Explore, prepare data, & an introduction to MplusAutomation

Adam Garber

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

March 15, 2020

Contents

---

1 ~~~~~ Lab outline

0. guideline for MplusAutomation workflow
1. create an R project in a dedicated project folder (on the desktop or in a class folder)
2. install & load packages
3. read in data to R

4. view data in R
5. view metadata (from SPSS files)
6. write .sav / .csv / .dat files
7. fix character names to have less than 8 character
8. filtering rows & selecting columns
9. change variable classes in R
10. visualize and explore data
11. introduction to mplusObjects

---

2 ~~~~~ Preparing to work with MplusAutomation

1. R PROJECTS: We will use R Projects for **ALL** labs & assignments.

- This is because `MplusAutomation` involves specifying many filepaths.

2. THE {`here`} package: To make filepaths unbreakable (reproducible)

- The same code will work across operating systems

3. PROJECT SUB-FOLDERS: Thoughfully organize files in sub-folders.

- This is **critical**, by the end of the quarter the number of Mplus files for an assignment will multiply rapidly

4. LOCATION OF PROJECT FOLDERS: on **desktop** or within a **single enclosing folder**.

- There is a limitation with the "mplusObject" function due to the fact that Mplus only reads the first 90 columns in each line.

e.g., if/your/filepath/has/many/nested/folders/it/will/be/longer/than/the/90character/limit/data.dat

---

2.1 Tools we will use in lab

Tool/Package	Purpose/Utility	Advantages
{MplusAutomation} package	Current capabilities supporting full SEM modeling	Flexibility (approaching infinite)
R Project	Unbreakable file paths & neatness	Reproducibility (kindness to your future self)
{tidyverse} package	Intuitive/descriptive function names	Accesability to new users
{here} package	Unbreakable/consistent file paths across OS	Reproducibility (for Science's sake!)
{haven} package	Viewable metadata in R from SPSS datafiles	Getting to know your measures
{ggplot2} package	Beautiful, customizable, reporoducible figures	Publication quality data vizualizations
pipe operator (%>%) notation	Ease of reading/writing scripts	e.g., first() %>% and_then() %>% and_finally()

---

3 ~~~~~ Creating an R-Project

1. create a **project folder** (that will enclose all files associated with a given lab or assignment)

- the project folder should be located on the desktop or within a designated class folder
- each lab or assignment should have its own designated project folder

2. create a **new project** (upper right hand corner of the R-studio window)
3. create **two sub-folders** in the project folder, one called "data", and one called "basic_mplus"

---

4 ~~~~~ Installing & loading packages

1. Copy the packages below into your R-script
2. Save your R-script in your main project folder
3. If a yellow banner appears at the top of your screen click "install packages"

4.1 install the "rhdf5" package to read gh5 files

```
if (!requireNamespace("BiocManager", quietly = TRUE))
    install.packages("BiocManager")
 BiocManager::install("rhdf5")
```

```
# how to install packages?
install.packages("tidyverse")
```

4.2 load packages

```
library(tidyverse)
library(haven)
library(here)
library(MplusAutomation)
library(rhdf5)
library(reshape2)
library(corrplot)
library(GGally)
library(corrgram)
```

4.3 Keyboard shortcuts

- ALT + DASH(-) = <-
- SHIFT + CONTROL = %>%

---

5 ~~~~~ Read in data

```
# object_name <- function1(nested_function2("dataset_name.sav"))

exp_data <- read_spss(here("data", "explore_lab_data.sav"))
```

6 ~~~~~ A couple ways to explore & view data in R

```
# 1. click on the data in your Global Environment (upper right pane) or use...
View(exp_data)

# 2. str() allows you to view the structure of dataframe including class of variables (e.g., factors, o
str(exp_data)

# 3. summary() gives basic summary statistics & shows number of NA values
#               *great for checking that data has been read in correctly*
summary(exp_data)

# 4. names() provides a list of column names. Very useful if you don't have them memorized!
names(exp_data)

# 5. head() prints the top x rows of the dataframe
head(exp_data)

# 6. glimpse() and another way to look at the dataframe, just depends on your preference!
glimpse(exp_data)
```

7 ~~~~~ View dataframe with labels & response scale meta-data

7.1 Note: Use the "print" option to save a PDF as a codebook of metadata.

```
# the {haven} package keeps the meta-data from SPSS files

# package_name::function_within_package()

sjPlot::view_df(exp_data)
```

7.2 Types of data for different tasks

- SAV (e.g., spss_data.sav): this data format is for SPSS files & contains variable labels (meta-data)
- CSV (e.g., r_ready_data.csv): this is the preferable data format for reading into R (no labels)
- DAT (e.g., mplus_data.dat): this is the data format used to read into Mplus (no column names or strings)

NOTE: Mplus also accepts TXT formatted data (e.g., mplus_data.txt)

