214A: Intro to R

TA: Melissa Gordon Wolf

Fall 2019

1. Download R and R Studio

To use R, you first need to download R (https://www.r-project.org/) and then download RStudio (https://rstudio.com/products/rstudio/). The computers in the lab should already have these installed. Go ahead and open up RStudio (you will almost never open R - you will always interact with it through RStudio).

2. Setting a working directory

A working directory is just a fancy name for a folder on your computer. Wherever you save your R script will establish be your working directory. However, you can also set your working directory in R. You can just copy and paste this from your file explorer (PC), although you'll have to switch the back slashes to forward slashes. You can check what your current working directory is by using the function **getwd()**.

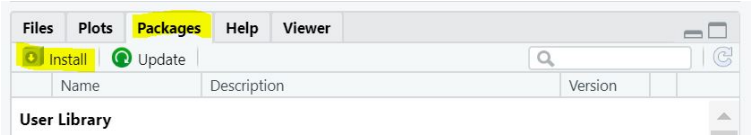
```
setwd("~/Users/Melissa/Documents/UCSB/214/Lab 8")
```

```
## Error in setwd("~/Users/Melissa/Documents/UCSB/214/Lab 8"): cannot change working directory
```

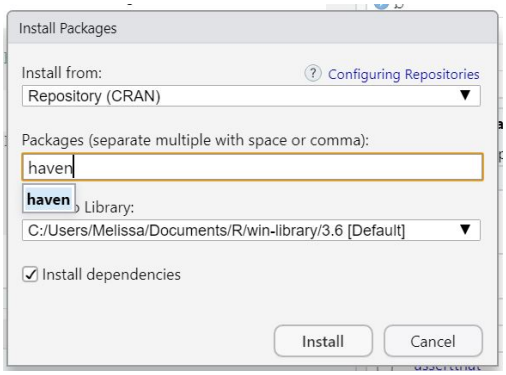
```
#ignore the error message
getwd()
```

3. Installing libraries and reading SPSS data into R

To get the dataset into R, we'll use the point-and-click interface in R. First, we need to install a package to read the data in from SPSS. On the right, click on **Packages** > **Install**.



Type in **haven**, check **Install dependencies** and press **Install**. Everything in R is case-sensitive, including library names.



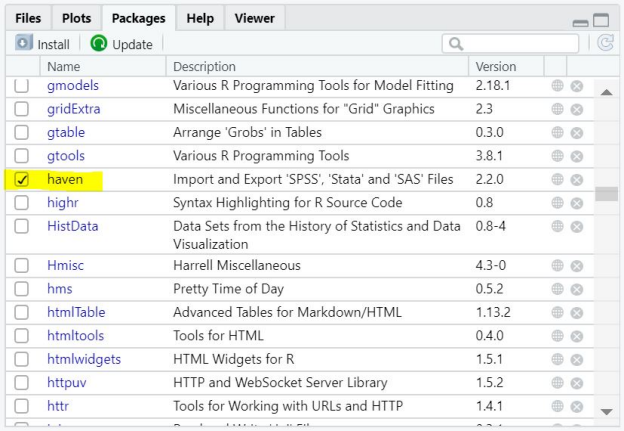
You can also do this in the R script:

```
install.packages("haven")
```

```
## Error in contrib.url(repos, "source"): trying to use CRAN without setting a mirror
```

```
#Remember to put the package name in quotes
#Ignore the error message
```

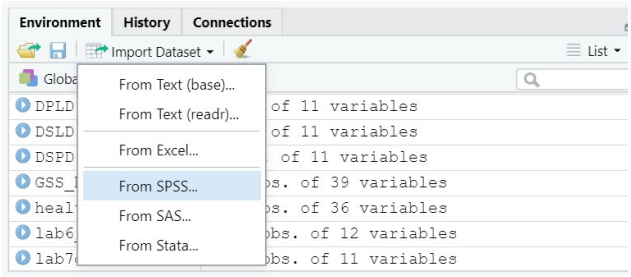
Once a library is installed, you'll never have to install it again. However, you will still have to **load** the library every time you open R. You can do this using the point-and-click interface in R. Scroll down under packages, and check the box next to **haven**.



