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 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.
- Enter the following URL into the "Repository URL" field but **DO NOT** press <enter> right away. Either type the URL by hand (no spaces) or copy it from the HTML version.

```
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. R can read files directly from a URL, so there is no need to download the file.

- Go to File > Import Dataset > From CSV ...
- Paste in the following URL: https :// hanoverstatslabs . github . io / resources / datasets / driving . csv
- Click the **Update** button to the right of the URL and you should see the data in the Preview Window. We now need to make sure the data is being read correctly before doing the actual import.

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. **HINT** If you can only see one row of data at a time, make the font smaller in the Import window.
- 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).
  - Click the **Import** button.
- Copying the appropriate code:
  - If the import went okay, you should now see the dataset in the data view. If it looks OK, then go to your report and add a new code chunk below the existing code chunks.
  - 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)".
  - From the Console, one at a time, copy the two lines that were run in the console (from clicking the Import button). Do not copy the > prompt. The first line to be copied is library (readr). The second starts with driving <−.
  - This code chunk will ensure 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 loads a new package ("library") called readr. This package contains useful functions for loading new data. The second line uses the read_csv method from this package to load the data and stores it in the name driving. You can access this data in the rest of your report via this name.

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. For that 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).

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

If you look at the data table after running the chunk, you should see the 3-letter abbreviations in the weekDay column instead of number codes.

Sidebar: Workflow practices

Working in the R Markdown document as opposed to working in the console can get confusing. And complicated commands have many steps and many places where things can go wrong. A simple workflow to follow is the following:

1. Make a code chunk using the **Insert** pulldown.
2. Enter a single line of code into the chunk and execute it via the "Run > Selected Line" from the **Run** pulldown. This will run just the current line in the console.
3. Make sure that line's output is correct before proceeding to the next line. If that line was an assignment then there is no output by default, but you can use the Environment pane to look at the assigned value or add a new line of code asking for that assigned value.

For example, when trying the above commands, we may do the following (**DO NOT** really do this now, since we've already run these commands; just think about it):

1. Type the first command, (`days <- ...`) and run it.
2. Add a second command saying just `days` and run it. This will print out what `days` is. You can then remove that second line. Alternatively you can look in the Environment pane, to make sure that the `days` value looks correct.
3. Repeat for each line.

Explanation of the three lines

The three 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 this syntax when changing a variable in a dataset, or want to create a new variable in a dataset.

Statistical Investigations

Important formatting note: As you work through this lab assignment, 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 R Markdown Syntax Sheet[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?

Week Days

The first set of questions relates to the various days of the week present in the dataset. You should do both a frequency table (tally) and a barchart of the weekDay variable.

1. The instructor was teaching only four days a week. Determine the workday on which the instructor was not teaching. Explain.
2. Looking at the weekday frequencies, explain why most of the frequencies are even. How do you explain those that are not?
3. What would you expect the typical value for the workday frequencies to be, based on the Hanover College academic calendar? To what extent does that match the data? How do you explain the workday values that deviate from the expected typical value?

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 median or summary of the miles against direction .

Here are the questions for you to answer.

4. What is the typical driving distance for the instructor? Does that roughly match the distance between Louisville and Hanover?

[1] ../rmarkdownBasics.html

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, 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.

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:

```
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 <
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? How strong is the relationship?

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)
```

   How are the various points in the graph positioned relative to these lines? 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 about the special graph settings used in a future lab):

```
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?