Rasch Analysis in R

Psychologische Institut für psychologische Methodenlehre,
Evaluation und Statistik

Carolina Fellinghauer, PhD, Dpl.Stat.
carolina.fellinghauer@uzh.ch

HS 2020

Introduction

Willkommen! Welcome!

Welcome to the Seminar about Rasch analysis with R!

About me

- Post Doc Assistent at the University of Zürich.
- External Consultant for WHO and World Bank: Disability surveys, Rasch analyses.
- PhD in Health Sciences (Uni Luzern, CH).
- Diploma in Statistics (Uni Neuchâtel, CH).
- Master in Clinical Psychology (Uni Fribourg, CH).

Motivation

- Data collections often include questionnaires on a topic (satisfaction, quality of life, health, knowledge)
- Derived statistical analysis may prefer a parsimonious score than using single questionnaire items
- With ordinal responses (1=do no agree, 2=agree somewhat, 3=totally agree) creating sum scores and averages is wrong
- Solution: Rasch analysis

Objectives of the Seminar

After this seminar, you should be able to:

1. Describe the reasoning and process behind Rasch Analysis
2. Chose and run the appropriate Rasch model
3. Test the different assumptions of the Rasch model
4. Assess the quality of the Rasch model
5. Make informed decisions based on Rasch analysis outputs
6. Adjust the data to improve the quality of the Rasch model
7. Derive Rasch scores and Rasch score transformation tables

Procedure

- Fall semester 2020, weekly course (14 weeks including an exam)
  - 1st part: Presentation of the theory (45 min)
  - 2nd part: Individual or group R-exercises for the application of theory (45 min).
- Room:
  - Virtual: MS-Team *RaschSeminar*
  - Real world: *AND-2-46*
- Dates:
  - 16.09.2020 - 16.12.2020
  - Wednesdays 16:15- 18:00

Materials

- Update course materials are made available weekly via OLAT. Course files can also be found on GitHub and consist mainly of:
    - Slides (Power Point, Beamer) or HTML workbook for the 1st part.
    - R-Codes for exercises and solutions for the 2nd part.
    - Additional Literature
- Seminar teaching language German/English.
- Course material English.
- Today's Special:
    - Getting Ready with R-Studio and R
    - Brief overview to R Syntax

R and RStudio

- R and RStudio are downloaded and installed
- RStudio is a friendly user-interface for R

Think of a car...

