Lab 0 BIS 505b

Maria Ciarleglio

2/1/2021

Goal of Lab 0

In **Lab 0**, you will become familiar with using **R** Markdown documents.

R Markdown

R Markdown is a unified authoring framework and reporting medium for data analysis that combines code, results, and commentary in one stand-alone document. **R** Markdown provides an alternative to saving **R** code in an **R** script file. When running code from a script file, your numerical output is printed to the *Console* window and graphical output is generally sent to the *Plot* tab. When creating your lab reports, you would traditionally write your code in an **R** script file, execute the code, then copy and paste the relevant output to a Word document, where you would also type up your interpretation and commentary. The benefit of using **R** Markdown is that you can use a single **R** Markdown file to both save your code, execute that code, and generate high quality reports. **R** Markdown documents are fully reproducible and support many different output formats that can easily be shared with others.

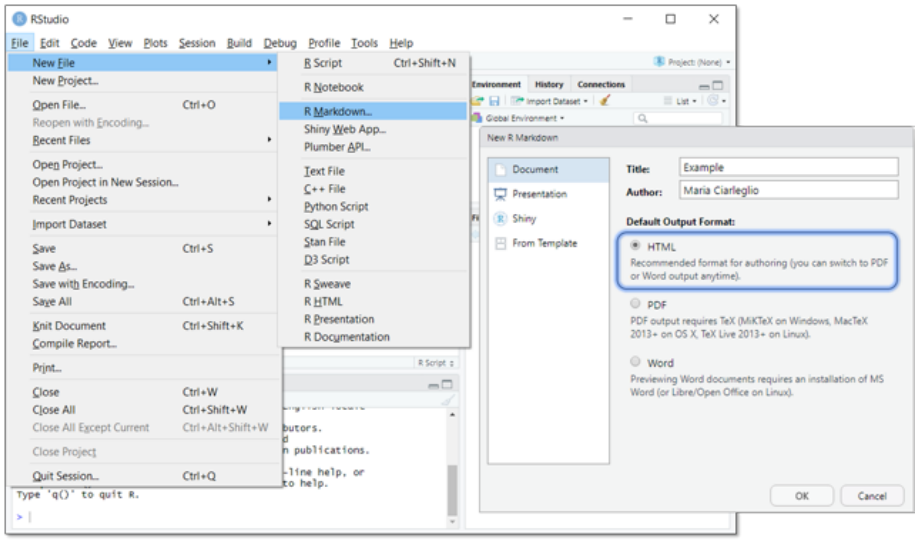
We will use **R** Markdown documents in our **R** labs. You will follow an **R** Markdown document during the lab presentation and you will use an **R** Markdown document to conduct your analyses and describe your findings in your lab reports. This file is an **R** Markdown file (notice that the file name has the extension .Rmd).

Getting Started with R Markdown

To get **R** Markdown working in **RStudio**, you need to first download the `rmarkdown` package. Installing `rmarkdown` will also install the `knitr` package, which we will use. Please also install the `dplyr` and `ggplot2` packages. Download and install packages in your **R** library by running the following code in the **R** or **RStudio** *Console*:

```
install.packages("rmarkdown")
install.packages("dplyr")
install.packages("ggplot2")
```

To create a new **R** Markdown file (.Rmd) in **RStudio**, navigate to `“File” > "New File" > “R Markdown”` . Alternatively, click on the white square with a green + and select “R Markdown” from the drop-down menu. For now, we will generate out output in an .html format. This means that when we execute (or render or “knit”) our .Rmd file, it will create a local .html file that is saved in the same directory that contains your .Rmd file. **RStudio** will open a window that displays the rendered file.

