Lab 6: More Fun with Graphs in R

Introduction

In this lab we will learn a couple more techniques for working with graphs and variables in R. In particular we will see how to create new variables from existing ones.

Overall Goals

In this lab we will learn how to:

- create a new project and start a new R Markdown document in RStudio,
- compute quantiles (percentiles) for a variable,
- control the bins in a histogram,
- control the panels in a paneled histogram,
- add a "smooth fit curve" to a scatterplot to look for the overall pattern,
- add a regression line to a scatterplot,
- control the size and color of added lines,
- compute new scalar variables using an equation,
- compute new categorical variables by binning the values of a scalar variable.

New R commands introduced in this lab: `quantile`, `cut`, `seq`, `panel.loess`, `panel.lmline`, `with`

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 *first* 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.
- 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)
```

[1] ../rmarkdownBasics.html

If this was successful, you should see a heightSubset entry in the Environment pane, showing around 99% of the initial observations.

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)
```

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 whole number data such as height, we have to take care with our histogram bins / number of breaks, since whole number 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 whole numbers (2.8 to 4.2 has 3 and 4) while others have one whole number (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 whole number 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)
```

   Notice how the whole numbers on the x-axis are now centered below their corresponding bars.

   Describe the height distribution: overall shape, center (make a 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, using the heightSubset data.

   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 heightSubset 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.

    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.

Body-Mass Index / Creating New Variables

We will now learn how to create new variables out of existing variables. We will learn about two specific methods:

- Creating a new scalar variable from an equation using existing scalar variables.
- Creating a new categorical (factor) variable from an existing scalar variable specifying breakpoints.

Creating a New Scalar Variable

We start with creating a new variable that measures the **body mass index**[2], typically abbreviated as BMI. The formula is "weight over height squared". If pounds and inches are used, then a conversion factor of 703 must be applied. We will use the heightSubset rather than the whole brfss.

We use the dollar sign notation here to add a new column to the data. We also introduce the with command, which allows us to refer to the columns of a particular dataset in the equation:

```
heightSubset$bmi <- with(heightSubset, weight / (height^2) * 703)
```

If this has succeeded, you should now see a new column in the heightSubset, called bmi.

7. Calculate a summary (favstats) and draw a basic histogram of the bmi variable using the heightSubset data. Describe the distribution. What can we say about the BMI values of the respondents?

[2] https://en.wikipedia.org/wiki/Body_mass_index

Creating a New Categorical Variable

We will now break the BMI values into categories according to a standard correspondence described by the World Health Organization. Even though the WHO defines eight different BMI ranges, we will simplify it to four:

Category	BMI Range
Underweight	Below 18.5
Normal	18.5–25
Overweight	25–30
Obese	Above 30

To do this we use the new command cut. We have to specify where to break the values, and we can optionally give labels to the ranges (the default being an interval notation like (25,30]).

```
heightSubset$bmicat <- heightSubset$bmi %>%
    cut(breaks=c(0, 18.5, 25, 30, 100),
        labels=c("Underweight", "Normal", "Overweight", "Obese"))
```

After running this command, you should see another new column titled bmicat in the heightSubset data.

8. We now have a categorical variable called bmicat. Do a tally and barchart (you can also do a pie chart) of this variable. What can we say about the respondents' BMI values from this?

9. We would like to investigate how the BMI might be related to the general health of the respondents. Make an initial prediction as to how you expect the BMI level to be reflected in the general health.

10. In order to properly answer the previous question with the provided data, we will need to create a stacked bar chart of the BMI and general health. To do this you will need to follow similar steps to those used when we looked at healthVsExercise and healthVsIncome in the previous lab. Produce a stacked bar graph with *one bar for each BMI category* and *stacks determined by the genhealth variable*. The first command for this would be:

    ```
    healthVsBmicat <- tally(~genhealth|bmicat, data=heightSubset,
          format="percent", useNA="no")
    ```

    What do you see in the stacked bar graph? How does it compare with your prediction?

Submissions

- Make sure to knit one last time, then download the Lab6Report.Rmd file: in the **Files pane** (lower right), click the checkbox for the RMD file, then choose More > Export ... > Download. *Do the same for Lab6Report.html.*
- **Submit both files via Moodle**.