# Getting Started & Running a Simple Model

*Adam Garber*

A Course in `MplusAutomation`

October 12, 2021

---

## Outline

1. Create a new `R Project` (on your computers desktop or in a designated project folder)
2. Downloading a project repository from Github
3. Install & load packages
4. Read in data to R
5. View data in R
6. View metadata (from SPSS files)
7. Write `.sav` , `.csv` , and `.dat` files
8. Fix character names to have less than 8 character
9. Introduction to the "mplusObject method"
10. Run a first model using `MplusAutomation`

---

## Getting started repository:

https://github.com/garberadamc/QF-Getting-Started

---

## Load packages

```
# install.packages("MplusAutomation")

library(MplusAutomation)
library(tidyverse)
library(haven)
library(here)
library(sjPlot)
```

---

## Read in data

---

```
# object_name <- function_1("dataset_name.sav")

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

---

## View dataframe with labels & response scale meta-data

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)
```

---

## Types of data for different tasks

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

---

## Writing, reading, and converting data between 3 formats

---

## Location, location, location!

NOTE: default directory in an `Rproject` is the "top-most" project folder

## Prepare data: Remove SPSS labels

Write a `.csv` data file (preferable format for reading into R)

Read the unlabeled `.csv` data back into R

Write a `.dat` file using the `prepareMplusData()`

```
# This function removes header row and converts missing values to non-string characters
```

## Function `prepareMplusData()`:

1. This function produces as output minimal template of input syntax for an Mplus input file.
2. Behind the scenes `mplusObject()` will use a similar function to produce an input file & `.dat` file from the `R data.frame object` that the function takes as input.
3. By default missing values in your R object (`NA`) are converted to a period ( `.` ).

---

## Preparing column-names to be `MplusAutomation` ready

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

Renaming columns manually...

```
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 )
```

---

## A minimal example of writing, running, & reading models

---

## The "`mplusObject()` Method"

### What does the `mplusObject()` function do?

- Takes an `R data.frame` and produces an object that contains all the information necessary to generate an Mplus **input** file.

**What does the `mplusModeler()` function do?**

1. It generates a data file (`.dat`)
2. It generates the input file (`.inp`)
3. It **runs** or estimates the model producing the output file (`.out`). **Always check that the model estimated correctly!**

**NOTE:** Within the `mplusObject()` function there is a mix of `R` & `Mplus` syntax.

---

## `R` terminology - `functions` & `arguments`

- `mplusObject()` is a function from the {`MplusAutomation`} package (i.e., `MplusAutomation::mplusObject()`)
- If preferred you can mention the package explicitly for greater transparency (i.e., `MplusAutomation::mplusObject()`)
- Functions have one or more `arguments` or `inputs`
- The inputs for the `mplusObject()` function include `TITLE =`, `VARIABLE =`, `ANALYSIS =`, `usevariables =`, `rdata =` (among others)
- Arguments within functions are separated by a comma (`,`)

Within an `mplusObject()`:

- **Black colored text** = Arguments or inputs (i.e., `R` code)
- **Green colored text** (within quotation marks) = `Mplus` syntax (e.g., `"type = basic;"`)

---

## Create an `mplusObject()` & `mplusModeler()` template

---

## Run a first model using the `mplusObject()` method

Model is `type = BASIC;` (i.e., returns descriptive statistics)

## Always check your model!

- In the RStudio window pane on the `bottom-right`under the `files` tab click on the `basic_mplus` folder
- There should be 3 new files in this location that were produced by `mplusModeler()`
- Click on the output file (`.out`) to check if the model estimated or if there are any error messages

---

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.

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