Introduction to MplusAutomation

MM4DBER Training Team

Updated: August 29, 2023

Mixture Modeling for Discipline Based Education Researchers (MM4DBER) is an NSF funded training grant to support STEM Education scholars in integrating mixture modeling into their research.

- Please visit our website to learn more and apply for the year-long fellowship.
- Follow us on Twitter!

Visit our GitHub account to download the materials needed for this walkthrough.

Introduction:

In this exercise we will obtain the same Mplus *.out* file that we produced using the `Mplus` program but this time from our `RStudio` window.

Here we will use the `MplusAutomation` package (Hallquist & Wiley, 2018).

Review (rmarkdown basics):

- At the top of the rmarkdown is a couple of lines called the `yaml` for now we can leave this be (you can edit title or author name here)

- The first code chunk (bellow the `yaml`) is called the "r setup" code chunk. This will set the defaults for all code blocks in your document. For now we will leave this as is.
- The next code chunk in any given rmarkdown should include the packages you will be using.
- To insert a code chunk, either use the keyboard shortcut `ctrl + alt + i` OR click the green button with the letter `C` on it (top panel).

Step 1: Load packages

```
library(MplusAutomation)  # To estimate Mplus models from Rstudio
library(tidyverse)        # Collection of R packages designed for data science
library(here)             # To set our file.paths
library(psych)            # To use the describe() function
library(ggpubr)           # To use the ggdensity() and ggqqplot() functions
library(corrplot)         # To use the corrplot() function
library(jtools)           # For running logistic regression in R
```

Common error message types: [E.g.,. . .]

- If a function does not work and you receive an error message: `could not find function "xxx_function"`
- [OR] If you try to load a package and you receive an error like this: `` there is no package called `xxx_package` ``
- then you will need to install the package using: `install.packages("xxx_package")`

Step 2: Read in data set

```
data_lsay <- read_csv(here("data", "lsay_sci_data.csv"))
```

Let's take a look at our data

```
# 1. summary() gives basic summary statistics & shows number of NA values
summary(data_lsay)

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

# 3. head() prints the top x rows of the dataframe
head(data_lsay)
```

Step 3: Using `MplusAutomation`

To run a basic model using `MplusAutomation` we will use two main functions, the `mplusObject()` function and the `mplusModeler()` function.

What does the `mplusObject()` function do?

1. It prepares and 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

What does the `mplusModeler()` function do?

1. It runs or estimates the model (hopefully) producing the correct output. Always check!
2. It also reads the Mplus output file creating an object in your R environment
3. You can specify where you want the *.out* file saved using `modelout=`

Run descriptive statistics using MplusAutomation

NOTE:

- You do not need to specify the Mplus `MISSING` statement (`MplusAutomation` detects the missing value from the data set and does this for you).
- You also do not need to specify the Mplus `NAMES` statement (`MplusAutomation` detects the names from the R data object that is read in and writes this syntax for you).

```r
m_basic  <- mplusObject(

  TITLE = "PRACTICE 01 - Explore descriptives (TYPE = BASIC;)",

  VARIABLE =
  "usevar= Enjoy Useful Logical Job Adult Female;",

  ANALYSIS =
  "type = basic; ",

  usevariables = colnames(data_lsay),
  rdata = data_lsay)

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

- After running an `MplusObject` function, MplusAutomation will generate an output file.
- ALWAYS check your output before moving forward with your analyses.
- It's easy to skip past checking our output since MplusAutomation doesn't automatically present it to us after running the code.
- It's good practice to make it a habit to check your output file after every run.

Run a logistic regression model using MplusAutomation

```
logistic  <- mplusObject(

  TITLE = "PRACTICE 02 - Run a logistic regression model",

  VARIABLE =
  "usevar = Enjoy Female;
   categorical = Enjoy; ",

  ANALYSIS =
  "ESTIMATOR=ML;",

  MODEL =
  "Enjoy ON Female;",

  usevariables = colnames(data_lsay),
  rdata = data_lsay)

logistic_fit <- mplusModeler(logistic,
               dataout=here("mplus_files", "logistic.dat"),
               modelout=here("mplus_files", "logistic.inp"),
               check=TRUE, run = TRUE, hashfilename = FALSE)
```

Optional exercise

We may not get to this in class. If so, it is **highly recommended** that you run through this code after class.

Estimate a logistic regression model in R using the `glm()` function

Practice: Compare the logistic model outputs produced by `R` and `Mplus`