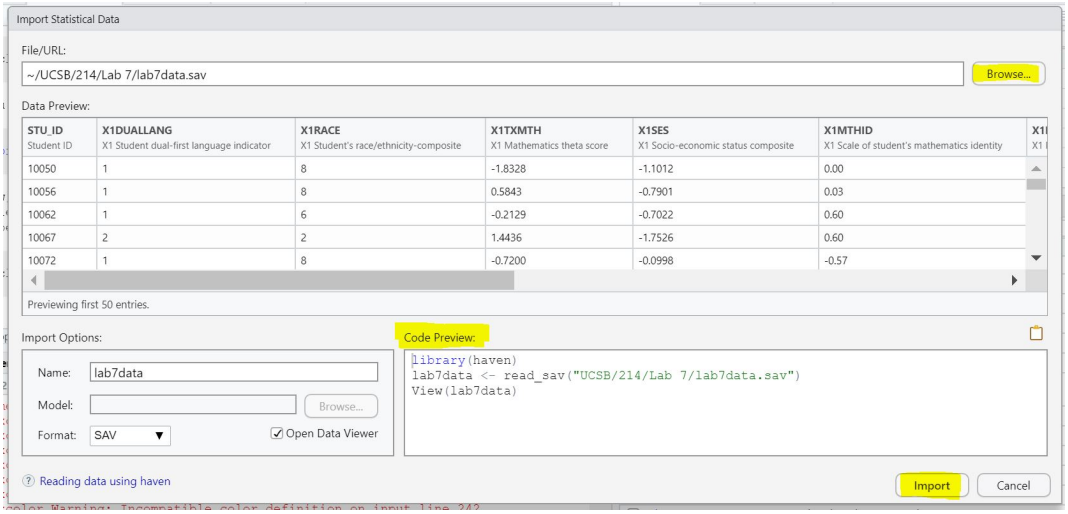
You can also do this in the R script:

```
library(haven)
```

Now, let's import the dataset. We'll use the same dataset from Lab 7. On the right, select **Import Dataset** and select **From SPSS**. As you will see, you can read several different file types into R.



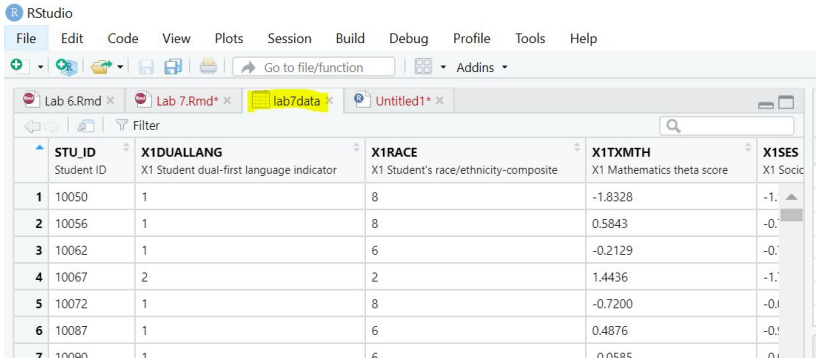
Select **Browse** and find the SPSS dataset. When you select it, you'll see it populate in the **Data Preview** window. Also, notice that R has automatically created the code you would need to read the dataset in under **Code Preview**. Select **Import**.



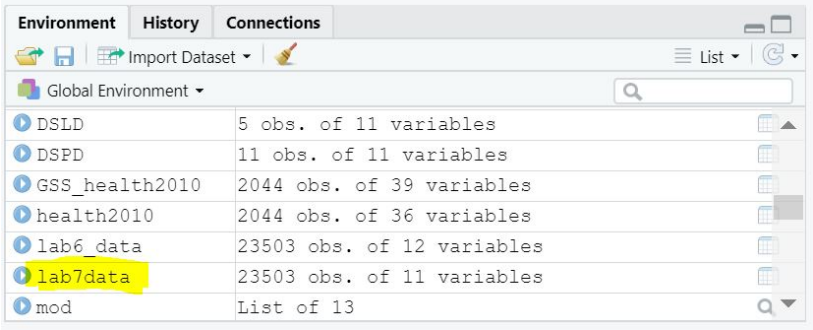
Alternatively, you can run the code that R generated (see below). However, I almost always use the point-and-click interface here.

```
library(haven)
lab7data <- read_sav("C:/Users/Melissa/Documents/UCSB/214/Lab 7/lab7data.sav")
```

Now, your dataset is in R! You'll notice that you can toggle back and forth between your R script and the dataset.



You can also locate (and open) the dataset from the Global Environment.



