Lab 6: More Fun with Graphs in R

Introduction

Overall Goals

In this lab we will:

When we plot integer data in a histogram, need to take care with the bins to avoid "false" modes.

For a paneled histogram, we want to control the layout of the panels for easier comparison.

Adding a "smooth fit line" (curve) to a scatterplot.

Controlling the symbol/size/fill for scatterplot dots.

Controlling line width/type/color.

Mosaicplot

New R commands introduced in this lab:

Make Your Own Project in RStudio

Instead of starting from a prepared project, as in previous labs, you will now create your own project.

- In RStudio, go to File > New Project > New Directory > Empty Project
- Make sure the parent directory (second textbox) is the folder where you want to keep your projects for this class. Use the browse button if necessary.
- Enter a name for the new directory; it is good practice to avoid spaces in your file names by using underscores in place of spaces. For example, Lab_6 would be a good name.
- Click Create Project.
- In the Files pane, you should now see a file Lab_6.Rproj. This is the project configuration file, and you don't need to do anything with it.

Now you need to start a new RMarkdown report file.

- Go to File > New File > R Markdown....
- In the Document tab of the resulting dialog, give your report a title (Lab 6 Report) and put your name in the Author textbox. Keep HTML as the output format. Click OK.
- You should now see your new RMarkdown document at the upper left; **save** it. In the Save File dialog, enter the file name with no spaces: Lab6Report. (Do not use a filename extension.)
- Everything below the provided code chunk is boilerplate and **should be removed**. Do so now.
- **Note**: The top-level section heading (one #) is already in use in this document for the report title. Use second-level headings (##) for the main sections of the report. (If you need subsections, use ###.)
- Use the Insert pulldown to add a new R code chunk. Add the command library(hanoverbase). Use the chunk options dialog to disable warnings, disable messages, and "show nothing (run code)".
- **Run the chunk** which you just created.

Import and View Dataset

- Make a new R code chunk and use the `data(brfss)` command (see Lab 1, for example) to load the dataset named `brfss`; **run this chunk**.
- Use the `View` command *in the console* to see the data in the Preview window. **Note**: Do not put the `View` command in an R chunk, it will prevent your report from knitting/compiling. Similarly, the `?` command should not be put into your R Markdown report.
- This data should be familiar from the previous lab.

You are now ready to start working on your report. The sections below give you questions to answer and commands to try. Here are a few reminders:

- **Important formatting note**: As you work through this lab assignment, answer the questions which are posed by typing into your R Markdown report, using formatting elements to make the report easy to read. See: R Markdown Syntax Sheet[1]
- **R Chunks**: As usual, create R Chunks (use the `Insert` pulldown) for your R commands. Use the R Cheatsheet for help as needed.
- **Knit Early and Often**

Height / Controlling Histogram Bins

We will start off by looking at the `height` variable. Use `?brfss` in the console to remember how this variable is measured.

1. Calculate a summary (`favstats`) and draw a basic histogram of the `height` variable using the `brfss` data. Does this default histogram give a good view of the data? Explain. Give at least one fact you can discern from the graph, and a concern you have about how the graph is scaled.

Because of the presence of outliers, the histogram is forced to show a wide range on the scale, so we do not get a very good view of the non-outliers. In order to get a better view, we will create a smaller dataset which removes the outliers. We will keep the middle 99% of the values, using the quantile function to find the lowest half-percent and highest half-percent in the `height` data.

2. Pipe the brfss data through a filter to remove the `height` outliers. First find the height cutoffs for the middle 99% and then use those cutoffs in a filter to form a subset of the `brfss` data:

```
lowCutoff <- quantile(~height, data=brfss, na.rm=TRUE, probs=.005)
highCutoff <- quantile(~height, data=brfss, na.rm=TRUE, probs=.995)
heightSubset <- brfss %>%
    filter(height >= lowCutoff & height <= highCutoff)
```

   Now draw the histogram with the filtered data. Note: Instead of letting the histogram breaks be set by default, we will ask for 20 breaks. Later on, we'll take finer control of the breaks.

```
histogram(~height, data=heightSubset, breaks=20)
```

[1] ../rmarkdownBasics.html

a. Describe the overall shape of the distribution.
b. Make a tally of the sex variable for the heightSubset data. What do you learn? How might this help to explain the shape of the histogram?

3. When working with integer data such as height, we have to take care with our histogram bins / number of breaks, since integer data can interact badly with the breakpoints for creating the histogram bins.

   For example, if we have bins of width 1.4 with the first bin starting at 0, then the breaks are at 0, 1.4, 2.8, 4.2, etc. Notice that some bins have two integers (2.8 to 4.2 has 3 and 4) while others have one integer (1.4 to 2.8 has only 2). So the frequencies of the bins are not easily comparable.

   Also, if we have bins of width less than 1 then some bins will be empty just because they do not contain an integer within their bounds.

   To avoid these problems, we can put breaks specifically at all the ".5" marks on the axis. We use the seq command to create a sequence of numbers with a given start value, stop value, and step size.

```
myBreaks <- seq(from=lowCutoff - 0.5, to=highCutoff + 0.5, by=1)
myBreaks          # this just prints the sequence of numbers
histogram(~height, data=heightSubset, breaks=myBreaks)
```

   Describe the height distribution: overall shape, center (from favstats), spread (for typical heights), number/location of modes (if any).

Height and Sex / Paneled Histogram

4. Because of the difference in average heights for males as compared to females, we might have expected the histogram to be clearly bimodal. Indeed, with a boxplot we can see this difference. Draw a bwplot of sex~height now.

   As a companion to the bwplot, let's also make a histogram which is paneled by sex. Notice the use of the formula ~height|sex for height versus sex, and the layout=c(1,2) option for forcing the panels to line up vertically (1 column, 2 rows):

```
histogram(~height|sex, data=heightSubset, breaks=myBreaks, layout=c(1,2))
```

   Use the boxplot and the paneled histogram to explain why the original histogram does not show a clear bimodal pattern.

Height and Weight / Smooth Fit Curve and Line Options

In this section we investigate the relationship between height and weight for our respondents.

5. Make a scatterplot of weight versus height for the brfss dataset.

   a. Why do the dots make vertical stripes on the scatterplot?
   b. Describe the overall pattern in the data. Do you see a strong relationship between height and weight for these subjects? Explain.

6. One way to summarize the data in a scatterplot is to add a smooth fit line (notice that the "line" in this context might be curved). Add the fit line with the panel.loess command. Note that lwd= is an option for setting line width.

```
ladd(panel.loess(x, y, col="darkgreen", lwd=2))
```

   The smooth fit line is almost straight, indicating some sort of linear association. Note: the *strength* of the linear relationship cannot be seen in this graph.

7. In addition to the smooth fit line, we can show the linear regression model:

```
ladd(panel.lmline(x, y, col="red", lwd=2))
```

   Describe the direction of the linear association. Explain.

TODO: more options: pch, cex

.....

=======

Mosaicplot

12. As an alternative to the stacked bar graph, we can draw a mosaicplot. A basic 2-variable example shows the relationship between income and health for the brfss participants:

```
healthVsIncome <- tally(~income+genhealth, data=brfss,useNA="no")
healthVsIncome %>% mosaicplot(color=brewer.pal(5,"RdPu"))
```

    a. What do you learn from this plot?
    b. What is the meaning of the varying bar widths in the mosaic plot?