```
lsay_fit <- glm(Enjoy ~ Female,
                family = binomial,
                data = data_lsay)

summ(lsay_fit          # print regression coefficients & model fit statistics
     #exp = TRUE,      # option to print coefficients as Odds Ratios
     #confint = TRUE   # option to include confidence intervals in output
    )
```

Observations	3042 (19 missing obs. deleted)
Dependent variable	Enjoy
Type	Generalized linear model
Family	binomial
Link	logit

Converting from logit to probability (intercept=.591, coeff_Female= -.267)

$\chi^2(1)$	12.79
Pseudo-R^2 (Cragg-Uhler)	0.01
Pseudo-R^2 (McFadden)	0.00
AIC	4051.36
BIC	4063.40

	Est.	S.E.	z val.	p
(Intercept)	0.59	0.05	11.23	0.00
Female	-0.27	0.07	-3.57	0.00

Standard errors: MLE

```r
# Write a function called `logit2prob` to convert logits to probabilities
logit2prob <- function(logit){
     odds <- exp(logit)
     prob <- odds / (1 + odds)
     return(prob)
     }

# Probability of reporting `Enjoy science` for Males:
logit2prob(.591)
```

```
## [1] 0.6435946
```

```r
# Probability of reporting `Enjoy science` for Females (intercept - coefficient):
logit2prob(.591-.267)
```

```
## [1] 0.5802988
```

```r
# Write equation manually (like a calculator)
exp(.591-.267)/(1+exp(.591-.267))
```

```
## [1] 0.5802988
```

Data Cleaning & Screening

- It's important to explore your data before running your analyses.
- First, lets rename our variables to something more meaningful using `rename()`.
- As a reminder, use the pipe operator `%>%` to create a sequence of functions, you can use the shortcut `crt + shift + m`:
- To save an object in R we use `<-`, you can use the shortcut `option + (-)`

Read in a new data file into R

```r
exp_data <- read_csv(here("data", "exp_data.csv"))
```

```r
clean_data <- exp_data %>%
  rename(school_motiv1 = item1,  # new_name = old_name
         school_motiv2 = item2,
         school_motiv3 = item3,
         school_comp1 = item4,
         school_comp2 = item5,
         school_comp3 = item6,
         school_belif1 = item7,
         school_belif2 = item8,
         school_belif3 = item9)
```

Descriptive Statistics

Let's look at descriptive statistics for each variable using `psych::describe()` function:

```r
clean_data %>%
  describe()
```

What if we want to look at a subset of the data?

- For example, what if we want to see those who identify as female?
- We can use `tidyverse::filter()` to subset the data using a specified criteria

```r
clean_data %>%
  filter(female == 1) %>%
  describe()

#You can use any operator to filter: >, <, ==, >=, etc.
```

Missing Values

- Let's check for missing values.
- First, how are missing values identified?
- They could be `-999`, `NA`, or literally anything else.
- The simplest way to do this is to look back at the `summary()` function.
- There are four variables with one missing value.

```r
clean_data %>%
  summary()
```

Recode Continuous Variable into Factor

- What if you want to recode a continuous variable into different levels (e.g., high, medium, and low)?
- Let's use the variable `school_belief1` as an example.
- First, let's recall the descriptives:

```r
clean_data %>%
  select(school_belif1) %>%
  summary()
```

Here, we can see that the values range from 1 - 10. Lets recode the variable into three intervals using cut points:

Low | 1 - 3 |
Medium | 4 - 6 |
High | 7 - 10 |

We use can use `cut()` to divide the continuous variables into intervals creating a `factor` (i.e., the name for a categorical variable in R):

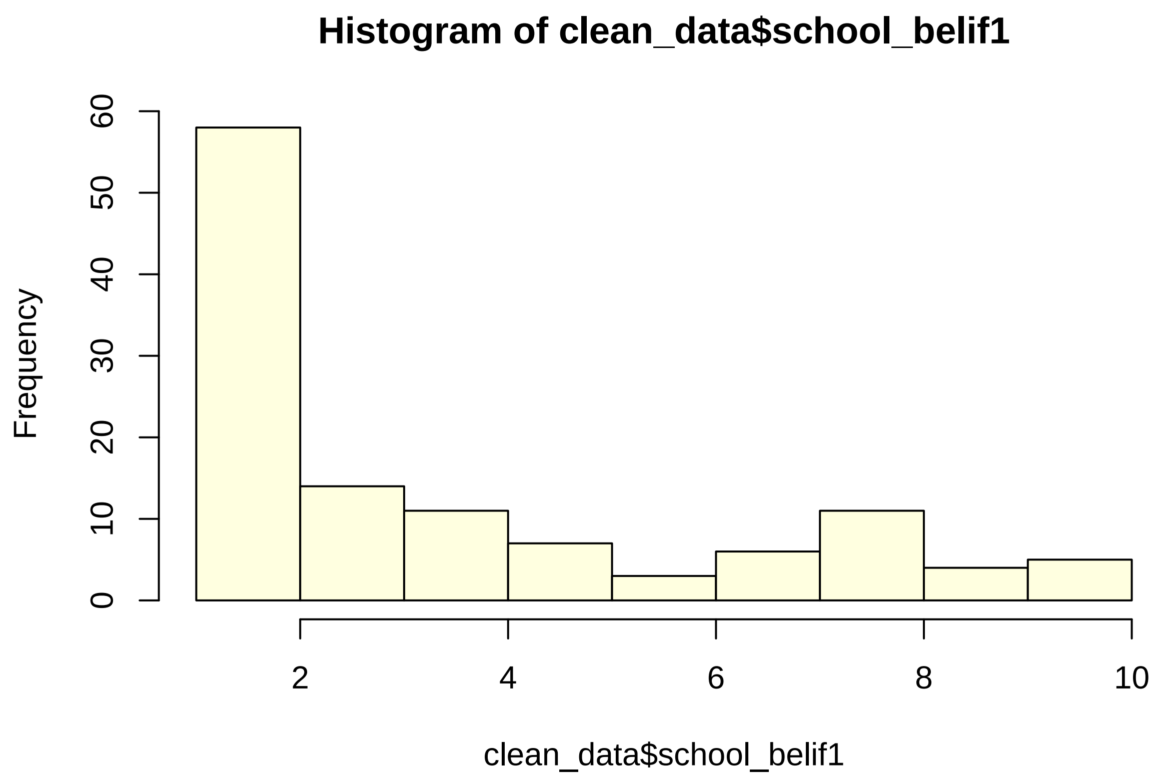
```
df_add_factor <- clean_data %>%
  mutate(school_factor = cut(school_belif1,
                      breaks = c(0, 3, 6, 10), #Use 0 at the beginning to ensure that 1 is included in
                      labels = c("low", "medium", "high")))
# View summary
df_add_factor %>%
  select(school_belif1, school_factor) %>%
  summary()
```