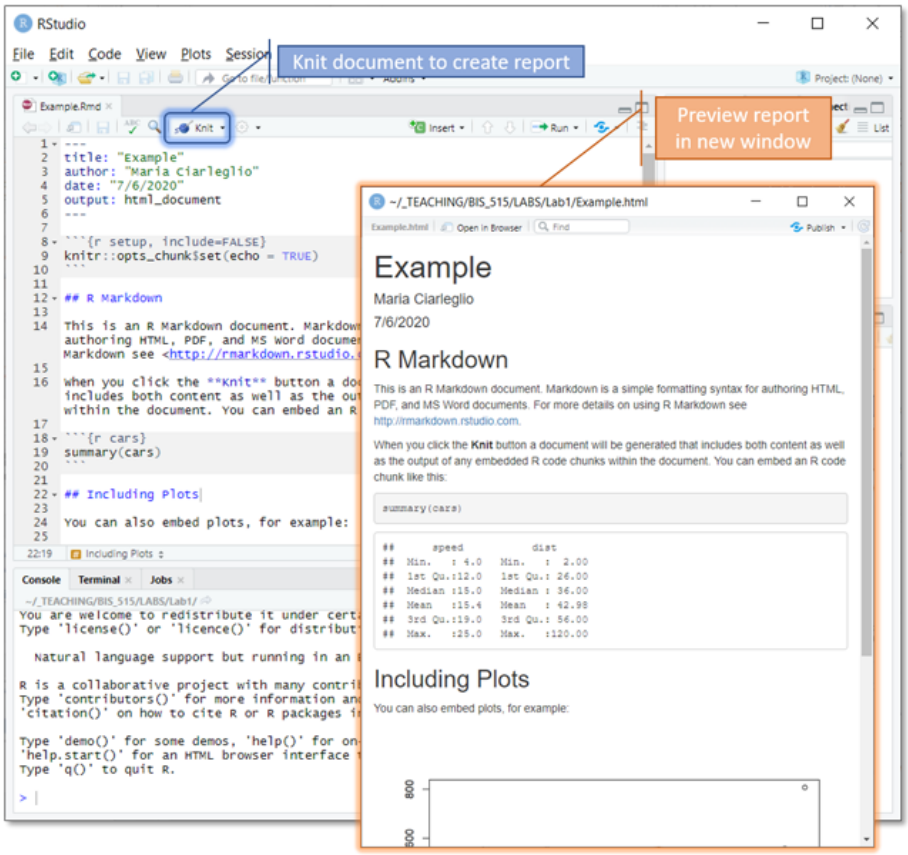


The newly created .Rmd file is a generic template with basic instructions. When you eventually create your own **R** Markdown document, you can delete the sample text below the header to begin from scratch. In this lab, we will work through this .Rmd file.

Note: It is possible to send your results to a PDF or a Word document instead. **RStudio** does not build PDF and Word documents from scratch. You will need to have a distribution of LaTeX installed on your computer to make PDFs and Microsoft Word (or a similar program) installed to make Word files. **Knitting to .html is perfectly fine in this course and seems to work for everyone** (if you have the necessary packages installed). If you would like to try knitting to .pdf, there are instructions for Mac users here (https://medium.com/@sorenlind/create-pdf-reports-using-r-r-markdown-latex-and-knitr-on-macos-high-sierra-e7b5705c9fd) and Windows users here (https://medium.com/@sorenlind/create-pdf-reports-using-r-r-markdown-latex-and-knitr-on-windows-10-952b0c48bfa9).

Rendering (Knitting) the R Markdown Document

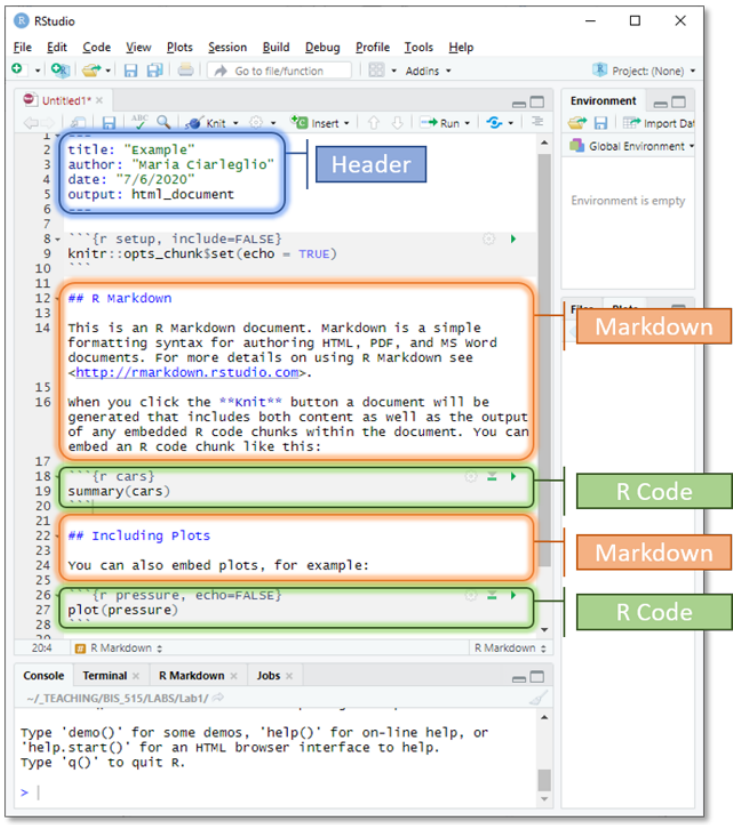
To compile the **R** Markdown document, click the `Knit` button in the toolbar or use the keyboard shortcut Ctrl + Shift + K (Cmd + Shift + K on macOS). This will render the file and display the output of your .Rmd file in a new window. All of the code chunks will be executed and a .html file will be saved in the same folder where you saved your .Rmd file. Note that if you have never saved your .Rmd file, you will be prompted to save the file before it will compile.



