R Markdown Demo

Jared Sampson

02/14/2017

Let's take a look at how to use R Markdown by going through one of the questions from Problem Set 2, using the `bodytempandheartrate.csv` dataset, which you should put in a new folder with the .Rmd source file you create in RStudio. The question we'll be using is:

> **1a. Calculate 95% confidence intervals for body temperature and heart rate for the sample as a whole and for men and women separately.**

Hopefully I did it right (statistically speaking), but in any case, you'll get the idea behind the R Markdown workflow.

Getting Started

First we'll load the data inside a "fenced code block", or, in `knitr` terms, a "chunk." Note that the characters that make the "fence" are backticks (`` ` ``), i.e. the key to the left of `1` on the top row of the keyboard, not the regular apostrophe/single quotation mark character. Three or more backticks are required, but more are also allowed as long as the opening and closing lines match. The `r` in curly braces indicates that we want this chunk to be evaluated as R code, and the name of this particular chunk (which must be unique if present, but can be arbitrary or even omitted) is `bthr_read`.

````
```{r bthr_read}
# Load the data frame
bthr <- read.table('bodytempandheartrate.csv', header=TRUE, sep=',')
```
````

Then we can calculate our 95% confidence intervals for temperature and heart rate.

````
```{r ci1}
temp_ci <- t.test(bthr$T, conf.int=TRUE, conf.level=0.95)$conf.int
temp_m_ci <- t.test(bthr$T[bthr$MF == 1], conf.int=TRUE, conf.level=0.95)$conf.int
temp_f_ci <- t.test(bthr$T[bthr$MF == 2], conf.int=TRUE, conf.level=0.95)$conf.int
# etc.
```
````

. . . but this is really repetitive and typo-prone. If we're going to be using these data columns a lot, or even if we aren't, we can make it easier to read by assigning some variables in a new block above where we want to calculate the CIs. (This isn't really specifically related to R Markdown, but I find it useful to create variables and custom functions with meaningful and descriptive names to help me know what a piece of data represents or what a section of code is doing.)

````
```{r vars}
# Temperature
temp <- bthr$T
temp_m <- bthr$T[bthr$MF == 1]
temp_f <- bthr$T[bthr$MF == 2]

# Heart Rate
hr <- bthr$HR
hr_m <- bthr$HR[bthr$MF == 1]
```
````

```
hr_f <- bthr$HR[bthr$MF == 2]
````

Now it looks a little cleaner, although still repetitive.

````
```{r ci2}
temp_ci <- t.test(temp, conf.int=TRUE, conf.level=0.95)$conf.int
temp_m_ci <- t.test(temp_m, conf.int=TRUE, conf.level=0.95)$conf.int
temp_f_ci <- t.test(temp_f, conf.int=TRUE, conf.level=0.95)$conf.int
# etc.
```
````

If we will be calculating a lot of confidence intervals and want to simplify the inline code as much as possible, we can define a custom function, `t_conf_int()` that will take 2 arguments, a data column and a confidence level, and return a confidence interval. Note that this doesn't cover the entire range of functionality for the `t.test()` function (multiple data sets, `mu` values, etc.), but is tailored to our specific use case. You may prefer to use the original `t.test()` function, which is totally fine.

````
```{r t_conf_int}
t_conf_int <- function(x, l=0.95) {
    return(t.test(x, conf.int=TRUE, conf.level=l)$conf.int)
}
```
````

Now, our 95% confidence intervals, calculated with our custom `t_conf_int()` function are:

````
```{r ci3}
temp_ci <- t_conf_int(temp)
temp_m_ci <- t_conf_int(temp_m)
temp_f_ci <- t_conf_int(temp_f)
hr_ci <- t_conf_int(hr)
hr_m_ci <- t_conf_int(hr_m)
hr_f_ci <- t_conf_int(hr_f)
```
````

which we can put into the text using inline R code quoted with single backticks such as:`` `r temp_ci` ``. Let's write it as a bulleted list:

- Temp (all): 98.1220029, 98.3764586
- Temp (men): 97.9314722, 98.2777586
- Temp (women): 98.2096189, 98.5780734
- Heart Rate (all): 72.5360699, 74.9870071
- Heart Rate (men): 71.9134314, 74.8250301
- Heart Rate (women): 72.1454691, 76.1622232

Figures

Next, we can try adding some plots for a figure.

````
```{r hist}
hist(temp_m)
hist(temp_f)
hist(temp)
```
````

Whoa, those plots would take up way too much space. Let's collapse them into a single row by adding the following line before the first `hist`.

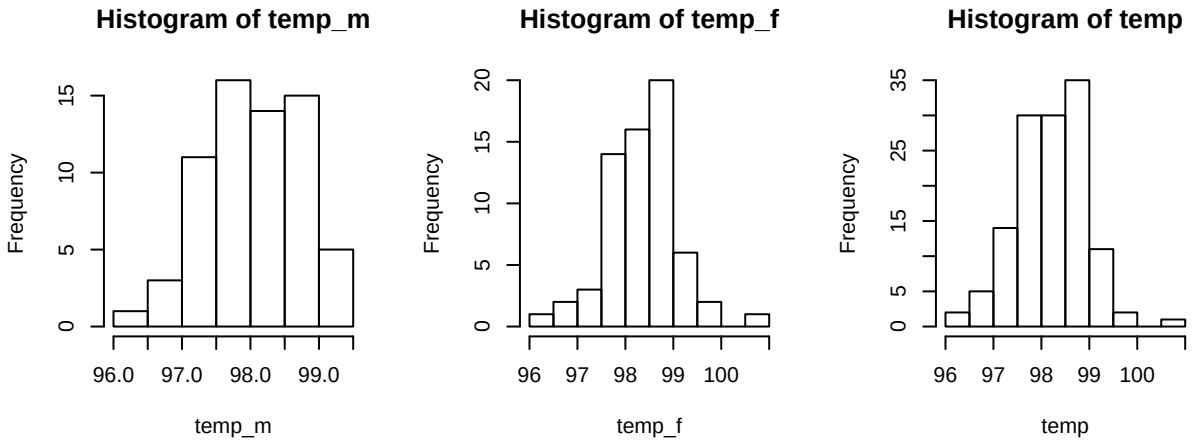

Figure 1: My awesome figure caption.

`par(mfrow=c(1, 3))`

But now, the aspect ratio for each panel is too tall. Let's fix it by adding a parameter after the chunk name.

`{r hist, fig.asp=0.4}`

Good! But what if we wanted to make it into a real floating figure (i.e. doesn't move with the text, but text wraps around it)? Add a caption.

`{r hist, fig.asp=0.4, fig.cap="My awesome figure caption."}`

Putting it all together gives us the figure at the top of this page.

```
par(mfrow=c(1, 3))
hist(temp_m)
hist(temp_f)
hist(temp)
```

If we want to refer to the figure in our text and have it change automatically if we change the order of the figures, one option is to use the package `bookdown` which is written by the same author as `knitr` and used for writing technical books. Install it by running the following in the Rstudio console.

```
install.packages("devtools")
devtools::install_github("rstudio/bookdown")
```

Now change the document format in the YAML frontmatter to `bookdown::pdf_document2` and use the syntax: `\@ref(fig:chunk_label)`. For example, `\@ref(fig:hist1)` shows up as "1". So in the text, we can say something like:

`The distributions of temperatures for men, women, and all study participants are presented in Figure \@ref(fig:hist1).`

For a quick reference guide to the R Markdown, `knitr`, and `pandoc` syntax and options, have a look at:

https://www.rstudio.com/wp-content/uploads/2015/03/rmarkdown-reference.pdf