R - Engine

RStudio - Dashboard
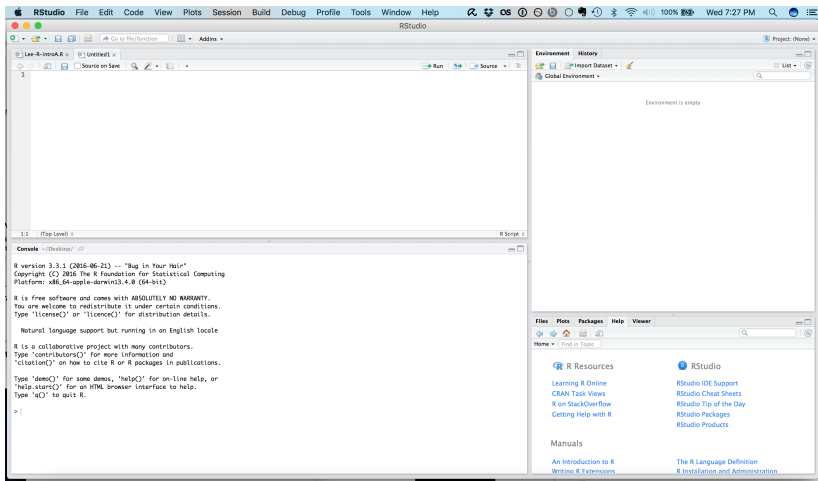
R - Don't open this

RStudio - Open this

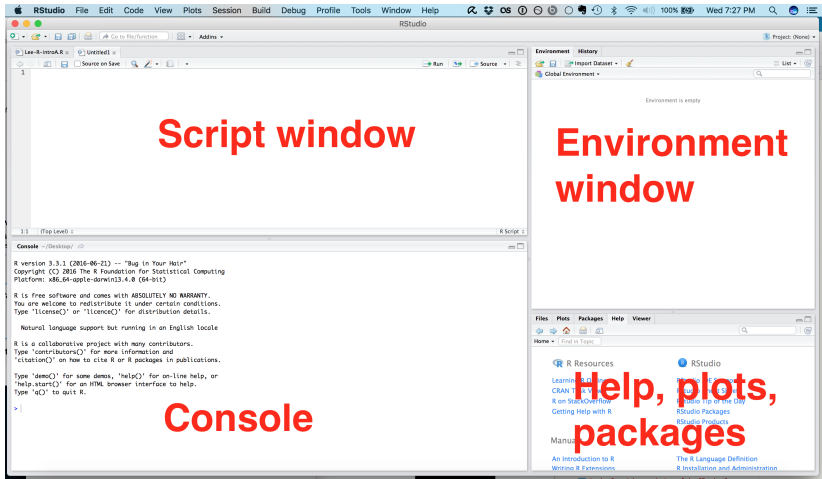# Opening RStudio - what are these windows?

- Open the program, and you should see the this window

RStudio windows

Script window

Environment window

Console

Help, plots, packages

RStudio windows

- **Console**: where you can type commands and see output
- **Script window**: where you type and edit longer codes and functions, and save them
- **Environment window**: contains lists of variables that you have created or loaded
- **Help, plots, packages**: where plots you create, help pages, and the list of available R packages appear

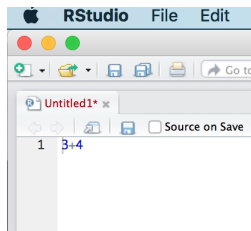Run a command via the console by typing the following after the >
and pressing `Enter`:

```
3+4
```

```
## [1] 7
```

Run code from a **script**

- Open a new script by typing `Ctrl+Shift+N` or going to File $->$ New File $->$ R Script
- Type 3+4 into the new script window
- Run the line of code by highlighting the line and clicking "Run" or typing `Ctrl+Enter`

Save and set working directory

- Save the script by clicking the Save button or typing `Ctrl+S`.
- The **working directory** is the folder that `R` is currently paying attention to and will automatically save materials in, unless otherwise told.
- We want to set the working directory to a folder where the script is: `setwd()` to source file location. Example: `setwd("C:/Users/carol/Documents/Work/UniZH")`.
- We can also set a path:
  `path<- "C:/Users/carol/Documents/Work/UniZH"`

Getting help with a function

```
?solve
?"*"
?trimws
```

Parts of a help page

Remove Leading/Trailing Whitespace

Description

Remove leading and/or trailing whitespace from character strings.

Usage

```
trimws(x, which = c("both", "left", "right"), whitespace = "[ \t\r\n]")
```

Arguments

x
> a character vector

which
> a character string specifying whether to remove both leading and trailing whitespace (default), or only leading (`"left"`) or trailing (`"right"`). Can be abbreviated.

whitespace
> a string specifying a regular expression to match (one character of) "white space", see Details for alternatives to the default.

Details

Internally, sub(re, "", *, perl = TRUE), i.e., PCRE library regular expressions are used. For portability, the default 'whitespace' is the character class [\t\r\n] (space, horizontal tab, carriage return, newline). Alternatively, [\h\v] is a good (PCRE) generalization to match all Unicode horizontal and vertical white space characters, see also https://www.pcre.org.

Examples

```
x <- "  Some text. "
x
trimws(x)
trimws(x, "l")
trimws(x, "r")

## Unicode --> need "stronger" 'whitespace' to match all :
tt <- "text with unicode 'non breakable space'."
xu <- paste(" \t\v", tt, "\u00a0 \n\r")
(tu <- trimws(xu, whitespace = "[\\h\\v]"))
stopifnot(identical(tu, tt))
```

Using packages

```
?RM
??RM
```

Using packages

```
library(eRm)
?RM

library(fun)
install.packages("fun")
library(fun)
```

Think of a smartphone...