```
##  school_belif1   school_factor
##  Min.   : 1.000   low   :72
##  1st Qu.: 1.000   medium:21
##  Median : 3.000   high  :26
##  Mean   : 3.689
##  3rd Qu.: 5.000
##  Max.   :10.000
```

Visualizing our distributions: Histogram plot:

- Puts continuous data into bins (Note: we can change default bin size)
- Also, a great way to look at our categorical (binary, nominal), or ordinal variables (i.e., Likert scales)

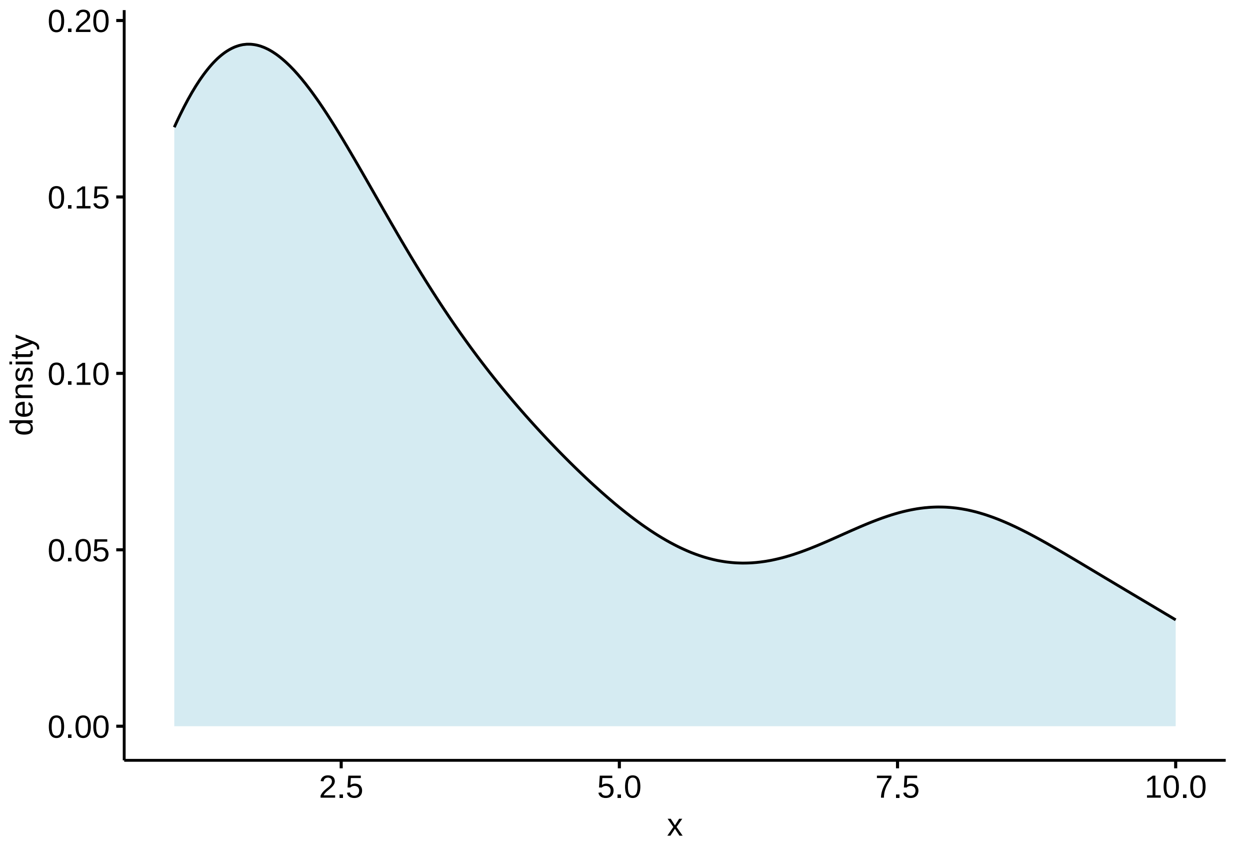
```
hist(clean_data$school_belif1, col = 'lightyellow')
```



Density plots:

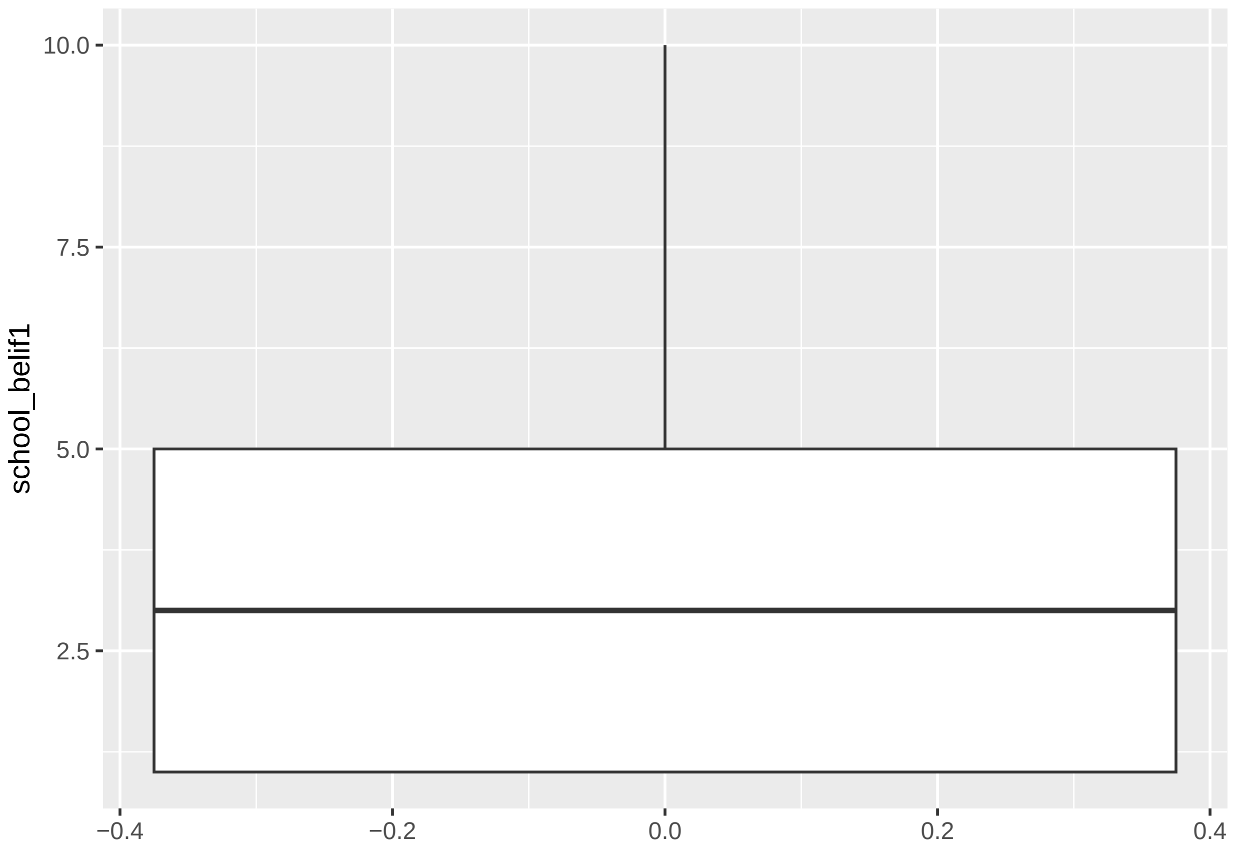
A density plot is a visualization of the data over a continuous interval. As we can see by this density plot, the variable `school_belif1` is positively skewed.

