Running a Simple Model & Finding a Routine

Adam Garber

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

June 01, 2021

Guide

1. create an R project in a dedicated project folder (on the desktop or in a designated project 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. introduction to mplusObjects

Install the "rhdf5" package to read `gh5` files (plot information)

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

Load packages

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

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 (meta-data)
- `.csv` (e.g., `r_ready_data.csv`): this is the preferable data format for reading into `R` & when estimating models with `MplusAutomation` (non-labeled data)
- `.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`)

Writing, reading, and converting data between 3 formats

Location, location, location!

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

```
here()
```

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

Prepare datasets, **remove SPSS labeling**

```r
# write a CSV datafile (preferable format for reading into R; !! removes labels !!)
write_csv(exp_data, here("02-run-models", "data", "exp_lab1_data.csv"))

# write a SPSS datafile (preferable format for reading into SPSS, labels are preserved)
write_sav(exp_data, here("02-run-models", "data", "exp_lab1_data.sav"))
```

```r
# read the unlabeled data back into R
nolabel_data <- read_csv(here("02-run-models", "data", "exp_lab1_data.csv"))
```

```r
# write a DAT datafile (this function removes header row & converts missing values to non-string)
prepareMplusData(nolabel_data, here("02-run-models", "data", "exp_lab1_data.dat"))
```

THINGS TO NOTE:

1. This function produces a 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` that the function is given.
3. By default missing values in your R object (`NA`) are converted to `.`
4. Note that if the following `file.string` is greater than **90 characters** (Mplus limit) your model will not estimate and return an error message.

```
DATA: FILE = "/Users/agarber/github/NTNU-workshop/02-run-models/data/exp_lab1_data.dat";
```

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...

```r
# 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 )
```

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

NOTE: Use regular expressions to remove strings in a more systematic manner to avoid duplicates.

```
# 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)
```

```
## [1] "There are 6  duplicates"
```

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

```
## [1] 2
```

A minimal example of writing, running, & reading models

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 **data file (`.dat`)** specific to each model
3. It **runs** or estimates the model (hopefully) producing the correct output. **Always check!**

NOTE: Within the `mplusObject()` function there is a mix of `R` and `Mplus` syntax. In general the black colored text will be R code and the green colored text within the quotation marks will be Mplus code (exception; when using {`glue`}).

PRACTICE: Using MplusObject() method

Model is `type = BASIC;` (i.e., returns descrptive stats)

```
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("02-run-models", "basic_mplus", "basic.dat"),
               modelout=here("02-run-models", "basic_mplus", "basic.inp"),
               check=TRUE, run = TRUE, hashfilename = FALSE)
```

Check your model (Always!)

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

---

PRACTICE SUBSETTING: 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("02-run-models", "basic_mplus", "fem_basic.dat"),
                  modelout=here("02-run-models", "basic_mplus", "fem_basic.inp"),
                  check=TRUE, run = TRUE, hashfilename = FALSE)
```

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("02-run-models", "basic_mplus", "EFA_Lab_DEMO.dat"),
  modelout=here("02-run-models", "basic_mplus", "EFA_Lab_DEMO.inp"),
  check=TRUE, run = TRUE, hashfilename = FALSE)
```

END