R - A new phone

R packages - Apps you can download

Interpreting the console

- > - R is ready for you to execute a code

```
Type 'demo()' for some demos, help() for on-li
'help.start()' for an HTML browser interface to
Type 'q()' to quit R.

>
```
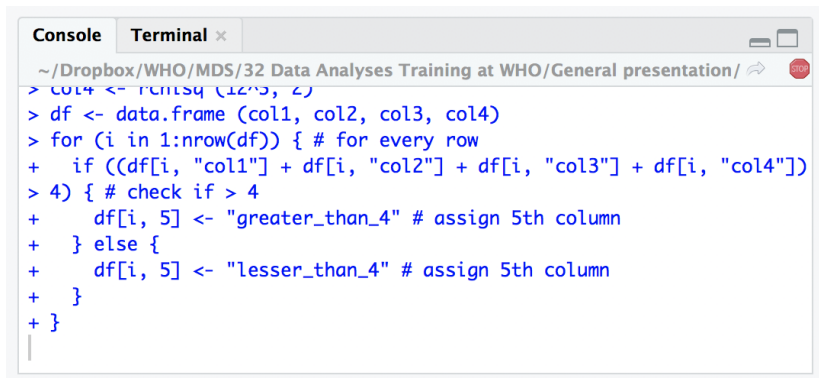
Interpreting the console

- \+ - R is waiting for you to finish something you have started
- If you see this, you have perhaps written a {, (, or " and did not close off the statement (},), ")

```
help.start()   for an HTML browser interface to help.
Type 'q()' to quit R.

> "hello my name is
+ |
```

Interpreting the console

- no symbol - `R` is currently executing some code
- You can terminate the code by clicking on the stop sign or pressing `Esc`.

```
Console    Terminal ×

~/Dropbox/WHO/MDS/32 Data Analyses Training at WHO/General presentation/
> col4 <- rchisq (12^3, 2)
> df <- data.frame (col1, col2, col3, col4)
> for (i in 1:nrow(df)) { # for every row
+   if ((df[i, "col1"] + df[i, "col2"] + df[i, "col3"] + df[i, "col4"])
> 4) { # check if > 4
+     df[i, 5] <- "greater_than_4" # assign 5th column
+   } else {
+     df[i, 5] <- "lesser_than_4" # assign 5th column
+   }
+ }
```

Important properties of R

- `R` is cAsE-sEnSiTiVe
    - `xxx` is different to `XXX` is different to `xXx`
- Spaces generally don't matter (though if used well can make your code easier to read)
    - A vector created with `c(10,2,3)` is the same as one created with `c(10,    2  , 3)`
    - Exceptions are within the names of objects and within strings.
        - `MickeyMouse` is one object, while `Mickey Mouse` is two.
        - The string `"Mickey Mouse"` is not the same as the string `"Mickey          Mouse"`

Vectors and assignment

```
x <- c(10.4, 5.6, 3.1, 6.4, 21.7)
y <- c(x, 0, x)
```

Vector arithmetic

```
2 + 3
4 - 5
x + x
x * x
```

Vector arithmetic

```
var(x)
sd(x)
sqrt(x)
x^2
```

Sorting and Ordering

```
sort(x)
order(x)
x[order(x)]
```

Generating regular sequences

```
1:10
10:1
seq(from=10,to=1,by=-1)
seq(10,1,-1)
seq(1,2,0.1)
seq(length=11,from=1,to=2)
rep(x,times=5)
rep(x,each=5)
```

Logical vectors

```
5 < -4
5 <= 5
x > 13
temp <- x > 13
temp
!temp
c1 <- 1:4
c2 <- 4:7
c1 %in% c2
```