Structure of the R Markdown Document

Notice that an **R** Markdown file contains three types of content:

1. A header at the top of this file containing metadata to configure the R Markdown build process.
2. Markdown text with simple formatting.
3. **R** code chunks surrounded by three “back-ticks”, ```` ``` ```` , which will generate numerical or graphical results.



Header

At the top of any **R** Markdown script is the header that contains the metadata of the document. The header uses three dashes (`---`) to separate directives from document content. By default, the header includes the document title, author, date, and output format (HTML, here, requested by specifying `output: html_document` in the header). You can omit the author and date, but should always include a title and output format. A PDF is created by instead specifying `output: pdf_document` , and a Word file is created by specifying `output: word_document` . There are many other options that can be specified in the header, for example,

Table of Contents

You can add a table of contents (TOC) using the `toc` option and specify the depth of headers using the `toc_depth` option. More on headers, below, but if the depth not specified, TOC depth defaults to 3, which will show level 1, 2, and 3 headers.

```
---
title: "Example"
output:
  html_document:
    toc: true
    toc_depth: 2
---
```

Section Numbering

You can add section numbering to headers using the `number_sections` option:

```
---
title: "Example"
output:
  html_document:
    toc: true
    number_sections: true
---
```

HTML Appearance

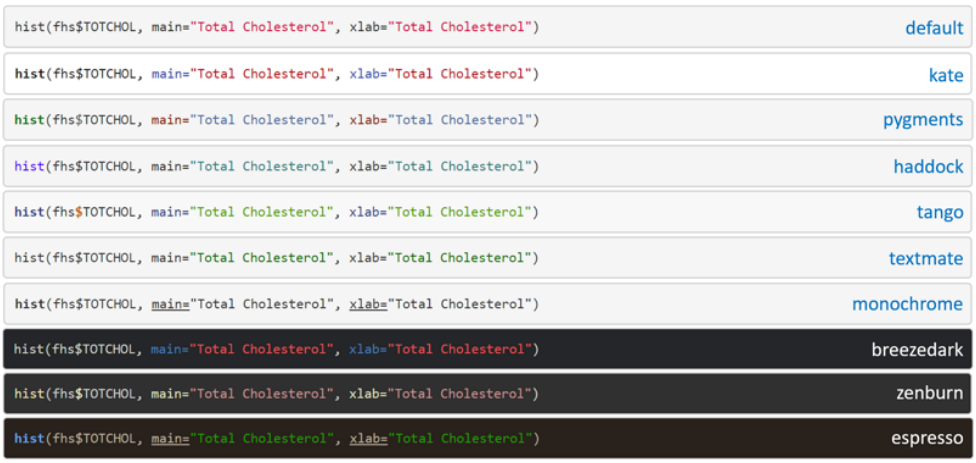
There are several options that control the appearance of HTML documents:

`theme` specifies the theme to use for the page. There are 13 themes in addition to the `default` theme that can be applied to your document without installing any additional packages. Valid themes include default, cerulean, journal, flatly, darkly, readable, spacelab, united, cosmo, lumen, paper, sandstone, simplex, and yeti. This site (https://www.datadreaming.org/post/r-markdown-theme-gallery/) gives examples of most of the themes. If you do not specify a theme, the `default` theme is used. Change the theme on an indented line under `html_document` in the header:

```
---
title: "Example"
output:
  html_document:
    theme: united
---
```

`highlight` specifies the **R** syntax highlighting style. Supported styles include default, tango, pygments, kate, monochrome, espresso, zenburn, haddock, breezedark, and textmate.

```
---
title: "Example"
output:
  html_document:
    theme: united
    highlight: tango
---
```



Note: Additional templates are available in the `prettydoc` and `rmdformats` packages.

Markdown Text

Markdown syntax can be used to format the text in an **R** Markdown document. Some common formatting commands are shown below.

Bulleted List

Unordered list items start with *, -, or +. You can nest lists by indenting the sub-list, e.g.,

- Item 1
- Item 2
    - Sub-item 2a
    - Sub-item 2b

```
* Item 1
* Item 2
    * Sub-item 2a
    * Sub-item 2b
```

Exercise: Create an **R** Markdown bulleted to do list for yourself below.

