Lab 3 - Introduction to RStudio and R Markdown. Presentations

Infinity, Michael Eirikson

20-Sep-2025

In Lab 3, you will learn how to use, edit and create a R Markdown document (like this one) using RStudio. You should follow the instructions in this document to complete the assignment. Knit this document to view the nicely rendered HTML, which can make it easier to read the questions.

If you need help as you use R Markdown in this lab and others in the future, consult the following resources:

- Cheat sheet
- Home page with guides
- Reference book

The below is a code chunk, but instead of using the `r` engine we're creating and alert block that will make the question show up with a blue background in the HTML output. Unfortunately, this creates and error when exporting to PDF, so it can only be used for HTML.

Submission Instructions

rubric={mechanics:3}

You receive mark for submitting your lab correctly, please follow these instructions:

- Follow the general lab instructions.
- Click here to view a description of the rubrics used to grade the questions.
- Push your `.Rmd` AND **all the files** you will create as part of the lab to your GitHub repository.
  - The reason for pushing all the files is that `.Rmd` does not contain the rendered output from running the cells. If someone is checking out your work there needs to be an HTML file to view the output, so it is good to get into this habit. - `.ipynb` renders nicely on GitHub, which is why we did not include the HTML file for previous labs.
- Upload a `.Rmd` version of your assignment to Gradescope.
- Include a clickable link to your GitHub repo for the lab just below this cell (it should look something like this https://github.ubc.ca/MDS-2022-23/DSCI_521_labX_yourcwl).

Repo link:

https://github.ubc.ca/mds-2025-26/DSCI_521_lab3_group05

Editing R Markdown documents

This document is called an R Markdown document. It is a literate code document, similar to Jupyter notebooks where you can write code and view its outputs. To start, let's set our working directory by creating a new R Project for lab 4.

Text and rendering R Markdown documents

In a R Markdown document any line of text not in a code chunk (like this line of text) will be formatted using Markdown. Similar to JupyterLab, you can also use HTML and LaTeX here to do more advanced formatting. To run a code chunk, you can press the green play button in the top right corner of the chunk.

Question 1 rubric={correctness:1}

As you work on this lab you will be rendering the document(s). Please make sure all source and rendered documents are in the repository.

I've talked in class about making sure you ignore rendered output, but for this assignment we'll treat it as an exception because we want to confirm that the documents are properly formatted and rendering.

Nothing to post. We will be looking at your final repository contents.

Question 2 rubric={mechanics:1}

Create a new code chunk below using the r language engine that runs some R code (it does not need to be complicated, but it should have an output). Ensure that you can render/knit the document after you add that chunk.

```
sum_of_some_fib_numbers <- 1 + 1 + 2 + 3 + 5 + 8 + 13
print(sum_of_some_fib_numbers)
```

```
## [1] 33
```

Question 3 rubric={mechanics:1}

Create a new code chunk, and add a meaningful name to the code chunk. Try using the pop-up-like menu to navigate between the named code chunks Don't forget to knit/render the document after you make this change to ensure everything is still working.

```
print('This is a very meaningful print statement')
```

```
## [1] "This is a very meaningful print statement"
```

Question 4 rubric={mechanics:1,reasoning:1}

Create a new code chunk that uses a code chunk option. Write out in your own words what the code chunk option is doing.

```
## [1] 2
```

The echo = FALSE option stops the code from being printed to the console when run. Only the results should show.

Multiple code chunk options To have multiple code chunk options you separate them by a comma. For example, if in addition to suppressing warnings, we want to run the code but not output the results, then we can add the `include = FALSE` argument to the code chunk after the `warning = FALSE` option.

Question 5 rubric={mechanics:1,reasoning:1}

Create a new code chunk that uses at least two code chunk options. At least one must be different to the ones mentioned above. Write in your own words what each code chunk option is doing.

```
1 + 2
```

```
## [1] 3
```

Answer: The include = TRUE options let the knitr include the chunk and its output, and warning = FALSE suppresses warning messages and echo = TRUE makes sure the code is output not just the results.

YAML Header and document output options

R Markdown files contains three types of content:

1. Plain text mixed with simple Markdown formatting.
2. Code chunks surrounded by “‘.
3. An (optional) YAML header surrounded by `---`.