Character vectors

```r
"Hello!"
words <- c("Hello!", "My name is", "Carolina")
paste(words, sep=" ")
paste(words, collapse = " ")
sentence=paste(words, collapse = " ")
nchar(sentence)
substr(sentence, 1, 6)
substr(sentence, 19, nchar(sentence))
```

Index vectors

```r
z <- c(10:12,NA)
z[1]
z[2:4]
z[rep(2:3,each=4)]
z[10]
```

Index vectors

```
is.na(z)
!is.na(z)
z[!is.na(z)]
z[-1]
z[-c(1,3)]
```

```
fruit <- c(5, 10, 1, 20)
names(fruit) <- c("orange", "banana",
                  "apple", "peach")
fruit
lunch <- fruit[c("apple","orange")]
lunch
```

```
z <- 0:9
z
mode(z)
length(z)
digits <- as.character(z)
digits
mode(digits)
length(digits)
d <- as.integer(z)
identical(z,d)
```

The class of an object

```
class(fruit)
class(names(fruit))
class(1:5)
mode(1:5)
class(cars)
cars
unclass(cars)
```

A specific example

```r
canton <- c("ZH", "FR", "GE", "VD", "NE")
cantonf <- factor(canton)
cantonf
levels(cantonf)
```

Ordered factors

```
ranking <- c(1,3,2,1,3)
rankingf <- factor(ranking,ordered=TRUE)
rankingf
class(rankingf)
rankingf <- ordered(ranking)
rankingf
class(rankingf)
```

Arrays and matrices

Arrays

```
z <- numeric(30)
matrix(0,nrow=3,ncol=10)
array(0,dim = c(3,2,5))
a <- matrix(1:25,5,5)
b <- matrix(1:25,5,5,byrow=TRUE)
c <- array(1:100,dim=c(5,5,4))
dim(c)
dim(fruit)
```

Array indexing

```
a[1,1]
a[3,4]
a[2:3,]
a[,5]
c[,,4]
c[,2,3]
```

Index matrices

```r
tea <- sample(c("black","green"),10,replace=TRUE)
coffee <- sample(c("with sugar", "with milk"),10,
                 replace=TRUE)
table(tea)
table(coffee)

table(tea,coffee)
rows <- rownames(table(tea,coffee))
which(rows=="black")
```

setting a seed

```r
sample(c("black","green"),10,replace=TRUE)

set.seed(1234)
sample(c("black","green"),10,replace=TRUE)
```

Forming partitioned matrices

```
cbind(1:3,4:6)
rbind(1:3,4:6)
cbind(1:2,4:6)
```

Saving and Reading Data

```r
df_write <- data.frame(
  sex=sample(c("M","F"), 100, replace=TRUE),
  age=sample(18:100,100, replace=TRUE),
  score=runif(100,0,100))
write.csv(df_write,"sampledata_wd.csv", row.names=FALSE)

write.csv(df_write,
          paste(path, "sampledata_path.csv", sep=""),
          row.names=FALSE)
```

read.csv()

```r
df_read_wd <- read.csv("sampledata_wd.csv")
df_read_path <- read.csv(
        paste(path, "sampledata_path.csv", sep=""))

head(df_read_wd)
df_read2_wd <- read.csv("sampledata_wd.csv",
                        header=FALSE)
head(df_read2)
```

read.csv() from url

The data can be read directly with library(readr) from the Github, where course materials are found. To do that:

1. Go to the github repository link where you have the CSV file
2. Click on the *raw* option present on the top right of the data
3. This will open a new window in the browser
4. You have to copy-paste this link to download csv file from the Github to the R-syntax file

read.csv() from url

```r
library(readr)

#example of link found for the SRG.data.
urlfile = "https://raw.githubusercontent.com/
           CarolinaFellinghauer/
           UNIZH_HS2020_Rasch/master/
           SRG_Data_Course_UNIZH.csv?
           token=AB5GB44FKMEBAHFLVMA7X727GUGR6"

srg.data=read.csv(url(urlfile))
dim(srg.data)
```