▶ Answer:

Numbered List

You can also nest bulleted lists within numbered lists, e.g.,

1. Item 1
   i). Sub-item 1
2. Item 2
    - Sub-item 2a
    - Sub-item 2b

```
1. Item 1
    i). Sub-item 1
2. Item 2
    + Sub-item 2a
    + Sub-item 2b
```

Headers

Section headers are written after a number of pound signs, e.g.,

```
# Level 1 Header

## Level 2 Header

### Level 3 Header
```

Formatting

Italic text or *Italic text*
Bold text or **Bold text**
superscript^2
subscript$_1$
~~strikethrough~~
`verbatim code`

> block quote (requires blank line above)

```
*Italic text* or _Italic text_
**Bold text** or __Bold text__
superscript^2^
subscript~1~
~~strikethrough~~
`verbatim code`

>block quote (requires blank line above)
```

Link

Hyperlinks (https://www.rstudio.com) are created using the syntax `[link](https://www.rstudio.com)`

Manual Line Breaks

Option 1: Insert a blank line after your line break

Option 2: End a line with two or more spaces, e.g.,

I want to insert a line break here
and here
.

Option 3: Use the `<br>` command

Line
break

```
Line<br>
break
```

Horizontal Line

Three or more asterisks or dashes will produce a horizontal line:

```
****

----
```

Comments

```
<!--This text comment will not display in the rendered html-->
```

Manually Creating a Table (“Pipe Table”)

You can create a simple formatted table using pipes (i.e., vertical bars, `|`) to separate the table cells. The table begins with a header row, and the columns are each separated by a pipe `|`. The header row is separated from the remaining rows by hyphens `---`. The row containing the hyphens can be used to format the alignment of the column using the colon symbol, `:`. The default cell alignment is left alignment, which may also be specified using `:---`. Right-alignment is requested using `---:` and center-alignment is requested using `:---:`. A hard return, followed by a `:` and text creates a table description.

```
| Right | Left | Default | Center |
|------:|:-----|---------|:------:|
| 12 | 12 | 12 | 12 |
| 123 | 123 | 123 | 123 |
| 1 | 1 | 1 | 1 |

: Table description
```

Table description

Right	Left	Default	Center
12	12	12	12
123	123	123	123
1	1	1	1

R Code Chunks

Embed **R** code into the report using code chunks. The code chunks are where we will conduct our data analysis. **R** will run the code and include the results under the printed code chunk when it renders the report. Code chunks should be enclosed by three ticks, ```` ``` ````.

The code chunk begins with ```` ```{r} ````, e.g., (look at the .Rmd file to see the syntax of the code chunk below)

```
# Fahrenheit to Celsius conversion
tempf <- 73
tempc <- (tempf-32)*5/9
tempc
```

```
## [1] 22.77778
```

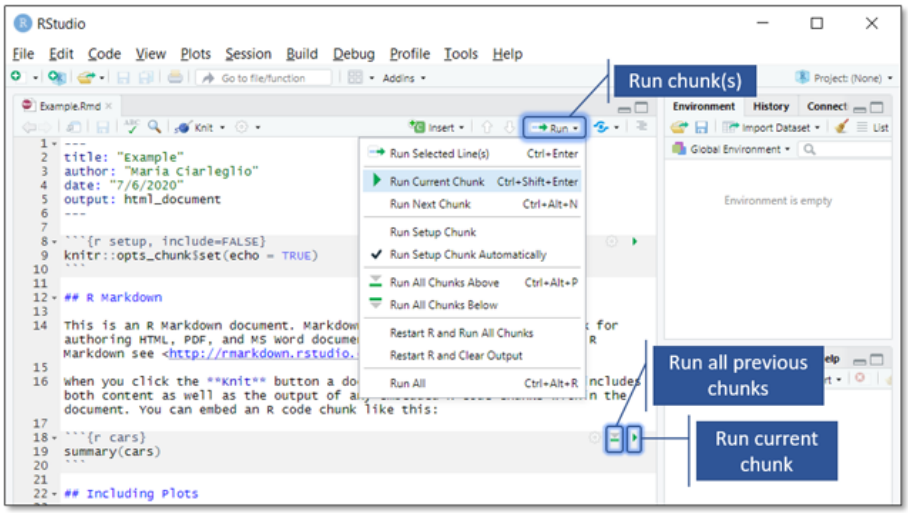
Note that objects created in earlier code chunks can be directly referenced in subsequent code chunks (i.e., no need to re-define a variable over and over again in each code chunk). Below, I'm referencing the variable `tempc` that I created in the code chunk above.

```
round(tempc, 2)
```

```
## [1] 22.78
```

Run Individual Code Chunk in RStudio

Run an individual chunk of code by placing your cursor inside the code chunk and clicking on the `Run` button in the toolbar. Select `"Run" > "Run Current Chunk"`. Alternatively, select the green run button ▶ in the upper right-hand corner of the individual code chunk to run the current chunk. By default, any printed results will appear immediately below the code chunk. This is equivalent to running those lines of code in the **R** *Console*, except that the results will appear below your code chunk instead of in the *Console* window.

