Splits & Iterators

Adam Garber

Norwegian University of Science and Technology - A Course in `MplusAutomation`

May 30, 2021

Outline

1. Randomly split data into 2 equal parts (calibration & validation samples)
2. Introduction to using iterators or loops with `mplusObject()`

Getting started - following the routine...

1. Create an R-Project
2. Install packages (ONLY IF NEEDED)
3. Load packages

Folder structure:

Parent folder: `03-efa`

Nested folders:

- `data`
- `efa_mplus`
- `figures`

New packages:

- {`janitor`}

Begin

DATA SOURCE: This lab exercise utilizes the NCES public-use dataset: Education Longitudinal Study of 2002 (Lauff & Ingels, 2014) See website: nces.ed.gov

---

loading packages...

```
library(janitor)
library(tidyverse)
library(haven)
library(MplusAutomation)
library(here)
library(corrplot)
library(glue)
```

Change starting location to folder `04-splits-iterators`

```
source("rep_functions.R")

change_here(glue("{project_location}/04-splits-iterators"))

here()
```

```
## [1] "/Users/agarber/github/NTNU-workshop/04-splits-iterators"
```

read in the raw dataset

```
lab_data <- read_spss("https://garberadamc.github.io/project-site/data/els_sub1_spss.sav")
```

create a subset of the dataset called `school_trouble`

```
school_trouble <- lab_data %>%
  select(41:55)
```

make a new codebook from the `school_trouble` subset

```
sjPlot::view_df(school_trouble)
```

write a CSV datafile

```
write_csv(school_trouble, here("data", "school_trouble_data.csv"))
```

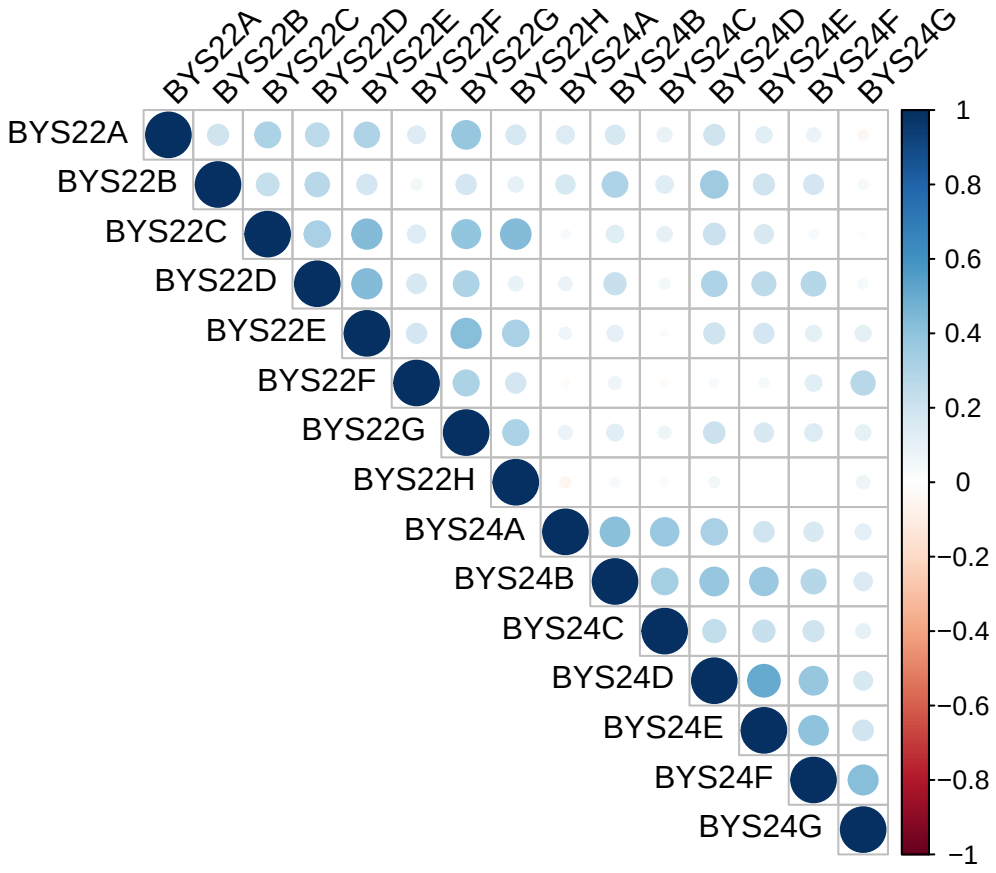
read the unlabeled data back into R

```
trouble_data <- read_csv(here("data", "school_trouble_data.csv"))
```

check items to see if reverse coding is needed

```
cor_matrix <- cor(trouble_data, use = "pairwise.complete.obs")

corrplot(cor_matrix, method="circle",
         type = "upper",
         tl.col="black",
         tl.srt=45)
```



