Data Analysis Week 4

Example Report

Robert Edwards

1 Introduction

Section 1: This will link to he first section

1.1 Subsection

2 Code Chunks {#sec:code chunks}

Code "chunks" allow for R code to be embedded within a document. Not only can the code be easily included within a document, the code can also be evaluated. Hence, you can produce an entire report based on an analysis that is contained within a single file instead of having separate files containing your R code, plot images and comments. Using R code within an R Markdown document is done within the following environment:

- **echo**: include the R code within the code chunk in the document (TRUE/FALSE, default=TRUE);
- **eval**: evaluate the R code within the code chunk (TRUE/FALSE, default=TRUE);
- **warning**: suppress warnings from R (TRUE/FALSE, default=TRUE); and 3
- **message**: suppress messages from R (TRUE/FALSE, default=TRUE).

Often the first use of a code chunk in a R Markdown file is to load packages so that functions and datasets are available for analysis. For example, the following code chunk will load these packages:

- ggplot2 - to access the tidyverse visualization functions
- dplyr - to access the tidyverse data manipulation functions
- skimr - to access the tidyverse data summary functions
- moderndive - to access the student feedback data evals that we looked at in Week 3
- datasets - to access the iris data frame

Then, let's say we wanted to select the score and bty_avg variables from the evals data set to be used later, we can do that using the following code chunk:

This will evaluate the R code and store the subsetted data set as the object evals.scores so that it can be used later. If you want to embed the code within the Markdown document then you would simply set echo = TRUE and it will include the R code as follows:

```
evals.scores <- evals %>%
dplyr::select(score, bty_avg)
```

2.1 Inline Code{sec:sub}

This section had errors

3 Tables of Data{sec:tables of data}

Let's say we wanted to create a table of the first 5 rows of the iris data from the datasets library. We can create the table using the kable function as follows:

```
kable(head(iris, n = 5), caption = '\\label{tab:iris} The first 5 rows of the iris data.')
```

Table 1: The first 5 rows of the iris data.

Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
5.1	3.5	1.4	0.2	setosa
4.9	3.0	1.4	0.2	setosa
4.7	3.2	1.3	0.2	setosa
4.6	3.1	1.5	0.2	setosa
5.0	3.6	1.4	0.2	setosa

Notice that within the caption argument of the kable function there is \label{tab:iris}. This is how you label tables in order to refer to the within the text. For example, Table 1 displays the first 5 rows of the iris data... will produce Table 1 displays the first 5 rows of the iris data... where, like for sections, the "1" will be a hyperlink directed to the table.

See also the Example Report Markdown file for an example of using the kable_styling function for changing the size and positioning of a table. For example, its often useful to include kable_styling(font_size = 10, latex_options = 'HOLD_position') as the final term in the "pipe" to control the size of the font used to construct the table and to ensure the table appears at that location in the document (otherwise it may be placed at the bottom or top of the page).

3.1 Table Summaries{sec:sub}

Often we need to report summary statistics for different subsets in a set of data. For instance we can construct a table of summaries for the sepal length of the different species in the iris data using:

```
library(kableExtra)
iris %>%
group_by(Species) %>%
summarise(n=n(),Mean=round(mean(Petal.Width),digits=1), St.Dev=round(sd(Petal.Width),digits=1),
Min=min(Petal.Width), Q1 = quantile(Petal.Width,0.25), Median=median(Petal.Width),
Q3 = quantile(Petal.Width,0.75), Max=max(Petal.Width)) %>%
kable(caption = '\\label{tab:summaries} Summary statistics on the sepal length by species of irises.') %
kable_styling(font_size = 10, latex_options = "hold_position")
```

```
## Warning in kable_styling(., font_size = 10, latex_options =
## "hold_position"): Please specify format in kable. kableExtra can customize
## either HTML or LaTeX outputs. See https://haozhu233.github.io/kableExtra/
## for details.
```

Table 2: Summary statistics on the sepal length by species of irises.