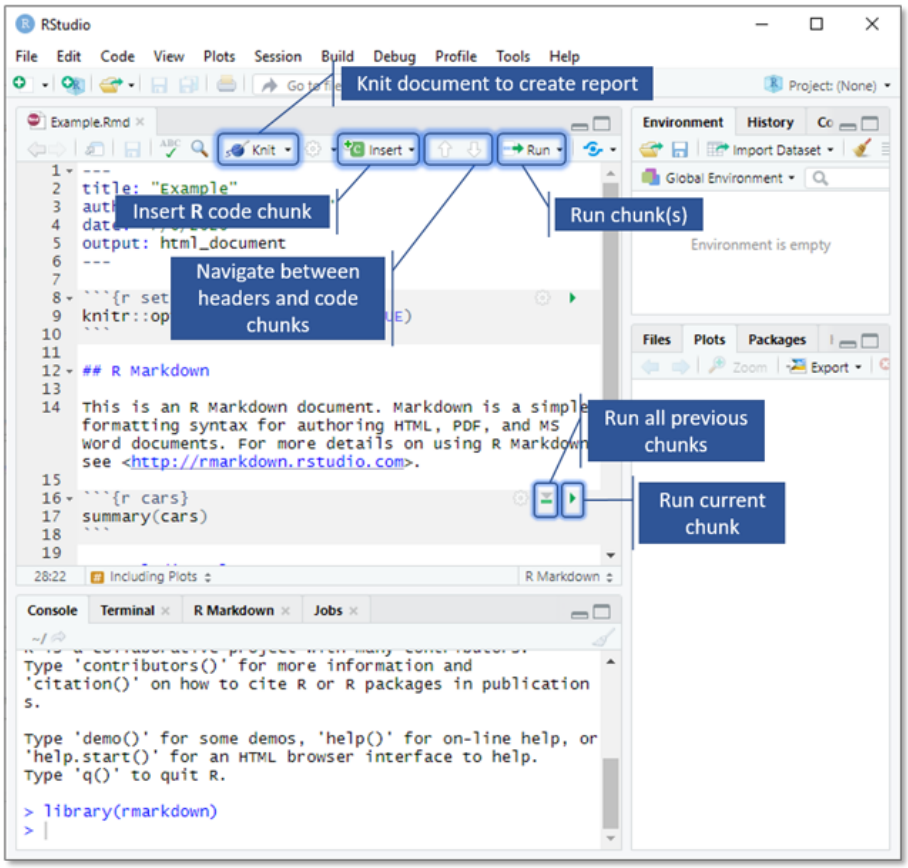


Exercise: If you haven't already done so, run the *Fahrenheit to Celsius conversion* code chunks above using the green run button in those chunks.

Notice that when we run individual code chunks, variables or objects we create will appear in the *Environment* tab of **RStudio** (usually the upper right-hand panel of **RStudio**). However, when we knit a document, **R** will run all of the code chunks in your file, but will not store the objects that are created in the .Rmd file in your **R** session's Global Environment.

New Code Chunk

To insert an **R** code chunk, click on the `Insert` button in the toolbar or use the keyboard shortcut Ctrl + Alt + I (Cmd + Option + I on macOS).



Importing Data Sets and Packages

To analyze an external data set, you must import or read in the data. I generally set my WD and read in any data sets in an **R** code chunk at the *beginning* of my .Rmd document (see **code chunk 3** at the beginning of this .Rmd document). Below, is the code used to read in the Framingham Heart Study data from the participants' first examination period contained in the data file `fhs_exam1.csv`. This code assumes the CSV file is saved in the `C:\BIS_505\LABS\Lab0` folder on your computer, but you would change the path to point to your `Lab0` folder that contains `fhs_exam1.csv`. Note that the code below is commented so that it does not run and attempt to access a potentially non-existent folder on your machine.

```
# Uncomment and map to your Lab0 folder; csv file should be saved in your WD
# setwd("C:\\BIS_505\\LABS\\Lab0")
# fhs <- read.csv("fhs_exam1.csv")
```

Exercise: If you haven't already done so, go up to **code chunk 3** at the beginning of this .Rmd document and run that code chunk using the green run button in the upper right of the code chunk. After running that code chunk, you should see the `fhs` data frame object in the *Environment* tab of **RStudio**.