Randomly split a sample into 2 equal parts

- Get n-size of half of original sample using `nrow()`
- The `floor()` function helps with rounding

```
smp_size <- floor(0.50 * nrow(trouble_data))
```

set the seed to make your partition reproducible

```
set.seed(123)
```

the function `sample()` will pick at random the values of the specified number

```
calibrate_smp <- sample(seq_len(nrow(trouble_data)), size = smp_size)
```

create two samples called "calibrate" & "validate"

```
calibrate <- trouble_data[calibrate_smp, ]
validate <- trouble_data[-calibrate_smp, ]
```

Run EFA with the `calibrate` sample

```
m_efa_1  <- mplusObject(
  TITLE = "School Trouble EFA - LAB 4 DEMO",
  VARIABLE =
    "usevar = BYS22A-BYS24G;",

  ANALYSIS =
    "type = efa 1 5;
     estimator = mlr;
     parallel=50; ! run parallel analysis",

  MODEL = "" ,

  PLOT = "type = plot3;",
  OUTPUT = "sampstat;",

  usevariables = colnames(calibrate),
  rdata = calibrate)

m_efa_1_fit <- mplusModeler(m_efa_1,
                dataout=here("efa_mplus", "efa1_trouble.dat"),
                modelout=here("efa_mplus", "efa1_trouble.inp"),
                check=TRUE, run = TRUE, hashfilename = FALSE)
```

Plot Parallel Analysis & Eigenvalues

read into R an Mplus output file

```
efa_summary <- readModels(here("efa_mplus", "efa1_trouble.out"), quiet = TRUE)
```

```
## <simpleError in seq.default(factorLB, factorUB): 'from' must be a finite number>
## <simpleError in if (is.null(summaries) || missing(summaries) || summaries$NCategoricalLatentVars ==
```

