COMP120: Lab 10 Worksheet

Welcome to the final lab of COMP120. We have now covered all the major activities pertaining to the data science workflow (programming, importing, tidying, visualising, modelling etc.). In this lab, we introduce another important aspect of data science – communicating your work. As noted in the lectures, it is very important that your work is reproducible by others, and the easiest way to do this is to provide all the code and supporting resources (e.g. data files, external images) in one project and let others “build” your final report. We call this *literate programming*. The lab introduces the method of literate programming that is native to RStudio called **R Markdown**.

Most of what follows is taken from the R Markdown chapter of “R for Data Science”[[1]](#footnote-1).

# Introduction

R Markdown provides an unified authoring framework for data science, combining your *code, its results, and your prose commentary*. R Markdown documents are fully reproducible and support dozens of output formats, like html, word files, PDFs, slideshows, and more.

R Markdown files are designed to be used in three ways:

1. For communicating to decision makers, who want to focus on the conclusions, not the code behind the analysis.
2. For collaborating with other data scientists (including future you!), who are interested in both your conclusions, and how you reached them ( i.e., the code).
3. As an environment in which to *do* data science, as a modern day lab notebook where you can capture not only what you did, but also what you were thinking.

R Markdown integrates a number of R packages and external tools. This means that help is, by-and-large, not available through ?. Instead, as you work through this lab document, and use R Markdown in the future, keep these resources close to hand:

* R Markdown Cheat Sheet: *Help > Cheatsheets > R Markdown Cheat Sheet*,
* R Markdown Reference Guide: *Help > Cheatsheets > R Markdown Reference Guide*.

Both cheatsheets are also available at <http://rstudio.com/cheatsheets>.

# R Markdown – The Basics

This is an R Markdown file, a plain text file that has the extension .Rmd. You have been already given this file called diamond\_sizes.Rmd.

---

title: "Diamond sizes"

date: 2016-08-25

output: html\_document

---

```{r setup, include = FALSE}

library(tidyverse)

smaller <- diamonds %>%

filter(carat <= 2.5)

```

We have data about `r nrow(diamonds)` diamonds. Only

`r nrow(diamonds) - nrow(smaller)` are larger than

2.5 carats. The distribution of the remainder is shown

below:

```{r, echo = FALSE}

smaller %>%

ggplot(aes(carat)) +

geom\_freqpoly(binwidth = 0.01)

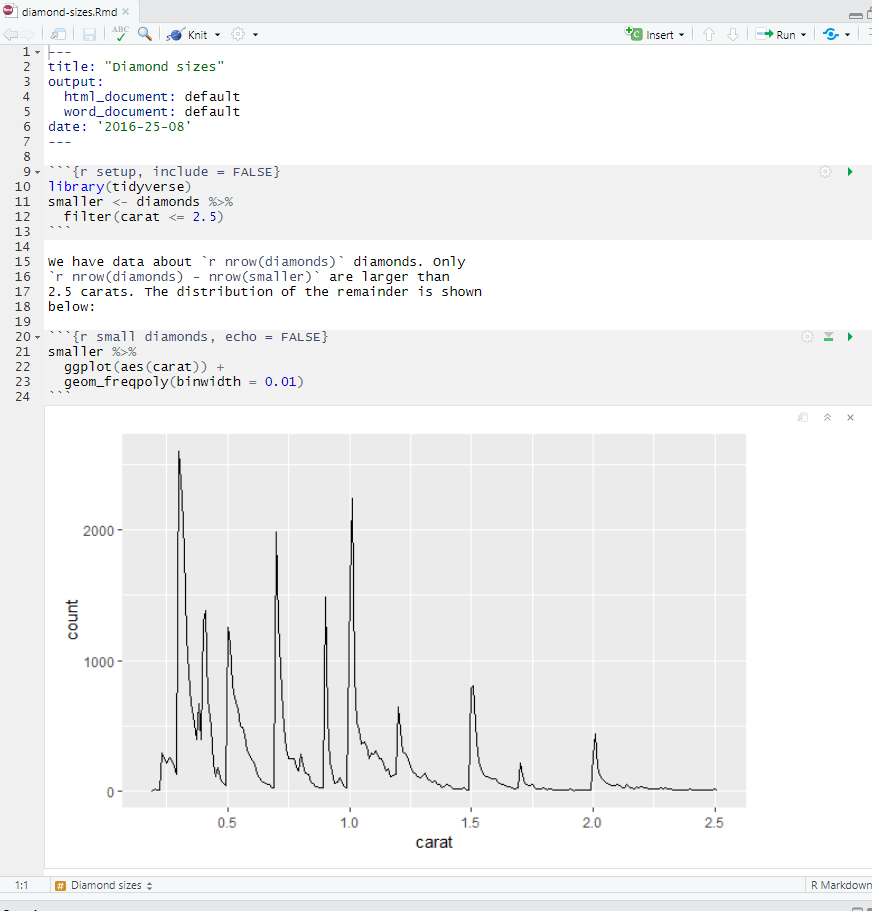
```

