MY457: Problem Set Template

[Your Name Here]

Wed/22/Jan

1 R Markdown

You must submit your problem set solutions as a .pdf. We strongly recommend that you use `R Markdown`, which allows you to integrate R code and your own writing/commentary into a single reproducible document. This specific template is set up as a `Bookdown` document written to .pdf. You should click the **Knit** button in `RStudio` to produce the .pdf document. The majority of your code should appear **only** in the code appendix at the end of the document. The main part of your document should be written text, plus tables, plots, and occasionally code chunks where relevant (e.g. if you write a particular function you want to discuss in detail).

2 Code

`R Markdown` documents allow you to write and evaluate (run) code in `chunks`, which can support a variety of languages. For this class you will only need to use `R`. It is always advisable to given your chunks meaningful names. You can set specific options in the code chunk, or you can set global options that can be overridden by specific chunks. Your code should be well-formatted and clearly written, and we advise heavy commenting of code using the normal `R` style. In this document we leave an open line at the end of code chunks to make the code appendix easier to follow.

In general, if you include a code chunk that does something you do not need to set additional options. The global options used in this template mean that code won't be print in your main document, but will be printed in your code appendix. This is the preferred approach to most code you write for your problem sets.

```
## [1] "hello world"
```

If you do want to include and run some code for some reason, you should use `echo=TRUE` in the chunk options. Please use this sparingly as it can become overwhelming very quickly if you include lots of code in your write-up. This is why we have the code appendix.

```
# everything in this chunk will be printed in the main document,
# plus the output of the code.
print("hello world")
```

```
## [1] "hello world"
```

By contrast, if you just want to include code but not run it (e.g. to illustrate something without producing and including any results), you should use `echo=TRUE` and `eval=FALSE`:

```
# everything in this chunk will be printed in the main document,
# but the code won't run and so the output won't be included.
print("hello world")
```

3 In-line references

It can often be useful to use in-line references to the value of particular objects that are defined in your `R` environment. For example, you might generate an estimate in `R` and want to discuss or interpret that estimate in your write-up. Referencing the value of an object from your environment will look like this in your markdown text: `` `r object_name` ``. Below we provide a practical example.

```
# here we define alpha, so that we have an object to reference in-line
alpha <- 12345
```

The value of alpha is equal to 12345. You can also do operations in-line. The value of alpha divided by 10 is 1234.5.

4 Plots

To produce and include plots, create a chunk of code which generates the plot. Because the global options are set as `echo=FALSE` the code itself will not be shown, only the plot. Please give your figures labels, and discuss them in your text. You can do this by referencing the figure using `\@ref(fig:chunk-name)` in-line. For example, Figure 1 shows the cars data.

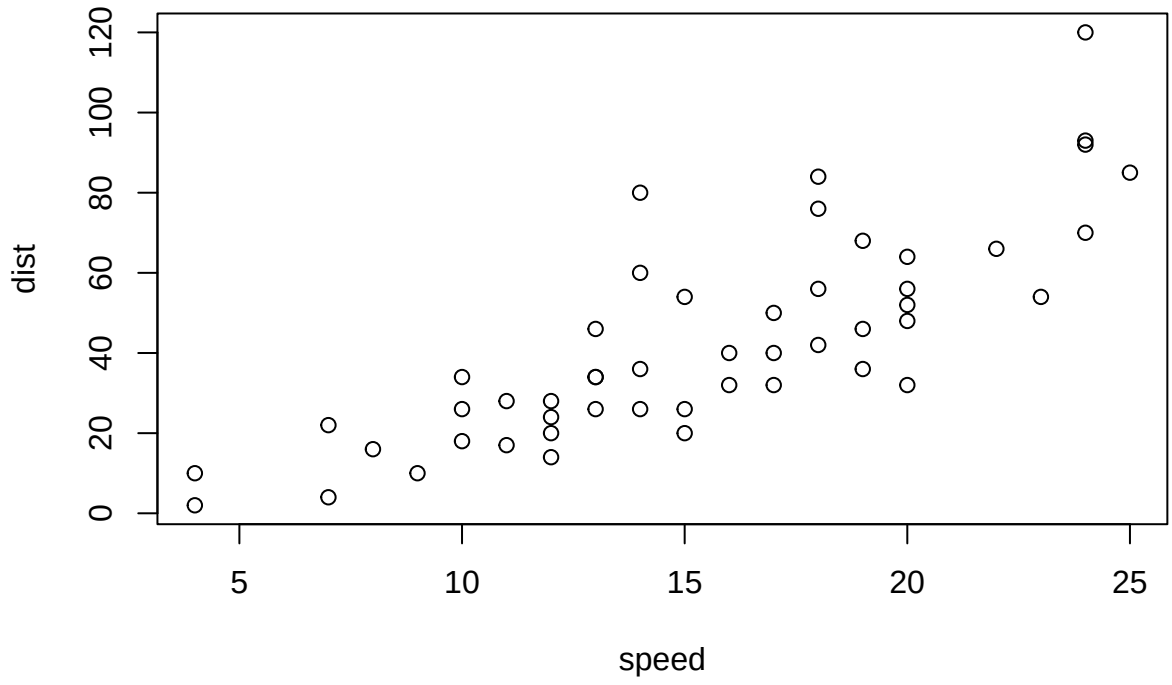

Figure 1: A simple scatterplot of the cars data

5 Tables

To produce and include tables, create a chunk of code which generates the table. You can generate the table using numerous `R` packages including but not limited to `xtable`, `mtable`, `stargazer`, `kable` and `kable`

Table 1: The first five rows of mtcars

	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb
Mazda RX4	21.0	6	160	110	3.90	2.620	16.46	0	1	4	4
Mazda RX4 Wag	21.0	6	160	110	3.90	2.875	17.02	0	1	4	4
Datsun 710	22.8	4	108	93	3.85	2.320	18.61	1	1	4	1
Hornet 4 Drive	21.4	6	258	110	3.08	3.215	19.44	1	0	3	1
Hornet Sportabout	18.7	8	360	175	3.15	3.440	17.02	0	0	3	2

`extra`. Below we shown an example using `kable` from the `knitr` package. Because the global options are set as `echo=FALSE` the code itself will not be shown, only the plot. As with figures, pelase give your tables labels, and discuss them in your text. You can do this by referencing the table using `\@ref(tab:chunk-name)` in-line. For example, Table 1 shows the cars data.

6 Code appendix

```
# this chunk contains code that sets global options for the entire .Rmd.
# we use include=FALSE to suppress it from the top of the document, but it will still appear in the app
knitr::opts_chunk$set(echo=FALSE, warning=FALSE, message=FALSE, linewidth=60)

# you can include your libraries here:
library(tidyverse)

# and any other options in R:
options(scipen=999)

# nothing in this chunk will be printed in the main document
# except the output of the code.
print("hello world")

# everything in this chunk will be printed in the main document,
# plus the output of the code.
print("hello world")

# everything in this chunk will be printed in the main document,
# but the code won't run and so the output won't be included.
print("hello world")

# here we define alpha, so that we have an object to reference in-line
alpha <- 12345

# making a simple scatterplot using the cars data
plot(cars)

# making a simple table of the mtcars data
knitr::kable(mtcars[1:5,], caption = "The first five rows of mtcars")

# this chunk generates the complete code appendix.
# eval=FALSE tells R not to re-run (``evaluate'') the code here.
```