extract relavent data & prepare dataframe for plot

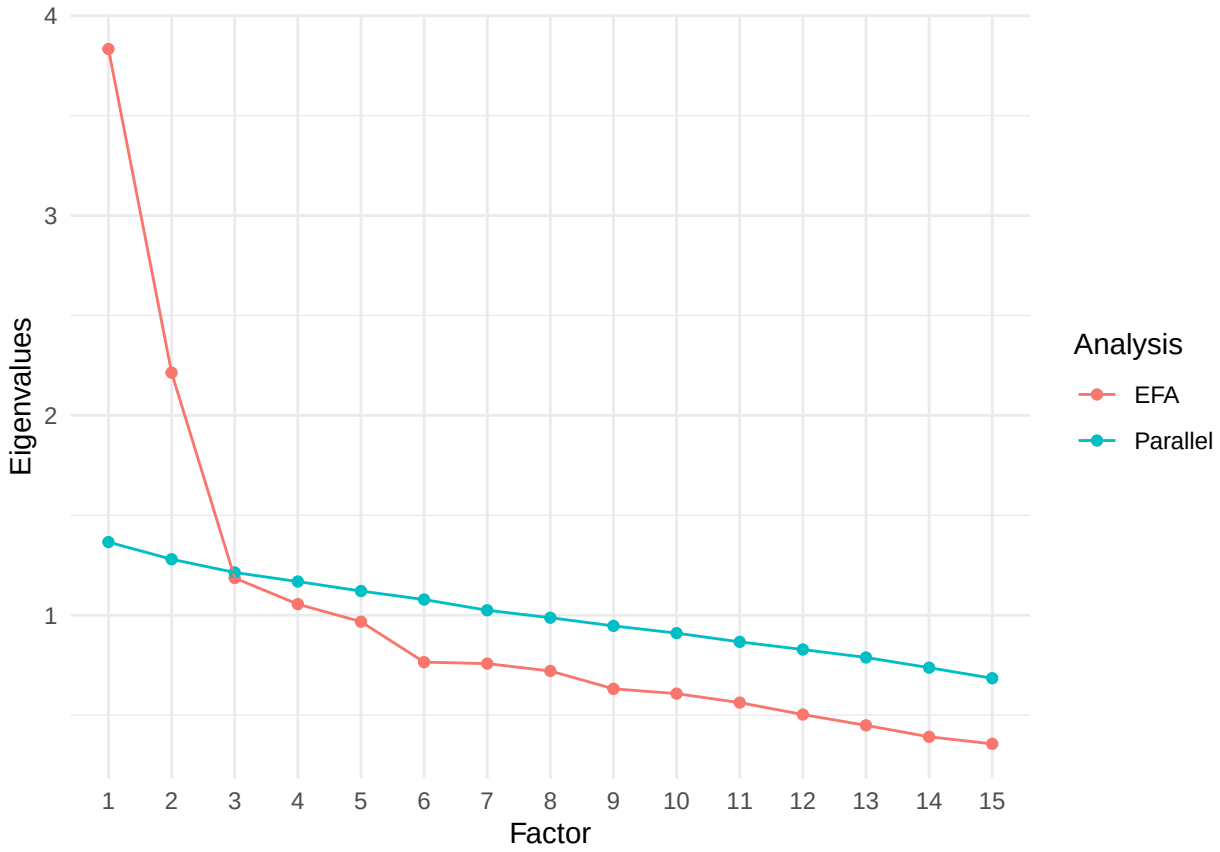
```
plot_data <- list(EFA=efa_summary[["gh5"]][["efa"]][["eigenvalues"]],
                  Parallel=efa_summary[["gh5"]][["efa"]][["parallel_average"]]) %>%
  as.data.frame() %>%
  mutate(Factor = row_number()) %>%
  mutate(Factor = factor(Factor))
```

pivot the dataframe to "long" format

```
plot_data_long <- plot_data %>%
  pivot_longer(EFA:Parallel,                # The columns I'm gathering together
               names_to = "Analysis",       # new column name for existing names
               values_to = "Eigenvalues")   # new column name to store values
```

plot using ggplot

```
plot_data_long %>%
  ggplot(aes(y=Eigenvalues,
             x=Factor,
             group=Analysis,
             color=Analysis)) +
  geom_point() +
  geom_line() +
  theme_minimal()
```



save figure to the designated folder

```
ggsave(here("figures", "eigenvalue_elbow_rplot.png"), dpi=300, height=5, width=7, units="in")
```

Introduction to MplusAutomation with iterators

Alternate way to run an EFA with the "calibrate" sample

```
m_efa_k15  <- lapply(1:5, function(k) {

  m_efa  <- mplusObject(
    TITLE = "School Trouble EFA - LAB 4 DEMO",
    VARIABLE =
      "usevar = BYS22A-BYS24G;",

    ANALYSIS =
      paste("type=efa", k, k),
```

```
    MODEL = "" ,

    PLOT = "type = plot3;",
    OUTPUT = "sampstat;",

    usevariables = colnames(calibrate),
    rdata = calibrate)

  m_efa_fit <- mplusModeler(m_efa,
                dataout=sprintf(here("efa_mplus2", "efa_trouble.dat"), k),
                modelout=sprintf(here("efa_mplus2", "efa_%d_trouble.inp"), k),
                check=TRUE, run = TRUE, hashfilename = FALSE)

})
```

References

Hallquist, M. N., & Wiley, J. F. (2018). MplusAutomation: An R Package for Facilitating Large-Scale Latent Variable Analyses in Mplus. Structural equation modeling: a multidisciplinary journal, 25(4), 621-638.

Horst, A. (2020). Course & Workshop Materials. GitHub Repositories, https://https://allisonhorst.github.io/

Muthén, L.K. and Muthén, B.O. (1998-2017). Mplus User's Guide. Eighth Edition. Los Angeles, CA: Muthén & Muthén

R Core Team (2017). R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria. URL http://www.R-project.org/

Wickham et al., (2019). Welcome to the tidyverse. Journal of Open Source Software, 4(43), 1686, https://doi.org/10.21105/joss.01686