Lab 5: Customizing Graphs in R

Introduction

In this lab we learn how to

Overall Goals

In this lab we will:

New R commands introduced in this lab:

Start a New Project in RStudio

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

- In your web browser, open this Lab 5 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.
- Paste this URL in but **DO NOT** press <enter> right away:

```
https://github.com/HanoverStatsLabs/Lab-Customize-Graphs.git
```

- Press Tab to go to the next field — it should auto-complete the directory name. Change it, if you want, to lab_5_yourname.
- Make sure the parent directory is the folder where you want to keep your projects for this class.
- Click Create Project.
- Your Files pane should now show the provided files, in particular a file titled Lab5Report.Rmd. Click the file name to edit the file, and then run all the R chunks.
- 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.

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.
- 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
```

- Click on **Create Project** to finalize the setup process.
- Your Files pane should now show the provided files, in particular a file titled Lab5Report.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)`.

Import and View Dataset

- In your R Markdown report, make a new R code chunk. In the new chunk, use the `data` command 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.
- You should see that the data has 24 variables and over 1.3 million rows.
- Use the `?` command *in the console* to view information about the brfss data file (the "codebook"): `?brfss`

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 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?

General Health

1. Make a tally and a barchart (as you did for WeekDays in a previous lab) for the `genhealth` variable. Which category of the variable is the most frequent?

2. It would be useful to have **percentages** rather than raw counts in our tally. Add a *command option* by inserting the code `, format="percent"` before the closing parentheses of the tally command, like this:

```
tally(~genhealth, data=brfss, format="percent")
```

   What percent of respondents report being in either "Very good" or "Excellent" health? What percent of respondents did not answer the question (NA)?

[1] ../rmarkdownBasics.html

In the barchart, we like to have control of the **bar colors**. This is controlled with the `col=` option. For example, add the option `col="green"` to redraw the `genhealth` barchart with green bars.

To see all the named colors, use the command `colors()`. (Do this in the console, or hide the output if you put it into a code chunk – it's a *long* list.) Experiment with some of the named colors to find one you like, and include that in the barchart command in your report.

To make gradations or use colors from other sorts of palettes, you can use the RColorBrewer package. To learn more details of working with color palettes, and with colors in general, see the Color Specifications section of the commands documentation[2]. For instance, try the following:

```
tally(~genhealth, data=brfss) %>% barchart(col=brewer.pal(5, "YlGnBu"))
```

Up to this point, we have been uncritical of the default **chart labeling**. Let's see how to take control of this in R. In particular, for our barchart, we should improve the x-axis label and add an overall title.

A correct label for the x-axis is "Percent of Respondents". To add an x-axis label, use the `xlab=` option, as follows:

```
tally(~genhealth, data=brfss, format="percent", useNA="no") %>%
    barchart(col=brewer.pal(5, "YlGnBu"), xlab="Percent of Respondents")
```

To add a main title, like "Distribution of General Health", use the `main=` option:

```
tally(~genhealth, data=brfss, format="percent", useNA="no") %>%
    barchart(col=brewer.pal(5, "YlGnBu"), xlab="Percent of Respondents",
        main="Distribution of General Health")
```

3. **Pie charts** are popular, but also tend to be over-used. Once in a while it's nice to have one, as long as "proportion of the whole" is the item of interest. To show the `genhealth` responses in a pie chart, try piping the tally into a piechart command, as follows:

   ```
   tally(~genhealth, data=brfss, format="percent", useNA="no") %>%
       pie(col=brewer.pal(5, "YlGnBu"))
   ```

   Compare the piechart and the barchart. Describe an aspect of the distribution which is obvious from the piechart but not from the barchart. Then, do the opposite: find an aspect which is clear in the barchart but not in the piechart.

General Health vs. Exercise

Stacked bar graphs are nice for visualizing the relationship between two categorical variables. Unfortunately, this is one of the trickier plots to make in R. Suppose we're interested in the relationship between `genhealth` and `exerciseany`. We expect to see that those respondents who never exercise have worse general health (right?).

4. Make a tally of percentages for `genhealth` against `exerciseany`, and store the result:

[2] ../commands.html