It contains three important types of content:

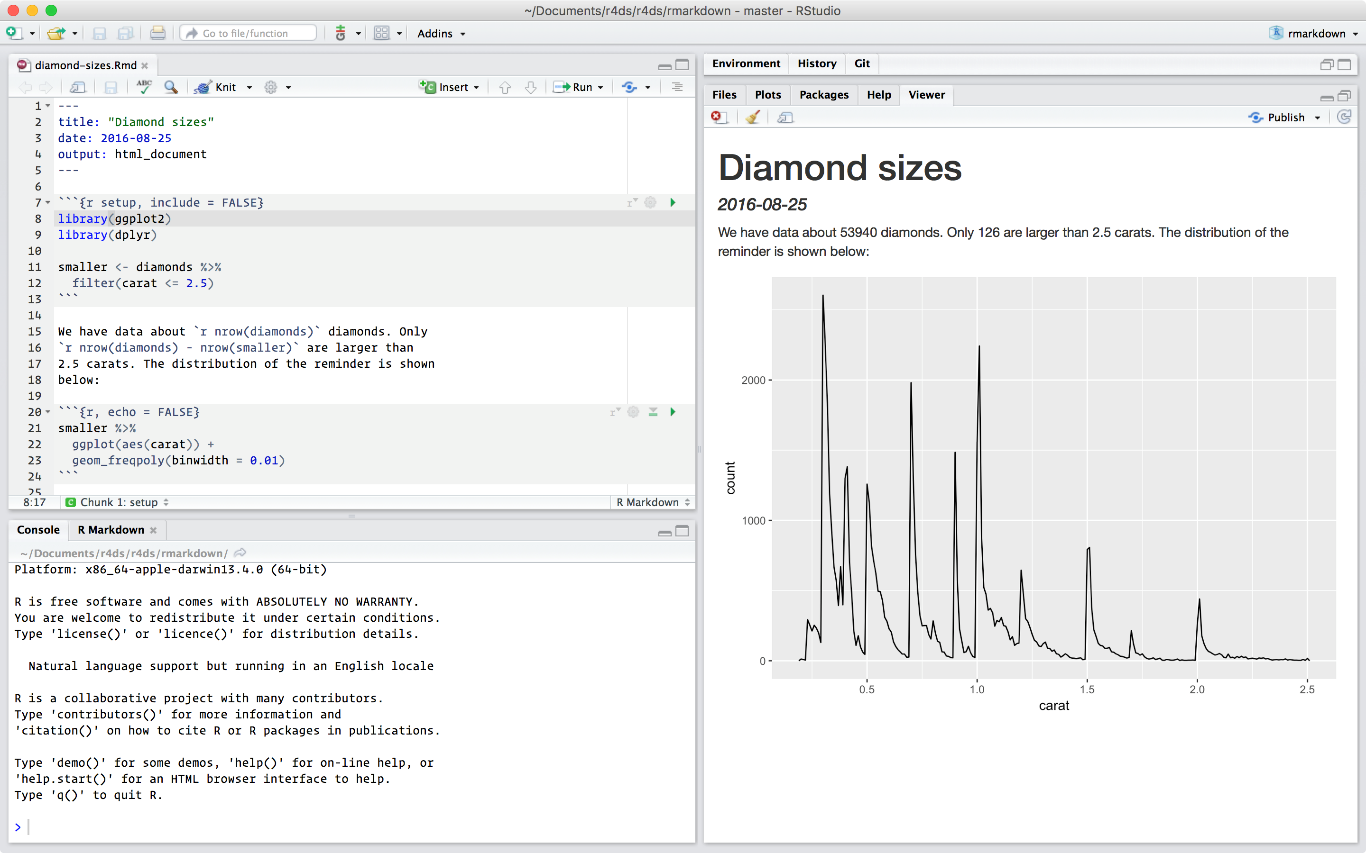
1. A **header** surrounded by --- (the three lines at the top part of the snippet above encapsulated within --- lines). This contains details such as the title of the document, the author name and the output format for the document (html, word, pdf etc. – html in this case).
2. **Chunks** of R code surrounded by three back tick marks (i.e., ```).
3. **Inline R code** - text mixed with simple text formatting like # heading and \_italics\_ also containing R code.

