Working with Data

R Workshop - 8/14/2018

Introduction

Data are read into R using several functions written for this purpose. In my experience .csv format is the most well behaved. The `read.csv()` function is used to assign a data file to an object:
`data <- read.csv("`the path and file.csv`",`other arguments here`)`

Let's take a look at the help: `?read.csv`.

There is a more general function, `read.table`, that can read different types of files. But, it requires more programming. In addition, there is a package called `foreign` that has functions for reading files from other programs (e.g. `read.spss` reads SPSS files; `read.dta` reads Stata files, etc.). However, in my experience, if you are sending something from Stata to R, for example, it is better to just have Stata write a .csv file using: the `outsheet` function in Stata (type `help outsheet` in Stata for information).

Example Data File

Let's work through an example. Start by downloading the "R_Workshop_data.csv" file from this link on my website and save the file to a known directory. Open it and take a look at what is in the file. The file contains 52 individuals and 4 variables. The variables are: respondents id ("id"), a binary variable indicating whether the respondent is male or female ("male" where "1" is male), a measure of the respondent's age ("age"), and a measure of risky behaviors engaged in by the respondent ("risky").

Loading Data Files

Let's go ahead and import it into R:

First, set the directory where the file is by using the `setwd("/...")` function.

```r
#read in the data file.
data <- read.csv(
    "R_Workshop_data.csv", #the data file.
    header = TRUE, # tell R to read the first row as variable names.
    as.is = TRUE, # tell R to not make any conversions.
    na.strings = "." #tell R that missing values are periods.
    )
data #look at the object.
```

If you go to "Open Document" in the pulldown menu, and open the file this way, you can get an idea of how R views the file and why we give it particular instructions.

Note that we could skip the step of storing the .csv file locally and simply call file from the website url. For example:

```r
data <- read.csv(
    "https://www.jacobtnyoung.com/uploads/2/3/4/5/23459640/r_workshop_data.csv",
    header=TRUE, as.is=TRUE, na.strings="." #all the other arguments remain the same.
    )
data #look at the object.
```

Working with Objects

Now, we can work with the `data` object. As we saw before with matrices, we can index parts of the object by referring to the dimensions:

```
dim(data) # what are the dimensions of the object?
data[,1] #first column or first variable.
data[1,] #first row or first case.
data[,c(1,2)] #first two columns.
data[,1:2] #same thing as data[,c(1,2)].
data[1:25,] #just the first 25 cases.
```

We can also refer to the variables (i.e. columns) by their names enclosed in `""` or by using the `$` key.

```
data[,"id"] #returns the entire column named "id".
data$id #returns the column called "id" within the object "data".
```

Referencing parts of an object in this way makes it easy to execute functions. For example, the function `summary()` provides the range, mean, median, and info about missing values:

```
summary(data) #summarize all the variables.
summary(data$age) #just summarize age.
summary(data[,"age"]) #same thing, different syntax.
summary(data[,c("age","male")]) #just age and male.
```

Let's read into the data, but exclude the header to see how to assign names:

```
#toggle the header argument by setting it to FALSE.
data <- read.csv("R_Workshop_data.csv", header = FALSE, as.is = TRUE)
data     # look at the object, note the first row.
data[1,] # R has created the variable names for us.

#we can see this from the colnames() function.
colnames(data)

#change the names.
names <- c("id","male","age","risky") # create an object of names.
colnames(data) <- names #tell R what the column names are.
```

Obviously, if we had the names we would read them in with `header=TRUE` in the `read.csv` line.

Merging Data Files

We can merge two objects (or datasets) with the `merge()` function. For example, let's create two datasets where the cases do not match exactly (i.e. some cases are missing data on a merged variable) by using the `data.frame()` function:

```
# Create the datasets (note the difference between them).
data.a <- data.frame(id = c(1,2,3,4,5), age = c(10,12,14,15,11))
data.b <- data.frame(id = c(1,2,3,5)  , sex = c("m","m","f","m"))

class(data.a) # object "data.a" is of class data.frame.

?merge # note the different merging options.

# We can now merge these two datasets to get a single object.
```

```r
data.c <- merge(data.a,data.b,by.x = "id")
data.d <- merge(data.a,data.b,by.x = "id",all.x=TRUE) #note difference.
```

Exporting Data Files

We can also write data files out for analysis in another program. For example, say we have created a variable in our data set that we could only create in R and we want to export the file to another program. For .csv files, we can just use the `write.csv` function. As with reading files, there is a more general function, `write.table`, that can write different types of files (see `?write.table`). For `write.csv` we just tell it what object we want to write out to a file and the file name:

First, set the directory where the file will go by using `setwd("/...")`. Then:

```r
write.csv(data,"New_R_Workshop_data.csv") #file path can be added also.
```

Saving Data Files

Recall that to save the workspace use the `save.image()` function. This function requires a file path, a file name, and the extension ".RData" which is the format for an R workspace file. Since our imported data is an object (i.e. `data`), it will be saved to the workspace:

First, set the directory where the file will go by using `setwd("/...")`. Then:

```r
save.image("data.RData") #file path can be added also.
rm(list=ls()) # clear the workspace.
ls() # check the contents of the workspace.
load("data.RData") #load the workspace that was previously saved.
```