Introduction

This file is a R Markdown template that you may use to help you generate the file of your take home problems. You are not required to write in R Markdown, but it is an easy way to show your R results.

In the case of this document, a pdf file is generated if you click in the **Knit** buttom above the script. You can check that the standard output for this file is a pdf on line 5, which stated **output: pdf_document**.

If the output is not defined as pdf, or for any other reason you want to generate a HTML/Word file, simply click on the arrow on the right side of the **Knit** buttom and change your output.

Sections

You can include sections and subsections easily on R Markdown by simply using the # symbol before the title of your current session. This also creates shortcuts sections in the pdf file. Let us try to add sections.

This is a subsection

And you can write whatever you need here.

This is a subsubsection

And as you noticed, this feature is easy to use.

Bullet Points, Bold and Italic

To create a Bullet point simply add the * symbol at the beginning of the line. You can also use the + symbol to include a dash

- This is a bullet point.
  - This is a dash.

To make something *italic*, simply write your text between two * symbols. To make it **bold**, use two * symbols on each side of the text.

Math Mode

One interesting thing about R Markdown is that you can use LaTeX code in the document, which means that the math mode is also available. If you want to write anything in math mode along the text, you simply need to include your text between two **$** symbols.

For example, I can write $\exp(x^2)$ and it is written in math mode. If the expression is too complex, or too long, you can use two **$** symbols on each side, creating a chunk for the math expression:

$$f\left(x|\mu,\sigma\right)=\frac{1}{\sqrt{2\pi\sigma^2}}\exp\left(-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2\right).$$

There are plenty of tutorials and generators of LaTeX code online, so this feature might be useful.

About the Code in your file

Finally, R Markdown can run R code inside chunks. You can create a new chunk using the shortcut **Ctrl + Alt + I (Cmd + Option + I on macOS)**. You can give names to those chunks, but **if two chunks in the .Rmd have the same name, you will not be able to generate the pdf**.

For the exam, usually we do not want to see your code, only your results, which means that those chunks of R code should not be included. This is possible by including arguments inside the curly brackets of your chunk. First, let us run a function hiding the code chunk by using the argument `echo = FALSE`.

```
##                    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
## Valiant           18.1   6  225 105 2.76 3.460 20.22  1  0    3    1
```

Note that the `echo = FALSE` parameter was added to the code chunk to prevent printing of the R code that generated the results, so only what was shown in the console was printed.

Not all messages from the console are desirable, for example, let us turn on the *tidyverse* package:

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.4     v readr     2.1.5
## v forcats   1.0.0     v stringr   1.5.1
## v ggplot2   3.5.1     v tibble    3.2.1
## v lubridate 1.9.3     v tidyr     1.3.1
## v purrr     1.0.2
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become error
```

Note that those messages are not important to the analysis, so you may want to use the `message = FALSE` argument. You can also surpress undesirable warnings using the `warning = FALSE` argument.

You can include multiple arguments in the curly brackets, separating those arguments with a comma.

A better option if your simply want the code to run and surpress both the code and the results from the code from appearing in the final file is to use the `include = FALSE` argument. Do not forget that no results will be printed by using this option, se be careful.

For example, I will created an object in one chunk with the `include = FALSE` argument, but the result will not be printed.

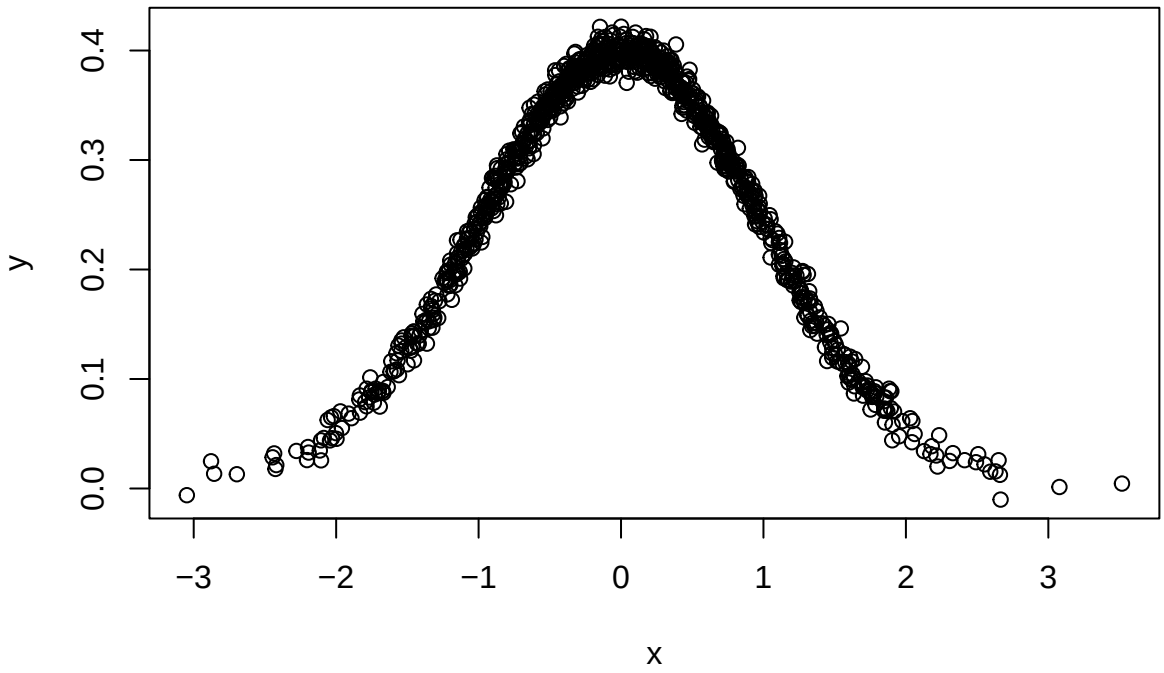
However, I can call the same object in another chunk.

```
## [1] 55
```

Basically you need to separate your chunks properly to show only the desirable results.

Including Plots

Finally, let us generate some data and print a plot.



It is as easy as it looks. Note that the `echo = FALSE` argument surpressed the code and only the plot was printed. You can also include arguments in your chunk that will change the features of your plot, such as width, height and alignment.