Lists and data frames

Lists

```
Lst <- list(name="Fred", wife="Mary", no.children=3,
            child.ages=c(4,7,9))
Lst[[1]]
Lst[[4]]
Lst[[4]][2]
Lst[["child.ages"]]
Lst$child.ages
Lst[4]
class(Lst[4])
class(Lst[[4]])
```

Making data frames

```
cantonf <- factor(c("ZH", "FR", "GE", "VD", "NE"))
incomes <- c(60,68,80,52,90)
incomef <- cut(incomes,
               breaks=seq(from=50,to=90,by=10))
data_canton <- data.frame(home=cantonf, total=incomes,
                   group=incomef)
data_canton$home
data_canton[,1]
data_canton[1,]
A <- matrix(1:6,2,3)
as.data.frame(A)
```

Filtering and selecting (old school)

```r
data_canton[,c("home", "group")]
data_canton[which(data_canton$total > 70), ]
data_canton[which(data_canton$total > 70
                  & data_canton$home == "GE"),]
```

Filtering and selecting (fancy)

```r
library(dplyr)
dplyr::select(data_canton, home, group)
dplyr::filter(data_canton, total > 70)
dplyr::filter(data_canton,
        total > 70, home == "GE")
```

Applying functions

R-packages will contain the functions to run the Rasch analysis (e.g. eRm, iarm, mirt). Some additional programming skills are required for data preparation and output selective extraction and saving.

The help pages show how to use a function, e.g. ?PCM

To find a function, search through the package, google, course materials. . . .

In that analysis the function is often stored as an object PCM_model <- PCM(data) to keep the output and use it further Result <- summary(PCM_model)

RStudio tips

More about the windows:

- **Environment**: List of all objects in the environment
- **History**: List of all commands you have sent to the console
- **Files**: List of files available in your working directory
- **Plots**: Window for plots
- **Packages**: List of all packages available
- **Help**: Window for help pages
- **Viewer**: Window for viewing local web content

RStudio shortcuts

Action	Windows	Mac
Search command history	Ctrl+UpArrow	Cmd+UpArrow
Interrupt current command	Esc	Esc
Fold selected	Alt+L	Cmd+Opt+L
Unfold selected	Shift+Alt+L	Cmd+Shift+Opt+L
Attempt completion	Tab	Tab
Insert <-	Alt + -	Opt + -
Comment/uncomment line	Ctrl+Shift+C	Cmd+Shift+C
Reflow comment	Ctrl+Shift+/	Cmd+Shift+/
Reindent lines	Ctrl+I	Cmd+I
Reformat selection	Ctrl+Shift+A	Cmd+Shift+A

Many more available. . . go to Tools − > Keyboard Shortcuts Help

Error messages

- One of the most frustrating things is getting an error message and not knowing what it is about!
- Often it's better that you are getting an error message than if your program "fails silently"
- It is completely normal to struggle with R in the beginning. . .
- . . . and really the struggle never ends!

When you get an error

1. Read the error message, try to understand what it means
2. Try to isolate what exactly is causing the problem (debug!)
   - Rerun the code with different inputs
   - Try to isolate pieces of functions
   - Use RStudio's debugging features
3. Google it.
   - Google will often lead you to a site called "StackOverflow", where people post programming questions and other people answer them. I answer about 90% of my R problems this way

Thank you!!

References

R:

- "An Introduction to R" by W. N. Venables, D. M. Smith and the R Core Team
    - click on "Contributed" under "*Documentation*" in the left menu: ▸ CRAN
- ▸ "An Introduction to Statistical Data Sciences via R" by C. Ismay and A. Y. Kim
- ▸ "Advanced R" by H. Wickham
- "Introductory Statistics with R" by P. Dalgaard
- ▸ "R for Data Science" by G. Grolemund and H. Wickham
- ▸ R-Bloggers site on functions
- ▸ R-Bloggers site on conditional execution