Lab 4 - Random splits, Iterators, & Handling large data

Factor Analysis ED 216B - Instructor: Karen Nylund-Gibson

Adam Garber

1/28/2020

Lab 4 outline

1. Randomly split data into 2 equal parts (calibration & validation samples)
2. Introduction to MplusAutomation with iterators
3. Dealing with large data

Getting started - following the routine...

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

R-Project instructions:

1. click "NEW PROJECT" (upper right corner of window)
2. choose option "NEW DIRECTORY"
3. choose location of project (on desktop OR in a designated class folder)

Within R-studio under the files pane (bottom right):

1. click "New Folder" and name folder "data"
2. click "New Folder" and name folder "efa_mplus"
3. click "New Folder" and name folder "figures"

SKIP (return here if you recieve an error when loading packages)

1. First check if the packages load (run lines 32-39)

2. If an error is returned for package(s)

3. then run the following lines of code

4. AFTER ANY INSTALLATION... MUST RE-LOAD PACKAGES!

** **NOTE:** the following code ONLY needs to be run once per computer!

```
# to install one package at a time...
install.packages("janitor")
install.packages("haven")

# to install multiple pacakages at once...
install.packages(c("janitor",
                   "MplusAutomation",
                   "tidyverse",
                   "here",
                   "haven",
                   "corrplot"))
```

IF package rhdf5 does not load then run:

```
if (!requireNamespace("BiocManager", quietly = TRUE))
  install.packages("BiocManager")

BiocManager::install("rhdf5")
```

Lab 4 - Begin

---

loading packages...

```
library(janitor)
library(tidyverse)
library(haven)
library(MplusAutomation)
library(rhdf5)
```

```
library(here)
library(corrplot)
```

read in the raw dataset

```
lab_data <- read_spss(here("data", "lab_efa_cfa_data.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")
```

Randomly split a sample into 2 equal parts

---

find the size of half of original sample.

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, ]
```

Let's run an 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", "lab4_efa1_trouble.dat"),
                            modelout=here("efa_mplus", "lab4_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", "lab4_efa1_trouble.out"))
```

extract relavent data & prepare dataframe for plot

```
x <- list(EFA=efa_summary[["gh5"]][["efa"]][["eigenvalues"]],
          Parallel=efa_summary[["gh5"]][["efa"]][["parallel_average"]])

plot_data <- as_data_frame(x)
plot_data <- cbind(Factor = paste0(1:nrow(plot_data)), plot_data)

plot_data <- plot_data %>%
  mutate(Factor = fct_inorder(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  <- lapply(1:5, function(k) {
  m_efa2  <- 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_2_fit <- mplusModeler(m_efa2,
                              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)
})
```

Cleaning & subsetting large datasets

---

reading SPSS files is a lot slower than reading CSV formatted files

```
hsls_raw <- read_spss(here("data", "hsls_16_student_sub_v1.sav"))
```

make all column names "lower_snake_case" style

```
hsls_tidy <- hsls_raw %>%
  clean_names()
```

select usig the starts_with() function

```
hsls_x1 <- hsls_tidy %>%
  select(starts_with("x1")) # columns with first 2 characters "x1"
```

select using the end_with() function

```
hsls_not_sex <- hsls_tidy %>%
  select(!ends_with("sex")) # columns that do NOT end with "sex"
```

select using the end_with() function

```
hsls_science <- hsls_tidy %>%
  select(contains("sci")) # columns that contain characters "sci"

hsls_math <- hsls_tidy %>%
  select(contains(c("mth" , "math"))) # columns that contain "mth" or "math"
```

combine different select() arguements

```
hsls_math_sci <- hsls_tidy %>%
  select(contains(c("mth" , "math", "sci"))) %>%
  select(!starts_with("x1")) %>%
  select(!ends_with("sex"))
```

remove characters from the variable names that are greater than 8 characters

```
names(hsls_math_sci) = str_sub(names(hsls_math_sci), 1, 8)
```

check if culumn names are unique

```
test.unique <- function(df) { ## function to identify unique columns

  length1 <- length(colnames(df))
  length2 <- length(unique(colnames(df)))
  if (length1 - length2 > 0 ) {

  print(paste("There are", length1 - length2, " duplicates", sep=" "))
    }
}

test.unique(hsls_math_sci)
```

locate duplicates (this will find the column of the first duplicate)

```
anyDuplicated(colnames(hsls_math_sci))

names(hsls_math_sci)
```

Other functions to consider from the "stringr" package (part of tidyverse):

- str_remove()
- str_replace() # replace one string pattern with another
- str_match()
- str_pad() # to remove spaces
- str_count()
- str_detect()
- str_dup()
- str_extract_all()

End of lab 4 exercise

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.