```
ggdensity(clean_data$school_belif1, fill = "lightblue")
```



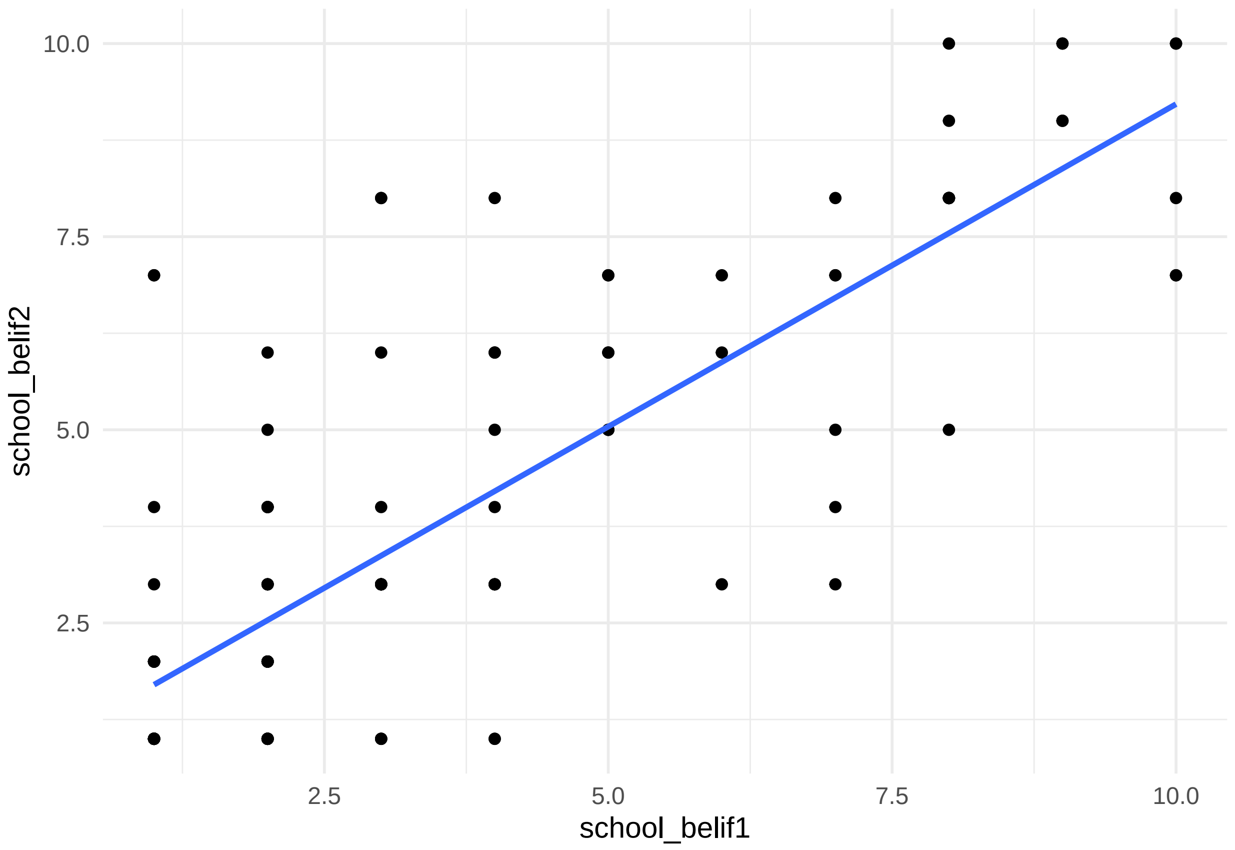
Box Plots Box plot can show us distributions, specifically the minimum, first quartile, median, third quartile, and maximum.