We’ll see more about these in a short while.

When you open an .Rmd file, you get a notebook interface where code and output are interleaved. You can run each code chunk by clicking the Run icon (it looks like a play button at the top right hand corner of the chunk), or by pressing Cmd/Ctrl + Shift + Enter. RStudio executes the code and displays the results inline with the code as shown below.



To produce a complete report containing all text, code, and results, click “Knit” or press Cmd/Ctrl + Shift + K. You can also do this programmatically with rmarkdown::render("diamond\_sizes.Rmd"). This will display the report in the viewer pane (or a browser window), and create a self-contained HTML file titled diamond\_sizes.html in your working directory that you can share with others. You can also select the ‘Knit to Word’ option by clicking on the downward pointing arrow next to “Knit” command. That will produce a word document named diamond\_sizes.docx which is stored in the current directory.



When you **knit** the document, R Markdown sends the .Rmd file to **knitr** (<http://yihui.name/knitr/>), which executes all of the code chunks and creates a new markdown (.md) document which includes the code and its output. The markdown file generated by knitr is then processed by **pandoc** (<http://pandoc.org/>), which is responsible for creating the finished file. The advantage of this two step workflow is that you can create a very wide range of output formats.



To get started with your own .Rmd file from the scratch, select *File > New File > R Markdown…* in the menubar. Provide an appropriate title for the file (e.g., My First Markdown) and the author’s name (your name). Retain the default output to be HTML. RStudio will pre-populate your file with useful content that reminds you how the key features of R Markdown work. The file that get created should be given a name. Save the file as myfirstexample.Rmd using *File > Save as* option. Knit the file into an html document and examine the output. Also, have a quick browse through the contents of the .Rmd file to see the different types of content present in this file. You’ll understand the different types of content in the file as you progress through the lab. Note, these were also covered in lecture 21.

The following sections dive into other components of an R Markdown document in more details: formatting text and using code chunks to record snippets (or blocks) of code.

# Text formatting with Markdown

Prose in .Rmd files is written in Markdown, a lightweight set of conventions for formatting plain text files. Markdown is designed to be easy to read and easy to write. It is also very easy to learn. The guide below shows how to use Pandoc’s Markdown, a slightly extended version of Markdown that R Markdown understands.

Text formatting

------------------------------------------------------------

\*italic\* or \_italic\_

\*\*bold\*\* \_\_bold\_\_

`code`

superscript^2^ and subscript~2~

Headings

------------------------------------------------------------

# 1st Level Header

## 2nd Level Header

### 3rd Level Header

Lists

------------------------------------------------------------

\* Bulleted list item 1

\* Item 2

\* Item 2a

\* Item 2b

1. Numbered list item 1

1. Item 2. The numbers are incremented automatically in the output.

Links and images

------------------------------------------------------------

<http://example.com>

[linked phrase](http://example.com)

Tables

------------------------------------------------------------

First Header | Second Header

------------- | -------------

Content Cell | Content Cell

Content Cell | Content Cell

The best way to learn these is simply to try them out. Copy and paste the code above in the myfirstexample.Rmd file you created above (at the end of the existing code in that document). Knit the document and examine the output. It will take more time for you to become comfortable with the syntax, but soon they will become second nature, and you won’t need to think about them. If you forget, you can get to a handy reference sheet with *Help > Markdown Quick Reference*.

Having completed the task above, ask yourself the following questions:

1. How will you create level 1 and level 2 headers?
2. How will you create a bulleted list?
3. How will you create a numbered list?
4. How will you create text in italics?

The answers to these questions should be obvious after looking at the markdown code given in the table above. If not, consult the lab staff.

# Code chunks

To create a code inside an R Markdown document that achieves a specific goal, you need to insert a chunk. There are three ways to do so:

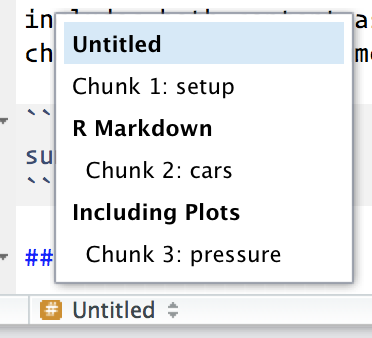
1. The keyboard shortcut command: Ctrl + Alt + I
2. The “Insert” button icon in the editor toolbar and selecting R.
3. By manually typing the chunk delimiters ```{r} and ```.