▶ Answer:

After the data set is imported, we can perform analyses using the data in subsequent code chunks. Again, if you are only viewing your results in the rendered web page after knitting your document, then you do not have to individually run **code chunk 3**. But if you want to run individual code chunks without knitting (i.e., send individual lines of code to the *Console*), then the `fhs` data frame must exist in your Global Environment before you can run the individual code chunks below.

```
names(fhs)
```

```
##  [1] "RANDID"   "SEX"      "TOTCHOL"  "AGE"      "SYSBP"    "DIABP"
##  [7] "CURSMOKE" "CIGPDAY"  "BMI"      "DIABETES" "BPMEDS"   "HEARTRTE"
## [13] "GLUCOSE"  "PREVCHD"  "PREVAP"   "PREVMI"   "PREVSTRK" "PREVHYP"
## [19] "TIME"     "PERIOD"   "DEATH"    "ANGINA"   "HOSPMI"   "MI_FCHD"
## [25] "ANYCHD"   "STROKE"   "CVD"      "HYPERTEN" "TIMEAP"   "TIMEMI"
## [31] "TIMEMIFC" "TIMECHD"  "TIMESTRK" "TIMECVD"  "TIMEDTH"  "TIMEHYP"
```

```
summary(fhs$TOTCHOL)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##     107     206     234     237     264     696      52
```

Similarly, if you are using any packages in your analysis, the packages must be loaded in the .Rmd file using `library()` as in a normal **R** script. See **code chunk 2** at the beginning of this .Rmd document where the `dplyr`, `knitr`, and `ggplot2` [1] packages are loaded.

Exercise: Report the dimensions of the `fhs` data frame (number of rows, number of columns).

▶ Answer:

Customizing R Output

Chunk output can be customized with arguments set in the `{r ...}` of a chunk header, with arguments separated by a comma. For example, `{r, echo=FALSE}` will prevent the **R** code in that particular code chunk from appearing in the final rendered document. Note, however, that the output will be displayed (useful when you do not want the reader to see your code, but want to display the results in your final report). `{r, include=FALSE}` will have **R** evaluate the chunk, but neither the **R** code nor the output will be displayed in the final document. This option is useful when loading libraries or importing data. Some commonly-used code chunk options are given in the table below:

Option (default)	Description
`echo=TRUE`	Code chunk displayed in output document? (output is displayed)
`eval=TRUE`	Code chunk evaluated (run)?
`include=TRUE`	Code and results included in final report?
`collapse=FALSE`	Collapse code and output into a single block?
`message=TRUE`	Display messages generated by the code in final report?
`warning=TRUE`	Display warnings in final report?
`error=FALSE`	Display errors in final report? (default =FALSE will cause rendering to stop if error encountered)
`cache=FALSE`	Cache results for future renders?
`comment="##"`	Comment character that prefaces printed results
`results="asis"`	Useful for displaying tables in future packages that we will use

---

Exercise: Use the `comment=` option to preface the **R** output that gives the dimensions of `fhs` with “>” instead of “##”. Also hide the code, but display the output in your report.

▶ Answer:

Global Chunk Options

If you use the same set of chunk options throughout a document, you can set global chunk options at the top of the .Rmd file (see **code chunk 1** at the beginning of this .Rmd document). For example, if we did not want to print any **R** code used in the analysis, we would set `echo=FALSE` in an initial code chunk by specifying `knitr::opts_chunk$set(echo = FALSE)`.

Inline Code

Warning: I **love** inline coding and use it whenever I can. In my opinion, this is one of the best features of **R** Markdown for creating reports that need to be updated frequently. If you've ever received a "Whoops, I'm re-sending you the data file and need you to re-run THE. ENTIRE. ANALYSIS" e-mail from an investigator, you will learn to love this feature, as well.

To embed **R** code in a line of text, surround the code with a pair of backticks and the letter `r`, e.g., `` `r round(mean(fhs$AGE), 2)` ``. Using this in a sentence: the average age of individuals enrolled in the FHS at their first examination is 49.93. Notice that inline code does not display the **R** commands, only their output.

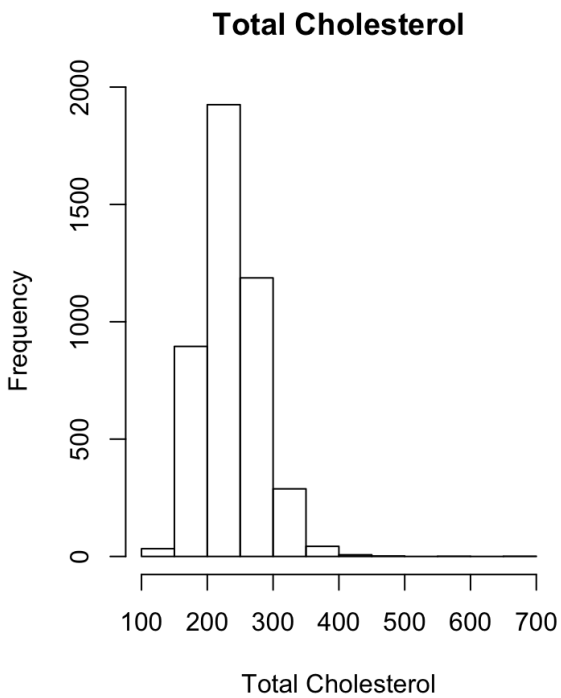
Exercise: Use my beloved inline coding to report the number of rows in the `fhs` data frame in a complete sentence. Remember the `nrow()` function.

