Lab 4: Importing Data

Introduction

In this lab we learn how to import new data from Excel or other sources into RStudio. Along the way we will also get more practice with RMarkdown reports and statistical analysis commands.

Overall Goals

In this lab we will:

- learn how to import new data from the common CSV data format,
- practice creating a report in RMarkdown,
- review the common analysis techniques we have seen in previous labs,
- learn how to draw scatterplots.

New R commands introduced in this lab: `$`, `c`, `ordered`, `print`

Creating an RStudio Project

We start similarly to the last lab. We will start a new project based on a prepared repository.

- In your web browser, open this Lab 4 assignment from the HanoverStatsLabs website. You will need to copy-paste a link from it in a few steps.
- In RStudio, go to File > New Project > Version Control > Git. You will then see three textboxes.
- Verify that the directory listed in the third textbox is the one you created for your labs. If not, then click the **Browse** button to display a list of all the folders in your home directory on vault and choose the correct one.
- Copy the following URL from the HTML page and paste it into the "Repository URL" field but **DO NOT** press <enter> right away.

  https://github.com/HanoverStatsLabs/Lab-Data-Import.git

- Move to the middle textbox and enter a name for the project directory. `Lab_4_yourname` might be a reasonable choice.
- Click on **Create Project** to finalize the setup process.
- Your Files pane should now show the provided files, in particular a file titled `Lab4Report.Rmd`. **Click the file name to edit the file.**
- Recall that we will mostly be editing this report file, rather than working directly in the console. You may want to make the report window larger than the console window.
- **DO NOW**: Edit the header section to add your own title, name, and date.
- Before moving on, make sure to **run the chunk** which contains the instruction `library(hanoverbase)`.

Importing Data

In the previous labs we have prepared the data for you in R's built-in system. But data you find in the real world is not typically in this form. It may be an Excel file, a comma-separated-format file (CSV) or one

of many other formats. RStudio has built-in ways to read CSV and Excel files. Some other more obscure formats may need more work and specific instructions.

For now we will learn how to import a CSV file. This file currently lives on a webpage, at the following URL: https://hanoverstatslabs.github.io/resources/datasets/driving.csv

- Download the file from the above page to your computer. This typically entails right-clicking the link and choosing something like **Save Link As**.
- Now upload the file you just downloaded into your project folder. To do this:
  - In the **Files** pane (lower right), click the **upload** button to start the Upload Files dialog.
  - Navigate to where you saved the file and choose the file.
  - Click **OK** to finish the upload. If this was successful, you should now see driving.csv in your Files pane.

Now we want to read the file into our R report/environment. To do that, click the filename in the File pane and choose "Import Dataset".

This data contains information recorded over the Fall 2007-2008 term when one of the faculty was living in Louisville and commuting to Hanover. Each row correponds to a one-way trip and contains details of that trip such as date, direction, departure and arrival times and car mileage at the start and end of the trip.

- Checking for misalignments:
  - Sometimes programs misidentify the column separators and may misalign some columns.
  - Scroll through the file to make sure that what shows up in each column makes sense for that column (for instance does it look like the columns got shifted?). **HINT** If you can only see one row of data at a time, make the font smaller in the Import window (use your browser zoom features, if any).
- Setting the column types:
  - For each column in the data, RStudio needs to know the type of values for that column (integer, character, date, etc.). The system is usually smart enough to do the right thing.
  - For this data set, we will let RStudio "guess" the types as best it can.
  - If you ever find that RStudio did not guess the type correctly, you can use the column heading in the import dialog to specify a type (or have RStudio skip a column entirely if you don't really need it).
- Copying the appropriate code:
  - Copy most of the code that is in the "Code Preview" section at the lower right of the Import window. You don't need the View line for now.
  - In your report, **add a new code chunk** for the import. Put it below the existing code chunks. Paste the code you copied into this chunk. The chunk will look something like this:

```
library(readr)
driving <- ...
```

  - Edit the long filename string, which probably with something like "~/..." to leave only the actual file name there, "driving.csv".
  - Set the chunk options (gear button to the right of the chunk) to not show warnings, not show messages, and to output "Show nothing (run code)".
  - Run the chunk. Then run a suitable View command in the console.
  - This code chunk ensures that when anyone compiles the report, the dataset will be loaded and ready to use.

Take a moment to look at the two lines of code in your new chunk. The first line uses the `library` command to load a new package (also called a library) named `readr`. This package contains useful functions for loading new data. The second line uses the `read_csv` command from this package to load the data and store it in the name `driving`. You can access this data in the rest of your report via this name. You should be able to see the `driving` entry in the Environment pane.

Before we move on, there is one more step we must take with the data. The `weekDay` column is a categorical variable coded as 1 through 7 (Monday through Sunday). We must tell RStudio about that, as right now it treats it as quantitative (numerical). You will need to run the following commands (make sure to include them in the code chunk you created earlier and to **run the chunk**). Recall that you can copy these lines from the corresponding HTML version of the instructions.

```
days <- c("Mon", "Tue", "Wed", "Thu", "Fri", "Sat", "Sun")
driving$weekDay <- driving$weekDay %>% ordered(labels=days)
```

Look at the data table after running the chunk. You should see the 3-letter abbreviations in the weekDay column instead of number codes.

The two lines we used above introduced some new features:

- The first is the use of the `c` command, to "concatenate" items into a list. The result is technically called a "vector" in R.
- The second line used the `ordered` command to turn the weekDay variable from the data into a categorical (ordinal) variable. We had to provide it the variable's name (`driving$weekDay`) and a list of the labels to use (`labels=days`).
- This also introduced the "dollar sign" syntax, which has the general form `<dataset>$<columnName>` and it is used to access an individual column in a dataset. It is not needed when we use the `data=<dataset>` syntax, but not all commands can use that syntax.
- In general you only need the dollar-sign syntax when changing a variable in a dataset, or when you want to create a new variable in a dataset.