You can continue to run the code using the keyboard shortcut that by now (I hope!) you know and love: Cmd/Ctrl + Enter. However, chunks get a new keyboard shortcut: Cmd/Ctrl + Shift + Enter, which runs all the code in the chunk. Think of a chunk like a function. A chunk should be relatively self-contained, and focussed around a single task.

The following sections describe the chunk header which consists of ```{r, followed by an optional chunk name, followed by comma separated options, followed by }. It is important that you don’t forget the comma after chunk name before specifying the option – it is a common rookie mistake to forget this. Next comes your R code and the chunk end is indicated by a final ```.

## Chunk Name

Chunks can be given an optional name: ```{r by-name}. This is advantageous because you can more easily navigate to specific chunks using the drop-down code navigator in the bottom-left of the script editor as shown in the picture below. It shows three chunks of code and these are named: setup, cars and pressure.



There is one chunk name that imbues special behaviour: setup. When you’re in a notebook mode, the chunk named setup will be run automatically once, before any other code is run. The other two chunks won’t be automatically executed. Each chunk has to be run individually or the whole document can be knitted.

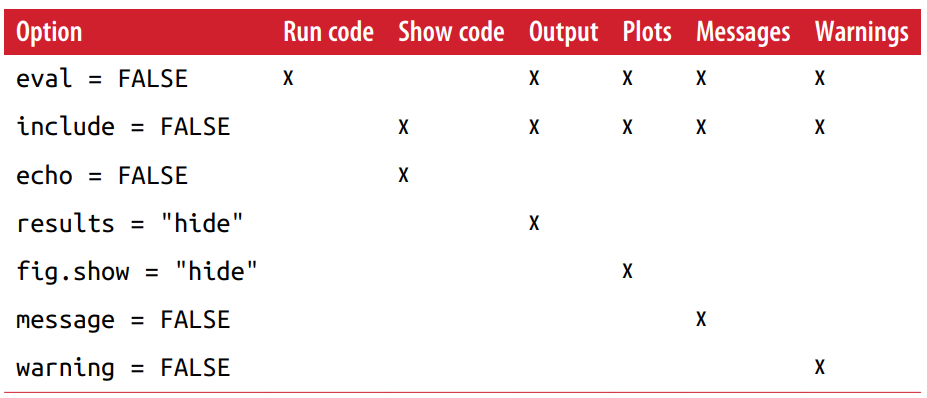
## Chunk options

Chunk output can be customised with **options**, arguments supplied to chunk header. Knitr provides almost 60 options (wow!) that you can use to customize your code chunks. Here we’ll cover the most important chunk options that you’ll use frequently (with the first three being the most important). You can see the full list at <http://yihui.name/knitr/options/>.

The most important set of options controls if your code block is executed and what results are inserted in the finished report:

* eval = FALSE prevents code from being evaluated. (And obviously if the code is not run, no results will be generated). This is useful for displaying example code, or for disabling a large block of code without commenting each line.
* include = FALSE runs the code, but doesn’t show the code or results in the final document. Use this for setup code that you don’t want cluttering your report.
* echo = FALSE prevents code, but not the results from appearing in the finished file. Use this when writing reports aimed at people who don’t want to see the underlying R code.
* message = FALSE or warning = FALSE prevents messages or warnings from appearing in the finished file.
* results = 'hide' hides printed output; fig.show = 'hide' hides plots.
* error = TRUE causes the render to continue even if code returns an error. This is rarely something you’ll want to include in the final version of your report, but can be very useful if you need to debug exactly what is going on inside your .Rmd. It’s also useful if you’re teaching R and want to deliberately include an error. The default, error = FALSE causes knitting to fail if there is a single error in the document.

The following table summarises which types of output each option suppresses:



## Inline Code

There is one other way to embed R code into an R Markdown document: directly into the text, with: `r `. This can be very useful if you mention properties of your data in the text. For example, in the example document we used at the start of the lab (diamond-sizes.Rmd) we had:

We have data about `r nrow(diamonds)` diamonds. Only `r nrow(diamonds) - nrow(smaller)` are larger than 2.5 carats.

When the report is knit, the results of these computations are inserted into the text:

We have data about 53940 diamonds. Only 126 are larger than 2.5 carats.

