Third_Excercise (reading ugly spreadsheets)

Joe

2018-07-26

File/Import Dataset command versus read.csv()

There are many ways to import data into R Studio. R Studio has a graphic interface for importing files such as Excel files. The graphic interface may be nice for some simple purposes, but it has drawbacks. The biggest drawback is that it tempts programmers to create scripts that do not have reproducible output. If someone loaded the incorrect dataset using the graphic interface, it may be very difficult to recognize this error after the fact.

There are also packages which allow users to import Excel files directly. I will avoid those methods here because there have been issues reported about its cross-platform compatibility.

I want to discuss the following approach for importing Excel data into R.

1. Export your data into comma separated values (csv) file.
2. Inspect the csv file to see how R will go about parsing your file. The function read.csv() will interpret each comma as separating columns of your data, and lines will (typically) be interpreted as a row.
3. Import data into R Studio using csv.read
4. Inspect your data. Pay special attention to your variable types and the existence of NA values (I will discuss what NA values are in a momment).

This four-step procedure is iterative. With messy spreadsheets you will often have to repeat the procedure. When a problem is encountered, it will be up to the user to decide whether it is better to (a) clean up the Excel file to make it more machine readable or (b) to modify R code.

To begin, inspect ‘Garbage Excel grades file.xlsx’. R Studio will have a lot of difficulty reading this document. It will not, by default, understand that the ‘Grade Total’ row does not correspond to student data. It will not know that the final two columns, which just refer to an overall grade breakdown, are not data columns. One particularily troublesome problem is that the header ‘Garbage Excel grades file’, will be interpreted as the first row of data. The easiest thing to do is simply remove these elements from your Excel file now.

Proceed to step 2 and export the Garbage Excel grades file.xlsx to a csv. Inspect the csv using a text reader such as TextEdit (Mac) or Notepad (PC). Each comma will be interpreted as a new column in R.

(Step 3) Try to import the csv using csv.read. First use the file browser to set the current working directory of R to the location of the csv. Afterwards, run

```
Data=read.csv('Garbage Excel grades file.csv', header=FALSE)
View(Data)
```

The data frame Data might look somewhat like the Excel sheet, but notice that the variables are not correclty named (read.csv has interpreted the column headers as rows of data). This is easily fixed by looking at the ? read.csv. There you will discover that the following parameter will tell R to interpret the first row of the file as

containing column names

```
Data=read.csv('Garbage Excel grades file.csv', header=TRUE)
View(Data)
```

This import of the Data looks better, but remember to inspect the classes of each variable. For example, attendance.1 and attendance.2 are not the same variable type.

```
class(Data$Attendance.1)
```

```
## [1] "integer"
```

```
class(Data$Attendance.2)
```

```
## [1] "factor"
```

The reason why is apparent from the original Excel sheet. Josiah was excused for missing the day corresponding to attendance.2. The string ‘Excused’ has forced R to interpret the entire column of data as a factor, rather than a integer. There is also something interesting about the integer attendance.1. Consider that

```
mean(Data$Attendance.1)
```

```
## [1] NA
```

returns NA. If you type in ?NA you will see that NA refers to missing values, and the mean function will (by default) return NA unless it is given a complete vector of numbers. We had better go into the excel file and replace these empty cells with a score of 0.

can also force read.csv to treat the string ‘Excused’ as a missing value.

```
Data=read.csv('Garbage Excel grades file.csv', header=TRUE, na.strings='Excused')
View(Data)
```

This will allow us to calculate an individual’s attendance score while ignoring the missing values.

```
mean(Data$Attendance.1, na.rm=TRUE)
```

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

Other sorts of data may appear in spreadsheets. An ‘Inf’ value may have been produced by a calculation error, for example. Another common value which could appear in data is NaN, which refers to ‘Not a Number.’ This refers to meaningless, rather than missing data. The distinction is useful as you probably want to treat these sorts of data differently. You can also distinguish between NA values (which might refer to missing data points), and the NULL value. Only the former are useful for labelling the missing observations in a dataset.

```
length(NULL)
```

```
## [1] 0
```

```
length(NA)
```

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

Excercise

1. Inspect Data for any additional weirdness. HINT: Check the attendance.6 column. Fix this problem.
2. Fix the problem with the term project column either by modifying R code or by changing the Excel files.
3. The final exam is worth 15%. Weight this column by 0.15.
4. One way to learn to code is by working with the code of others. Download the script below, SumStudentRows.R, from GitGub and provide more informative comments. This script will require three variables (Attendance, Attendance.1, and Attendance.2). All must be numeric.

```
#Script: SumStudentRows.R

#Goal: Sum each student's attendance grade

# idx plz
AttendanceDayOneIDX <- which(names(Data)=='Attendance')
AttendanceDayThreeIDX <- which(names(Data)=='Attendance.2')

# calc sumz
AttendanceNumbrz=rowSums(Data[,AttendanceDayOneIDX:AttendanceDayThreeIDX],na.rm=TRUE)

# Get grdzzz
data.frame(Data$FirstName,AttendanceNumbrz)
```

5. Verify that the script SumStudentRows.R completes its goal.