Species	n	Mean	St.Dev	Min	Q1	Median	Q3	Max
setosa	50	0.2	0.1	0.1	0.2	0.2	0.3	0.6
versicolor	50	1.3	0.2	1.0	1.2	1.3	1.5	1.8
virginica	50	2.0	0.3	1.4	1.8	2.0	2.3	2.5

3.2 Tables of Model Estimates{sec:sub}

Often we also need to report the results of fitting a model to our data. For instance if we modeled the sepal length on the different species in the iris data by:

```
model <- lm(Sepal.Length ~ Species, data = iris)
get_regression_table(model) %>%
dplyr::select(term,estimate) %>%
#Note that it seems necessary to include "dplyr::" here!!
kable(caption = '\\label{tab:reg} Estimates of the parameters from the fitted linear regression model.')
kable_styling(latex_options = 'HOLD_position')
```

```
## Warning in kable_styling(., latex_options = "HOLD_position"): Please
## specify format in kable. kableExtra can customize either HTML or LaTeX
## outputs. See https://haozhu233.github.io/kableExtra/ for details.
```

Table 3: Estimates of the parameters from the fitted linear regression model.

term	estimate
intercept	5.006
Speciesversicolor	0.930
Speciesvirginica	1.582

3.3 Tables By Hand{sec:sub}

Tables can also be produced "by hand"" in Markdown. For example, the table above corresponding to the first 5 rows of the iris data can be produced by hand by typing the following text (without any other text) into a .Rmd file:

Sepal Length	Sepal Width
1	2
3	4
5	6
7	8
9	10

The vertical separators | are used between columns, while — is placed below table/column headings. Alignment of the columns is done using colons, that is, for left alignment put :—, for right alignment put —:, and for centred alignment put :—:. A limitation of creating tables by hand is that you cannot give them a label to refer to them later within the text (which will also need to be done by hand).

3.4 Figures{sec:figures}

Including plots within an R Markdown document is straightforward. The R code for the plot is simply included within a code chunk including additional arguments for plot size and positioning. For example, this code chunk in the .Rmd file will produce the scatterplot of teaching and beauty scores below:

```
ggplot(evals.scores, aes(x = bty_avg, y = score)) +
geom_point() +
labs(x = "Beauty Score", y = "Teaching Score") +
```

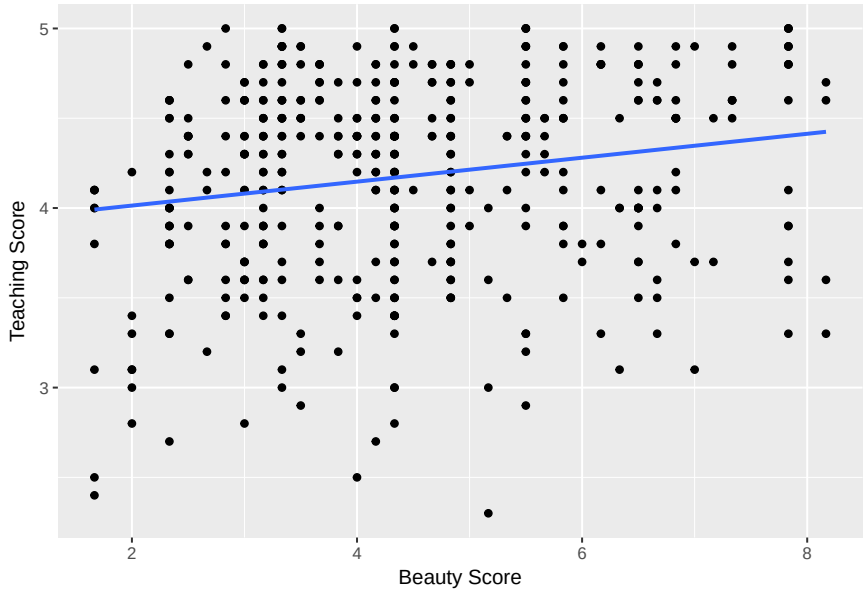

Figure 1: Relationship between teaching and beauty scores. The best-fitting line has been superimposed.

```
geom_smooth(method = "lm", se = FALSE)
```