Introduction to `MplusAutomation`

Adam Garber

May 31, 2021

Norwegian University of Science and Technology

About me:

- I am a doctoral student in the Education department studying quantitative methods at the University of California, Santa Barbara
- I study the development and communication of mixture modeling methods in-line with my advisor Dr. Karen Nylund-Gibson's work.
- The materials used in this workshop have been adapted from content I developed as a teaching assistant for courses in `Factor Analysis`, `Structural Equation Modeling`, & `Applied Mixture Modeling`.

How/why did I learn `MplusAutomation`?

- I taught myself this strategy in response to spending much of my time as a data analyst using Mplus sifting through messy research projects.
- The motivation for using this method is to increase reproducibility, organization, and transparency.

Goals (estabishing a `routine`):

- The goal of this method is to have an entire project in a single location following coherent naming and organization conventions.
- The code presented will be very repetitive by design. Creating a routine is key!

Pace:

- Please help me set the pace of this workshop. Feedback will be helpful and please ask questions.
- We have a large amount of material to cover so I will try and keep up a pretty fast pace.

Troubleshooting:

- Due to differences in operating systems and R environments, code that I have tested may not work on your computer.
- It may not be possible to address all of these issues during the workshop.
- Please reach out after the workshop and I will do my best to help (`agarber at ucsb.edu`)

Preparing to work with MplusAutomation

1. R PROJECTS: We will use R Projects to contain code & associated files in a single location

- `MplusAutomation` involves specifying many filepaths therefore organization and folder structure is key.

2. THE {`here`} package is used to make filepaths unbreakable (reproducible)

- The same code will work across operating systems

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

- This is **critical**, the large number of Mplus files produced necessitates careful consideration of folder location

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

---

Resources:

`Rproject` | `Rmarkdown` | `Git-Github`

- R-studio and R-Projects Tutorials
- Rmarkdown Basics Tutorial
- Connect Git-Hithub with R-studio and Download Repositories

`MplusAutomation`

- Published Documentation - Hallquist and Wiley, 2018
- Vignette Examples - Hallquist

Steps to download repositories from Github and create a version controlled R-project

0. Create a Github account and connect R-Studio with Git
1. Go to the repository link to **Fork** and **Clone** (copy address) the repository:
2. Within R-studio create a **New project** and choose the **Version Control** Option (Git)
3. Paste the repository address copied (cloned) from Github and save locally on your computer
4. After making changes in your branch of the repository to update the version on Github follow the following sequence of steps: `Stage`, `Commit` (add commit message), `Pull`, and then `Push`

A note on coding style:

- Naming conventions: **Be consistent!**
- I use the naming convention called *lower snake case* (e.g., `this_is_lower_snake_case`)
- Annotate code generously
- Let your code breath: use return often to spread code chunks out vertically

---

Preparation

Large repository procedure:

- Typically a project will have a single script containing all of the code (e.g., `.R` or `.Rmd`)
- Due to the number of scripts in this repository we will change the starting file-path at the beginning of each exercise to the folder associated with that script.
- To make this work each course participant must manually copy the file-path of their project folder location into the file named `rep_functions.R`.

```
project_location <- "/your/unique/file-path-stem/NTNU-start" # NOTE manual change required
```

- This can easily be done by loading the `here` package, running `here()`, and copying the path.
- This procedure is unique to this large repository structure & will generally not be a necessary step.

Packages to install & load before the beginning of the workshop:

Install the `rhdf5` package to read `gh5` files

```
# NOTE: This package is only necessary for plotting
if (!requireNamespace("BiocManager", quietly = TRUE))
    install.packages("BiocManager")
 BiocManager::install("rhdf5")
```

Install packages

```
install.packages(
  c(
  "tidyverse", "glue", "janitor", "haven", "here", "MplusAutomation", "gt", "rhdf5",
  "sjPlot", "corrplot", "semPlot", "DiagrammeR", "stargazer", "linguisticsdown",
  "gtsummary", "reshape2", "ggridges", "beepr", "praise", "Ecdat", "carData", "plotly",
  "viridis", "gganimate", "naniar", "cowplot", "poLCA", "gg3D", "viridis", "DT", "tidyLPA",
  "relimp", "psych"
  )
)
```

Load packages

```
# important
library(tidyverse)
library(glue)
library(janitor)
library(haven)
library(here)
library(MplusAutomation)
library(gt)
library(rhdf5)

# less important
library(sjPlot)
library(corrplot)
library(semPlot)
library(DiagrammeR)
library(stargazer)
library(linguisticsdown)
library(gtsummary)
library(reshape2)
library(ggridges)
library(beepr)
library(praise)
library(Ecdat)
library(carData)
library(plotly)
library(viridis)
library(gganimate)
library(naniar)
library(cowplot)
library(poLCA)
library(gg3D)
library(viridis)
library(DT)
library(tidyLPA)
library(relimp)
library(psych)
```