```
clean_data %>% #
  ggplot(aes(y = school_belif1)) +
  geom_boxplot()
```



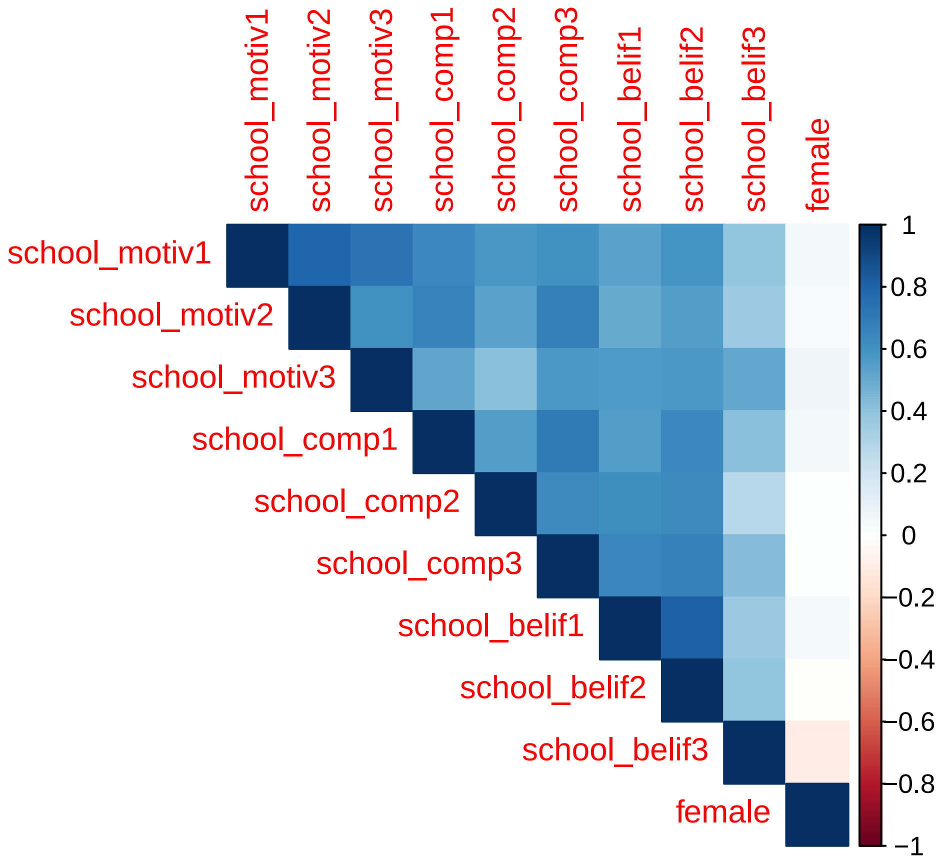
Bivariate Scatterplots We can look at a single bivariate scatterplot using `ggplot()`:

```
clean_data %>%
  ggplot(aes(school_belif1, school_belif2)) +
  geom_point() +
  geom_smooth(method = "lm", se =F) +
  theme_minimal()
```



We can also use `cor()` to look at bivariate correlations for the entire data set (Note: Missing values are not allowed in correlation analyses, use `drop_na()` to do list.wise deletion):

```
# A colorful plot:
f_cor <- cor(clean_data, use = "pairwise.complete.obs")
corrplot(f_cor,
         method="color",
         type = "upper")
```



```
#Fun tip: `apa.cor.table()` creates an APA formated correlation matrix and saves it to your computer
#apa.cor.table(physics, filename = "cor_table.doc")
```

UC SANTA BARBARA