## Warning in cor.smooth(R): Matrix was not positive definite, smoothing was done

## In smc, smcs < 0 were set to .0



That map looks much better now. When we run the pathway map on a polytomous model, each dot represents a threshold of an item instead of a whole item. The y-axis is still the latent dimension and the x-axis is still the infit t statistic. We can see that all the remaining thresholds are within our -2 to +2 boundary, which is great.