---

8 ~~~~~ Writing, reading, and converting data between 3 formats

---

8.1 prepare datasets, remove SPSS labeling

```
# write a CSV datafile (preferable format for reading into R, without labels)
write_csv(exp_data, here("data", "exp_lab1_data.csv"))

# write a SPSS datafile (preferable format for reading into SPSS, labels are preserved)
write_sav(exp_data, here("data", "exp_lab1_data.sav"))
```

```
# read the unlabeled data back into R
nolabel_data <- read_csv(here("data", "exp_lab1_data.csv"))
```

```
# write a DAT datafile (this function removes header row & converts missing values to non-string)
prepareMplusData(nolabel_data, here("data", "exp_lab1_data.dat"))
```

---

9 ~~~~~ Preparing column-names to be `MplusAutomation` ready

9.1 Task: Make all variable names fit within the 8-character name limit (Mplus) while avoiding duplicates.

9.2 Renaming columns manually...

```
# use function: rename(new_name = old_name)
new_names <- nolabel_data %>%
  rename( school_motiv1 = item1 ,
          school_motiv2 = item2 ,
          school_motiv3 = item3 ,
```

```
          school_comp1  = item4 ,
          school_comp2  = item5 ,
          school_comp3  = item6 ,
          school_belif1 = item7 ,
          school_belif2 = item8 ,
          school_belif3 = item9 )
```

9.3 What do you do if you have a large dataset with many column names that are > 8 characters?

- first, remove all characters greater than 8 using str_sub()
- second, make sure you don't now have duplicate variable names
- third, locate and change all duplicate names

```
# remove characters from the variable names that are greater than 8 characters
names(new_names) <- str_sub(names(new_names), 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(new_names)

# locate duplicates (this will find the column of the first duplicate)
anyDuplicated(colnames(new_names))
```

---

10 ~~~~~ Filtering rows (observations) & selecting columns (variables).

10.1 When to use?... create new dataframe subsets for particular analyses or purposes

```
# filtering observations & selecting variables

# new_data_frame <- existing_data_frame %>%
#       function_filter_rows(variable == value) %>%
#       function_select_columns(first_column:ninth_column)
```

```r
females <- nolabel_data %>%
  filter(female == 1) %>%
  select(1:9) # column numbers you are selecting for new dataframe object

# an alternative way to select columns, write a list of variable names (case-sensitive)

males <- nolabel_data %>%
  filter(female == 0) %>%
  select(item1, item2, item4)
```

10.2 A note on coding style:

- Naming conventions: **Be consistent!**
- I use the style lower snake case (e.g., this_is_lower_snake_case)
- Annotate code generously
- Let your code breath: use return often to spread code chuncks out vertically (dense paragraphs of code are a headache to look at)

---

11 ~~~~~ Changing variable class types

e.g., change class `numeric` to `factor`

```r
str(var_class)
var_class <- nolabel_data

# change variable "female" to a factor
var_class <- var_class %>%
  mutate(female = factor(female))

# change a set of variables to factors using "modify_at"
var_class %>%
  modify_at(c(1:9), as.factor) %>%
  str()

# change all factors back to numeric using "modify_if"
var_class %>%
  modify_if(is.factor, as.numeric) %>%
  str()
```

---

12 ~~~~~ visualizing data using `ggplot()`

12.1 boxplot

```
# using ggplot:
# whenever you are using ggplot and you specify a variable name
# it needs to be within "aes()" which stands for aesthetic

# Making a box plot
nolabel_data %>%          # the data frame
  ggplot(aes(y=item1)) +  # ggplot with aes( y = variable_name)
    geom_boxplot()        # specify the type of plot
```

12.2 histogram

```
nolabel_data %>%
  ggplot(aes(x = item2)) +
    geom_histogram()
```

12.3 density plot

```
nolabel_data %>%
  ggplot( aes(x=item3)) +
    geom_density(fill="#69b3a2",  # change the aesthetics
                 color="#e9ecef", # add a color
                 alpha=0.8)       # make fill transparent
```

12.4 jitter plot

```
nolabel_data %>%
  ggplot( aes(x=item3, y=item4)) +
    geom_jitter(alpha = .5 )
```

12.5 qq-plot

```
# Quantile-quantile plot:

nolabel_data %>%
ggplot(aes(sample = item3)) +
  geom_qq(size = .8, alpha = 0.5) +
  facet_wrap(~female) +
  stat_qq_line() +
  labs(title = "Quantile-quantile plots, check of normality")
```

13 ~~~~~ Visualizing & exploring sets of variables

In factor analysis you often want to look at a set of variables at once rather than one at a time. . .

```
# loops, use purrr::map() to make histograms for a series of items

 nolabel_data %>%
   select(1:9) %>%
   names() %>%
   map(~ggplot(nolabel_data, aes_string(x = .)) + geom_histogram())


# alternatively, use the facet wrap function with melt to make one graph

melt(nolabel_data[,1:9]) %>%                    # within brakets [rows, columns]
  ggplot(., aes(x=value, label=variable)) +
  geom_histogram(bins = 15) +
  facet_wrap(~variable, scales = "free")       # scales="free" allows the x-axes to vary
```