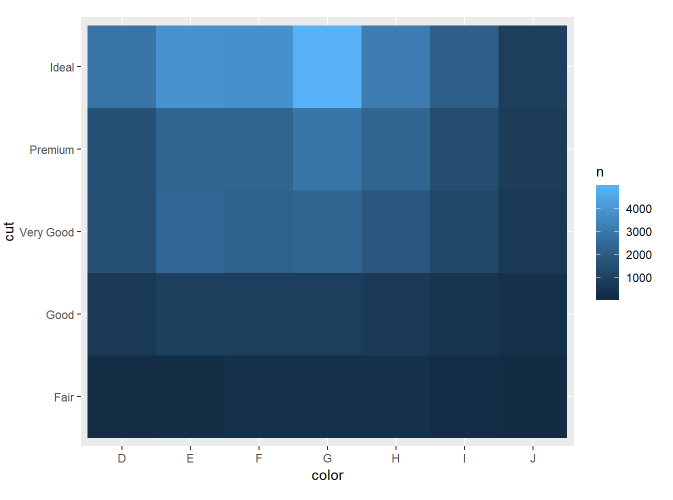
## Troubleshooting

Troubleshooting R Markdown documents can be challenging because you are no longer in an interactive R environment, and you will need to learn some new tricks. The first thing you should always try is to recreate the problem in an interactive session. Restart R, then “Run All” (under Run region), or with the keyboard shortcut Ctrl + Alt + R. If you’re lucky, that will recreate the problem, and you can figure out what’s going on interactively. Also, make sure that the Rmd file is in your current working directory.

If these don’t help, then copy paste the code that is causing problems with associated setup code into a script (R file) and then identify the problem. Once you have rectified the problem you can copy paste the code to the .Rmd file.

# To do tasks

1. In the diamond-sizes.Rmd file, add a new code chunk. That code chunk should contain the code for creating the tile plot showing the covariation between *cut* and *colour* variables in the diamonds dataset. Use the code given in slide 19 of lecture 14 (you can copy the code from the demo file provided for that lecture also). When this code chunk is run, it should show only the output (i.e., it should not show the code – so, you need to choose the right chunk option). Also, provide an appropriate name for this code chunk. The output after running the code chunk is given below. Modify the code in the chunk to include an appropriate title for the plot (currently missing!).



1. Add a new code chunk that will contain the code that shows the relationship between four variables in the diamond dataset using the facet\_grid function. Use the code given in slide 25 of lecture 14 for this purpose (you can copy the code from the demo file provided for that lecture also). The snapshot of the output is given below. Modify the R code to include an appropriate title for the graph (currently missing!). Also, give the code chunk an appropriate name. When the whole document is knit as an html file, the output should contain both the code and the plot as shown below. Remember, you must load appropriate library within the *setup* code chunk (e.g. tidyverse).



1. Write text that has embedded *inline code* that shows the following output. Hint: use min() and max() functions to get the values (326 and 18823).

The price range of diamonds is between 326 and 18823.

1. Now, knit the file as a word document and examine the output. Also, knit the file into an html file and examine the output. You should now have adequate skills to move on to the mastery tasks for this week. So, let us head there ☺

# Mastery X Tasks

Yay, you have reached the last mastery for this course! For this last mastery, you will need to create an Rmd file. You should name this file *mastery-10.Rmd*. When this document is knit, it should produce the output shown in *mastery-10.docx* file which has been given to you. So, examine this document thoroughly before starting to add content to your Rmd file.

As you may have noticed in examining *mastery-10.docx* file, the mastery task involves analysing two New Zealand politicians’ tweets (prime minister Jacinda Ardern’s tweets and deputy prime minister Winston Peter’s tweets). These publicly available tweets were obtained from Twitter using Twitter API at the end of 2019 which have also been given to you. Your job is to produce an Rmd file that has code, results and the explanation of the analyses you have conducted. Remember, this assignment is all about gaining skills on creating R Markdown file and knitting it into a word document (and not about politics!). But, politics provides an interesting context. If you want to dig deep into any insights for politics from these tweets you are welcome to, but that is outside the scope of this course 😊.

In the mastery-10.docx file you will find several instances of “To do” and these must be replaced with appropriate explanation. Also, any numbers indicated in the document (e.g. 1593 tweets) will need to be replaced with appropriate inline R code. You will also need to include appropriate details to the “About me” section. To identify what details need to be added/modified in the template provided, we have provided *mastery-10-annotated.docx* that contains annotated comments throughout the document. You must also look at this file carefully and read the comments provided before starting to work on your Rmd file.

Submit your file (mastery-10.Rmd) on Blackboard. You must **submit your work on Blackboard before 4pm on Monday the 5th of October**. *Note, this extension has been provided as you have your second test on 2nd of October.*

1. <https://r4ds.had.co.nz/r-markdown.html> [↑](#footnote-ref-1)