4. Renaming objects

Let's rename our dataset to something shorter, such as 'df'. 'df' stands for dataframe (this is how datasets are commonly referred to). You can either use an arrow (<-) or an equal sign (=). This simply tells R to

assign a new name to the dataset. Now, whenever we type "df", it will refer to this dataset. Try it!

```
df <- lab7data
df = lab7data
```

5. Understanding what the object is

You can use the function **class* to evaluate what kind of object you're working with. Different rules apply to different objects, and applying the wrong function to the wrong object can be a major source of error. This should be one of the first things you do to troubleshoot an error message.

```
class(df)
```

```
## [1] "tbl_df"     "tbl"        "data.frame"
```

6. Looking at the structure of your object or dataset

This can be overwhelming when you first look at it, especially when you read data in from SPSS, but it's basically the same thing as the **Variable View** tab in SPSS. It tells you about the various **attributes** of everything in the object (in this case, a data frame).

```
str(df)
```

```
## Classes 'tbl_df', 'tbl' and 'data.frame':    23503 obs. of  11 variables:
##  $ STU_ID     : 'haven_labelled' chr  "10050" "10056" "10062" "10067" ...
##   ..- attr(*, "label")= chr "Student ID"
##   ..- attr(*, "format.spss")= chr "A5"
##   ..- attr(*, "display_width")= int 5
##   ..- attr(*, "labels")= Named chr  "-7" "-8" "-9"
##   .. ..- attr(*, "names")= chr  "Item legitimate skip/NA" "Unit non-response" "Missing"
##  $ X1DUALLANG : 'haven_labelled' num  1 1 1 2 1 1 1 1 1 1 ...
##   ..- attr(*, "label")= chr "X1 Student dual-first language indicator"
##   ..- attr(*, "format.spss")= chr "F8.0"
##   ..- attr(*, "labels")= Named num  -9 -8 -7 1 2 3
##   .. ..- attr(*, "names")= chr  "Missing" "Unit non-response" "Item legitimate skip/NA" "First langua
##  $ X1RACE     : 'haven_labelled' num  8 8 6 2 8 6 6 3 6 5 ...
##   ..- attr(*, "label")= chr "X1 Student's race/ethnicity-composite"
##   ..- attr(*, "format.spss")= chr "F8.0"
##   ..- attr(*, "labels")= Named num  -9 -8 -7 0 1 2 3 4 5 6 ...
##   .. ..- attr(*, "names")= chr  "Missing" "Unit non-response" "Item legitimate skip/NA" "No bio/adop
##  $ X1TXMTH    : 'haven_labelled' num  -1.833 0.584 -0.213 1.444 -0.72 ...
##   ..- attr(*, "label")= chr "X1 Mathematics theta score"
##   ..- attr(*, "format.spss")= chr "F8.2"
##   ..- attr(*, "labels")= Named num  -9 -8 -7 -6
##   .. ..- attr(*, "names")= chr  "Missing" "Unit non-response" "Item legitimate skip/NA" "Component n
##  $ X1SES      : 'haven_labelled' num  -1.1012 -0.7901 -0.7022 -1.7526 -0.0998 ...
##   ..- attr(*, "label")= chr "X1 Socio-economic status composite"
##   ..- attr(*, "format.spss")= chr "F8.2"
##   ..- attr(*, "labels")= Named num  -9 -8 -7 -6
##   .. ..- attr(*, "names")= chr  "Missing" "Unit non-response" "Item legitimate skip/NA" "Component n
##  $ X1MTHID    : 'haven_labelled' num  0 0.03 0.6 0.6 -0.57 0.6 -0.57 0.6 0.03 -0.57 ...
##   ..- attr(*, "label")= chr "X1 Scale of student's mathematics identity"
##   ..- attr(*, "format.spss")= chr "F8.2"
##   ..- attr(*, "labels")= Named num  -9 -8 -7 -6
##   .. ..- attr(*, "names")= chr  "Missing" "Unit non-response" "Item legitimate skip/NA" "Component n
```

```
##  $ X1POVERTY  : 'haven_labelled' num  1 1 1 1 1 1 1 1 1 1 ...
##   ..- attr(*, "label")= chr "X1 Poverty indicator (relative to 100% of Census poverty threshold)"
##   ..- attr(*, "format.spss")= chr "F8.0"
##   ..- attr(*, "labels")= Named num  -9 -8 -7 0 1
##   .. ..- attr(*, "names")= chr  "Missing" "Unit non-response" "Item legitimate skip/NA" "At or above
##  $ X1FAMINCOME: 'haven_labelled' num  1 1 2 1 2 1 1 1 2 1 ...
##   ..- attr(*, "label")= chr "X1 Total family income from all sources 2008"
##   ..- attr(*, "format.spss")= chr "F8.0"
##   ..- attr(*, "labels")= Named num  -9 -8 -7 1 2 3 4 5 6 7 ...
##   .. ..- attr(*, "names")= chr  "Missing" "Unit non-response" "Item legitimate skip/NA" "Family incor
##  $ X2EVERDROP : 'haven_labelled' num  0 0 0 0 0 0 0 0 0 0 ...
##   ..- attr(*, "label")= chr "X2 Ever dropout"
##   ..- attr(*, "format.spss")= chr "F8.0"
##   ..- attr(*, "labels")= Named num  -9 -8 -7 0 1
##   .. ..- attr(*, "names")= chr  "Missing" "Unit non-response" "Item legitimate skip/NA" "No" ...
##  $ X3EVERDROP : 'haven_labelled' num  0 0 0 0 0 0 0 0 0 1 ...
##   ..- attr(*, "label")= chr "X3 Ever dropout"
##   ..- attr(*, "format.spss")= chr "F8.0"
##   ..- attr(*, "labels")= Named num  -9 -8 -7 -6 -4 0 1
##   .. ..- attr(*, "names")= chr  "Missing" "Unit non-response" "Item legitimate skip/NA" "Component no
##  $ X1TXMSCR   : 'haven_labelled' num  18.6 47.5 37.1 57.7 29.4 ...
##   ..- attr(*, "label")= chr "X1 Mathematics IRT-estimated number right score (of 72 base year items)"
##   ..- attr(*, "format.spss")= chr "F8.2"
##   ..- attr(*, "labels")= Named num  -9 -8 -7 -6
##   .. ..- attr(*, "names")= chr  "Missing" "Unit non-response" "Item legitimate skip/NA" "Component no
```