▶ Answer:

Plots

You can also embed plots in your report, e.g.,

```
hist(fhs$TOTCHOL, main="Total Cholesterol", xlab="Total Cholesterol")
```



Total Cholesterol at Exam 1 in Framingham Heart Study

R chunk options in the chunk header can set the figure dimensions, alignment, and specify the caption. `{r fig.align="center", fig.width=4", fig.cap="Total Cholesterol at Exam 1 in Framingham Heart Study"}` was used in the figure above. Some commonly-used code chunk options for figures are given in the table below:

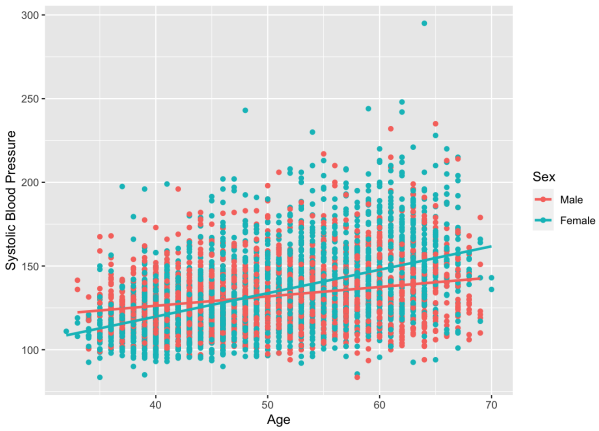
Option	Description
`fig.align`	'left', 'right' or 'center'

Option	Description
`fig.cap`	Figure caption
`fig.height`	Height in inches, `fig.height=5`
`fig.width`	Width in inches, `fig.width=7`
`out.width`	Graphics dimensions are scaled `out.width=50%`

You can display two plots one beside each other. Add `out.width=c('50%', '50%'), fig.show='hold'` to your chunk header. For example,

```
ggplot(fhs, aes(x=AGE, y=SYSBP, color=factor(SEX, levels=c(1,2), labels=c("Male","Female"))))
 +
  geom_point() +
  geom_smooth(method = "lm", se = FALSE) +
  labs(color = "Sex", x = "Age", y = "Systolic Blood Pressure")


ggplot(fhs, aes(x=factor(CURSMOKE, levels=c(0,1), labels=c("No","Yes")), y=SYSBP, fill=factor
(SEX, levels=c(1,2), labels=c("Male","Female")))) +
  geom_boxplot() +
  labs(fill = "Sex", x = "Current Smoker", y = "Systolic Blood Pressure")
```



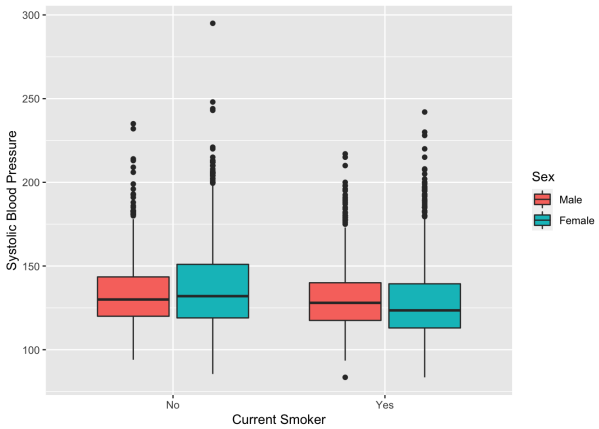


Table Formats

Numerical output appears as it would in the *Console* window (i.e., as plain text):

```
head(fhs)[,1:10]  # head of first 10 columns of fhs
```

```
##   RANDID SEX TOTCHOL AGE SYSBP DIABP CURSMOKE CIGPDAY   BMI DIABETES
## 1   2448   1     195  39 106.0    70        0       0 26.97        0
## 2   6238   2     250  46 121.0    81        0       0 28.73        0
## 3   9428   1     245  48 127.5    80        1      20 25.34        0
## 4  10552   2     225  61 150.0    95        1      30 28.58        0
## 5  11252   2     285  46 130.0    84        1      23 23.10        0
## 6  11263   2     228  43 180.0   110        0       0 30.30        0
```