13.1 faceted scatterplot

```
# scatterplot
melt(nolabel_data[,1:9]) %>%
  ggplot(., aes(y=value, x=variable)) +
  geom_jitter() +
  facet_wrap(~variable, scales = "free")
```

13.2 violin plot

```
# violin plot
melt(nolabel_data[,1:9]) %>%
  ggplot(., aes(y=value, x=variable)) +
  geom_violin() +
  facet_wrap(~variable, scales = "free")
```

---

14 ~~~~~ correlations - there are many ways to visualize bi-variate relations in R

14.1 package {corrplot}

```r
f_cor <- cor(females, use = "pairwise.complete.obs")

corrplot(f_cor,
         method="number",
         type = "upper")
```

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

14.2 package {corrgram}

```r
corrgram(nolabel_data[,1:9],
         order=TRUE,
         lower.panel=panel.ellipse,
         upper.panel=panel.pts,
         text.panel=panel.txt, diag.panel=panel.minmax,
         main="Explore Data")
```

14.3 package {ggpairs}

```r
ggpairs(
  nolabel_data[,1:9],
  upper = list(continuous = "density", combo = "box_no_facet"),
  lower = list(continuous = "points", combo = "dot_no_facet"))
```

---

15 ~~~~~ Simple introduction to MplusAutomation (writing, running, & reading models)

15.1 What does the mplusObject() function do:

1. It generates an Mplus **input** file (does not need full variable name list, its automated for you!)
2. It generates a **datafile** specific to each model
3. It **runs** or estimates the model (hopefully) producing the correct output. **Always check!**

---

16 PRACTICE: Using MplusObject() method (type = BASIC;)

```
m_basic  <- mplusObject(
  TITLE = "PRACTICE 01 - Explore TYPE = BASIC",
  VARIABLE =
 "usevar=
 item1 item2 item3 item4 item5
 item6 item7 item8 item9 female;

 ! use exclamation symbol to make comments, reminders, or annotations in Mplus files",

  ANALYSIS =
 "type = basic; ",

  usevariables = colnames(nolabel_data),
  rdata = nolabel_data)

m_basic_fit <- mplusModeler(m_basic,
                            dataout=here("basic_mplus", "basic_Lab1_DEMO.dat"),
                            modelout=here("basic_mplus", "basic_Lab1_DEMO.inp"),
                            check=TRUE, run = TRUE, hashfilename = FALSE)

## END: TYPE = BASIC MPLUS AUTOMATION PRACTICE
```

---

17 PRACTICE: Now explore descriptives for observations that reported as "female"

Add line of syntax: "useobs = female == 1;"

```
fem_basic  <- mplusObject(
  TITLE = "PRACTICE 02 - Explore female observations only",
  VARIABLE =
 "usevar=
 item1 item2 item3 item4 item5
 item6 item7 item8 item9;

 useobs = female == 1; !include observations that report female in analysis",

  ANALYSIS =
 "type = basic;",

  usevariables = colnames(nolabel_data),
  rdata = nolabel_data)

fem_basic_fit <- mplusModeler(fem_basic,
                            dataout=here("basic_mplus", "fem_basic_Lab1_DEMO.dat"),
```

```
                            modelout=here("basic_mplus", "fem_basic_Lab1_DEMO.inp"),
                            check=TRUE, run = TRUE, hashfilename = FALSE)
```

---

18 PRACTICE: Exploratory Factor Analysis (EFA)

```
## EXPLORATORY FACTOR ANALYSIS LAB DEMONSTRATION

efa_demo  <- mplusObject(
  TITLE = "EXPLORATORY FACTOR ANALYSIS - LAB DEMO",
  VARIABLE =
 "usevar=
 item1 item2 item3 item4 item5
 item6 item7 item8 item9;" ,

  ANALYSIS =
 "type = efa 1 5;
  estimator = MLR;
  parallel=50;",

  MODEL = "" ,

  PLOT = "type = plot3;",

  OUTPUT = "sampstat standardized residual modindices (3.84);",

  usevariables = colnames(nolabel_data),
  rdata = nolabel_data)

efa_demo_fit <- mplusModeler(efa_demo,
                             dataout=here("basic_mplus", "EFA_Lab_DEMO.dat"),
                             modelout=here("basic_mplus", "EFA_Lab_DEMO.inp"),
                             check=TRUE, run = TRUE, hashfilename = FALSE)

## END: EXPLORATORY FACTOR ANALYSIS LAB DEMONSTRATION
```

---

19 END OF LAB

---

20 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

UC SANTA BARBARA