Statistical Investigations

Important formatting note: As you work through this lab assignment, answer the questions which are posed by typing into your R Markdown report. You should use section headings, subsection headings, numbered lists, unnumbered lists, etc. to organize your work and make it easy for your reader to follow. You can refer to the Basic Cheatsheet[1] for help with creating these effects.

R Chunks: As you create graphs and numerical summaries, be sure you are adding these commands to your R Markdown report by creating R chunks for them. While you do have some freedom regarding how you structure your report, you should typically have sections that contain a heading, a brief introductory paragraph, and an R code chunk for any relevant computations, followed by (numbered) answers to the questions.

Knit Early and Often: Be sure to knit frequently and examine the resulting output document. Does it look the way you wanted it to? (If you do not knit often, you should at least **save** often.)

Refer to the R Cheatsheet and Previous Labs for Examples: We don't expect that you have memorized the intricacies of R syntax at this point. Be sure to refer to the provided R Cheatsheet for help.

[1] ../cheatSheet.html

Week Days

The first set of questions relates to the various days of the week present in the dataset. You should make both a frequency table (tally) and a barchart of the weekDay variable. Hint: Refer back to a previous lab for an example of piping a tally into a barchart.

1. The instructor was teaching only four days a week. Which do you think is the workday on which the instructor was not teaching? Explain.
2. Looking at the frequencies in the tally, explain why most of the frequencies are even numbers. How do you explain those that are odd numbers?
3. What would you expect the typical value for the workday frequencies to be, based on the Hanover College academic calendar (13 full weeks in Fall term, plus a week of exams)? To what extent does that match the data? How do you explain the workday values that deviate from the expected typical value?

Note: Don't forget to knit often!

Distance Traveled

In this set of questions we will investigate the distribution of the miles variable, which records the miles traveled per trip as a difference of the mileages recorded by the car's odometer. You will likely need several outputs:

- a summary (favstats) of the miles variable,
- a histogram of the miles variable,
- a histogram of the miles variable, but where the dataset has been filtered to only contain the rows with miles <= 48.
- a boxplot of miles against direction (possibly filtered as above), and
- a summary (favstats) of the miles against direction .

Here are the questions for you to answer.

4. Looking at the favstats summary and the first histogram, what is the typical driving distance for the instructor? Does that roughly match the distance between Louisville and Hanover?
5. The distribution should appear skewed to the right with a very sharp drop-off to the left, and some high outliers to the right. Explain why we might expect these three behaviors in this dataset.
6. When focusing on the values below 48 miles in the second histogram, you should notice that what appeared to be one mode is actually two modes, of roughly the same frequency, about half a mile apart from each other. Provide a plausible explanation for the presence of these modes that would account for their similar frequencies. (The boxplots and last summary may also be helpful here.)

Driving Times

The (elapsed) time variable is calculated as the difference in minutes between the instructor's departure time and arrival time. We should create the following summaries:

- A favstats summary and a histogram of the time variable.
- Two xyplots (scatterplots) of the time variable against the miles variable. In the first plot we exclude three particularly unusual/incorrect time values, and in the second plot we restrict further to the "miles < 50" region containing the typical distances. Use the following commands for this (may want to copy-paste them rather than type them in):

```
scatter1 <- xyplot(time~miles,
    data=driving %>% filter(time <= 90 & time > 40))
print(scatter1)

scatter2 <- xyplot(time~miles,
    data=driving %>% filter(time <= 90 & time > 40 & miles < 50))
print(scatter2)
```

  Note that we actually "stored" the graphs in names, and referred to those names to display the graphs with the print commands. In general, you can store graphs this way and process them further in later commands.

Here are the questions for you to answer.

7. Discuss the distribution of the time variable, including any unusual observations and plausible explanations for the various aspects of the distribution.

8. Describe the relationship between the time variable and the miles variable. What is the overall pattern, how does it make sense? Are there deviations? Is there a clear linear association?

9. For this question, we are enhancing the scatterplots from above by adding straight lines at the 55 mph speed limit. Note that 55 mph corresponds to 60 minutes per 55 miles, making the slope of the line (b) equal to 60/55:

```
ladd(panel.abline(a=0, b=60/55), plot=scatter1)
ladd(panel.abline(a=0, b=60/55), plot=scatter2)
```

   Notice that if a point is on the line, that trip has an average speed of 55 mph.

   a. There is a trip which took 79 minutes and 47.4 miles. Find that point on the scatterplot. What was the average speed for that trip, in miles per hour? Is the point for that trip below the line, or above the line?
   b. What does it mean when the point for a trip is below the line?
   c. Where are most of the points in relation to the line (below vs. above)? What does this tell you about the instructor's driving habits?

10. We will now color the points in the scatterplot based on the direction of travel (we will learn more in a future lab about the special graph settings used in this example):

```
xyplot(time~miles, data=driving %>%
    filter(time <= 90 & time > 40 & miles < 50),
    groups=direction, pch=19, auto.key=TRUE)
```

    What pattern do you see in the colors? How do you explain it?

Submissions

- Make sure to knit one last time, then download the Lab4Report.Rmd file: in the **Files pane** (lower right), click the checkbox for the RMD file, then choose More > Export ... > Download. *Do the same for Lab4Report.html.*
- **Submit both files via Moodle**.