7. Variable names

Since that was overwhelming, let's get a list of all of the variabels in the dataset.

```
names(df)
```

```
##  [1] "STU_ID"      "X1DUALLANG"  "X1RACE"      "X1TXMTH"     "X1SES"
##  [6] "X1MTHID"     "X1POVERTY"   "X1FAMINCOME" "X2EVERDROP"  "X3EVERDROP"
## [11] "X1TXMSCR"
```

8. Variable labels or levels using SPSS data

The categorical variables in this dataset have labels, but they were labeled in SPSS. Thus, R does not have a record of the variable labels in the traditional sense, and we will have to use a function from the library **haven** to look at the labels of each variable. **Note: This will only work if you've read an SPSS dataset in using haven.** We use the $ sign to call a specific variable from the dataset (in this case, X1RACE).

```
print_labels(df$X1RACE)
```

```
##
## Labels:
##  value                                        label
##     -9                                      Missing
##     -8                             Unit non-response
##     -7                        Item legitimate skip/NA
##      0      No bio/adoptive/step-parent in household
##      1      Amer. Indian/Alaska Native, non-Hispanic
##      2                           Asian, non-Hispanic
```

```
##      3           Black/African-American, non-Hispanic
##      4                    Hispanic, no race specified
##      5                       Hispanic, race specified
##      6               More than one race, non-Hispanic
##      7 Native Hawaiian/Pacific Islander, non-Hispanic
##      8                            White, non-Hispanic
```

9. Variable labels or levels using other data and Base R

I created a simple dataset in R to show how the **levels** function in base R works. All of the data generation stuff falls between the two ## lines, so you can ignore that if you aren't interested in data generation quite yet.

```
################################################
  #Generate data
  x<-c(1,2,3,4,5)
  y<-c(1,2,3,4,5)

  #Combine to create dataframe
  df_a<-cbind(x,y)
  df_a<-as.data.frame(df_a)

  #Assign value labels
  df_a$y<-factor(df_a$y,
                 levels=c(1,2,3,4,5),
                 labels=c("Strongly Disagree",
                          "Disagree",
                          "Neutral",
                          "Agree",
                          "Strongly Agree"))
################################################
```

To view the levels of a variable in this new dataset using a Base R function, we can just use the **levels** function.

```
levels(df_a$y)
```

```
## [1] "Strongly Disagree" "Disagree"          "Neutral"
## [4] "Agree"             "Strongly Agree"
```

Let's compare the **structure** function, as well, for a non-SPSS dataset. The output is much simpler!

```
str(df_a)
```

```
## 'data.frame':    5 obs. of  2 variables:
##  $ x: num  1 2 3 4 5
##  $ y: Factor w/ 5 levels "Strongly Disagree",..: 1 2 3 4 5
```

That's it for today! Want to learn some R basics over the holidays? Check out the R advent calendaR!

https://kiirstio.wixsite.com/kowen/post/the-25-days-of-christmas-an-r-advent-calendar


25 days of
chRistmas:
an R advent
calendaR
Created by Kiirsti Owen