The `kable()` function in the `knitr` package formats printed tables. The primary input in the `kable()` function is a data frame. Note that you must load the `knitr` library (`library(knitr)`) before you are able to use the `knitr()` function that is in this package. See **code chunk 2** at the beginning of this .Rmd document where the `knitr` package is loaded.

```
kable(head(fhs)[,1:10])  # head of first 10 columns of fhs
```

RANDID	SEX	TOTCHOL	AGE	SYSBP	DIABP	CURSMOKE	CIGPDAY	BMI	DIABETES

RANDID	SEX	TOTCHOL	AGE	SYSBP	DIABP	CURSMOKE	CIGPDAY	BMI	DIABETES
2448	1	195	39	106.0	70	0	0	26.97	0
6238	2	250	46	121.0	81	0	0	28.73	0
9428	1	245	48	127.5	80	1	20	25.34	0
10552	2	225	61	150.0	95	1	30	28.58	0
11252	2	285	46	130.0	84	1	23	23.10	0
11263	2	228	43	180.0	110	0	0	30.30	0

The `kable()` function also allows you to rename the columns, round the numeric values, and set a table caption. Take a moment to understand what the code chunk below is doing.

```
tab_01 = data.frame(
  Measure = c("Age", "Total Cholesterol", "Heart Rate"),
  N       = c(sum(!is.na(fhs$AGE)), sum(!is.na(fhs$TOTCHOL)), sum(!is.na(fhs$HEARTRTE))),
  MEAN    = c(mean(fhs$AGE, na.rm=TRUE), mean(fhs$TOTCHOL, na.rm=TRUE), mean(fhs$HEARTRTE, na.r
m=TRUE)),
  SD      = c(sd(fhs$AGE, na.rm=TRUE), sd(fhs$TOTCHOL, na.rm=TRUE), sd(fhs$HEARTRTE, na.rm=TRUE
)),
  MEDIAN = c(median(fhs$AGE, na.rm=TRUE), median(fhs$TOTCHOL, na.rm=TRUE), median(fhs$HEARTRT
E, na.rm=TRUE))
)
tab_01
```

```
##             Measure    N      MEAN        SD MEDIAN
## 1               Age 4434  49.92580  8.676929     49
## 2 Total Cholesterol 4382 236.98425 44.651098    234
## 3        Heart Rate 4433  75.89104 12.113635     75
```

```
kable(
  tab_01,
  col.names = c("**Measure**", "**N**", "**Mean**", "**SD**", "**Median**"),
  digits = 2,
  caption = "Summary Statistics"
)
```

Summary Statistics

Measure	N	Mean	SD	Median
Age	4434	49.93	8.68	49
Total Cholesterol	4382	236.98	44.65	234
Heart Rate	4433	75.89	12.11	75

Comments on Your First Lab Assignment

Lab Assignment 0 has been uploaded to Canvas (`Lab0Assignment_2021.Rmd`), due on 02/14/2021. The purpose of this lab assignment is for you to gain experience with **R** Markdown. You will also review some commonly-used **R** functions that you should already be familiar with from EPH 505a. [If you're feeling rusty, please see **R-Review_2021.pdf** for an introduction to **R**.] In the lab assignment, you will also produce some descriptive statistics that you most likely covered in your previous course, but these functions will be reviewed in **Lab Session 1** on 02/08/2021.

Save the `Rmd` file (`Lab0Assignment_2021.Rmd`) to your `Lab0` folder and perform your work in that file. You will be analyzing data (`hgb.csv`) from a study conducted to investigate the impact of herbicide exposure on maternal health. Before you can *knit* your `Lab0Assignment_2021.Rmd` **R** Markdown file for the first time, you must set your working directory and read in the `hgb.csv` data file (see lines 26-27 of `Lab0Assignment_2021.Rmd`). We recommend that you save `hgb.csv` to your `Lab0` folder and set the `Lab0` folder as your WD.

- Additional tutorials on **R** Markdown:
  - Link: http://rmarkdown.rstudio.com (http://rmarkdown.rstudio.com)
  - Link: https://bookdown.org/yihui/rmarkdown/html-document.html (https://bookdown.org/yihui/rmarkdown/html-document.html)
- **R** Markdown cheetsheet (https://github.com/rstudio/cheatsheets/raw/master/rmarkdown-2.0.pdf): In **RStudio**, go to `"File" > "Help" > "Cheatsheets" > "R Markdown Cheat Sheet"`
- **R** Markdown help in **RStudio** help tab: In **RStudio**, go to `"File" > "Help" > "Markdown Quick Reference"`

---

1. https://ggplot2.tidyverse.org/reference/ (https://ggplot2.tidyverse.org/reference/)↩