```
healthVsExercise <- tally(~genhealth|exerciseany, data=brfss,
    format="percent", useNA="no")
healthVsExercise
```

You should see a table with column-wise percentages. This means that the percentages **in each column** add up to 100%. So, for example, we can see that about 21.6% of the respondents *who exercise* report having excellent health.

What percent of the "non-exercisers" report having excellent health? Does it look like the "exercisers" tend to have better general health than the "non-exercisers"? Explain.

5. To show the data as a stacked bar graph, we will basically pipe the table into a barchart command. However, because of the limitations of the barchart command, we have to transpose the table (swap the rows and columns) before piping it into the barchart.

We use the t() command to transpose the table. auto.key is used to make a legend for the barchart. Below are *three different* ways of doing this; they each arrange the legend in a different way. The third includes a chart title. Try each version and note the differences:

```
healthVsExercise %>% t() %>% barchart(auto.key=list(columns=3))
healthVsExercise %>% t() %>% barchart(auto.key=list(space="top"))
healthVsExercise %>% t() %>% barchart(auto.key=list(space="right"),
    main="General Health vs. Does the Respondent Ever Exercise")
```

The default colors are not very pleasing for this barchart. To specify a different color palette, you really have to dig deep. Let's give it a try. Note: You might want to put all the following commands into the same R chunk and wait to run it until it's complete.

The brewer palette which ranges from red to purple will make an attractive spectrum. Let's store it with 5 color values:

```
myColors <- brewer.pal(5, "RdPu")
```

Store the levels (categories) of the genhealth variable:

```
healthLevels <- list(levels(brfss$genhealth))
```

Build the legend myKey using the levels for genhealth and the colors in myColors:

```
myKey <- list(text=healthLevels, rectangles=list(col=myColors))
```

Draw the stacked barchart with the new palette in both the bars and the legend:

```
healthVsExercise %>% t() %>% barchart(col=myColors, key=myKey,
    main="General Health vs. Does the Respondent Ever Exercise")
```

6. Does the relationship between genhealth and exerciseany prove that if a person exercises then they will improve their health? Explain.

General Health vs. Income and Age Group

Now we will see how income level and age group might be related to general health for this group of respondents.

7. **Before looking at any relevant graphs**, make some predictions. Do you think there is any relationship between income level and health? What about the relationship between age and health? Explain.

8. Using what you learned previously in this lab, make a nice stacked bar graph for general health vs. income level (income). The last (large) chunk will definitely be useful, but first you need to store a value for healthVsIncome, similar to how we stored healthVsExercise.

   What do you learn from the graph? Compare with your prediction above.

9. Also make a stacked bar graph for general health vs. age (using age7, which has 10-year intervals).

   What do you learn from the graph? Compare with your prediction above.

Let's investigate the 3-way relationship among general health, income, and age. In particular, is the relationship between general health and income constant across all ages, or does it vary with each age group? This calls for a 3-way graph, plotting general health against income and using age7 as a "panel" variable. The formula syntax for this is ~genhealth|income+age7. Store the tally and take a look at it:

```
healthVsIncomeAndAge <- tally(~genhealth|income+age7, data=brfss,
     format="percent", useNA="no")
healthVsIncomeAndAge
```

This makes a rather overwhelming table of numbers! To help the average human make sense of this, we will make a paneled stacked barchart. There is a "magic" bit of code here, aperm(c(2,3,1)), taking the place of t() for 3-way graphs.

```
healthVsIncomeAndAge %>% aperm(c(2,3,1)) %>%
     barchart(col=myColors, key=myKey)
```

10. For each age group, is it the case that those with higher income level report better general health? Explain how you see this in the graph.

11. Is the nature of the relationship between general health and income exactly the same across all ages, or do you notice it being weaker for some age groups? Explain what you see and how that makes sense.

Submissions

When you are ready to submit:

- Make sure to knit the final report before downloading.
- Download both the Lab5Report.Rmd and the Lab5Report.html files (one at a time).
- **Submit both files via Moodle**.