You have been introduced the first two types of content, but not the third (although you probably saw it at the top of this document). The (optional) YAML header, which is located at the very top of R Markdown files sets some general global parameters, including:

- title
- author
- output
- etc

Example YAML Header

```
---
title: "Reproducible Data Science Report"
author: "Florencia D'Andrea"
date: "September 4, 2022"
output: html_document
---
```

Most important from a workflow perspective is **output**. Possible output options include:

- `output: html_document`
- `output: md_document`
- `output: pdf_document`
- `output: word_document`
- `output: xaringan::moon_reader` (xaringan presentation - html)

Question 6 rubric={mechanics:1}

Navigate to the YAML header at the very top of this document and edit it so that you include an `author` (yourself) and a `date` (lab due date). Include what you added below here as well as a fenced Markdown code block.

q11-shoon

```
---
master
author: "Infinity", "Michael Eirikson"
date: "20-Sep-2025"
---
```

Creating R Markdown documents

You can use the “File” menu inside RStudio to create new R Markdown documents by selecting: **File > New File > R Markdown** This will bring you to another menu where you can choose the type of output (don’t be afraid to pick something, you can always change the `output` type once you have the `.Rmd` file).

To create a written report, we generally recommend using the default `output: html_document` as it is easier to read than PDF (note - LaTeX does not render nicely in such documents sadly, so if you are using a lot of LaTeX then you may want to choose `output: pdf_document`). If you want to create an `.md` file to publish on GitHub, it is recommend to instead use `output: github_document`. To get this from the menu above you need to navigate to the “From Template” option on the left panel and then select “GitHub Document (Markdown)”.

Question 7 rubric={mechanics:2}

1 - Create a new RMarkdown report (a different file than this one) in the same directory as this RMarkdown file. Use `html_document` as the `output`. After you have rendered it, paste the link to the HTML output as a link to your GitHub repository (remember to push all your files!)

2 - Then, navigate to the YAML header at the very top of that `.Rmd` document and edit it so that the `output` is `pdf_document`. Then knit/render the document. Note the different output. Add and commit that rendered both the `.html` and `.pdf` files to the GitHub repository for this lab and paste the two links below this question.

Link to the .html file:

https://github.ubc.ca/mds-2025-26/DSCI_521_lab3_group05/blob/master/html_yaml.html

Link to the .pdf file:

https://github.ubc.ca/mds-2025-26/DSCI_521_lab3_group05/blob/master/html_yaml.pdf

Question 8 rubric={mechanics:6}

1. Go back to the `.Rmd` file you created in question 7, and include at least two Markdown text sections (each should have a header) and at least two separate code chunks in it (these can be really simple). Save the new R Markdown document and give it a new meaningful name.
2. Render/knit the new R Markdown document to get an `.html` file. Put the `.Rmd` document and the rendered `.html` file under version control using Git, and push/upload the file to your GitHub repository for this homework. Paste a link to these files as your answer below.

YOUR ANSWER GOES HERE

Link to .rmd file:

https://github.ubc.ca/mds-2025-26/DSCI_521_lab3_group05/blob/master/simple_adding.Rmd

Link to .html file:

https://github.ubc.ca/mds-2025-26/DSCI_521_lab3_group05/blob/master/simple_adding.html

Question 9 (Optional) rubric={mechanics:1,reasoning:1}

1. Take the R Markdown report created in Question 8 and change the output to `github_document` and render it. Put the rendered `.md` file under version control using Git, and push/upload the file to your GitHub repository for this homework. Try to look at the file on GitHub.ubc.ca in your homework repo? What do you see? How is it rendered?

Its rendered completely as a .md document. For example even a datafram has been changed to a markdown table.

Link to github_document file:

https://github.ubc.ca/mds-2025-26/DSCI_521_lab3_group05/blob/master/simple_adding.md

Question 10 rubric={mechanics:6}

1. Create a presentation using RStudio. Do this in a different file than this one but in the same directory as this RMarkdown file. You can use xaringan or Quarto to create the slides. On the book you will find links that will guide you on how to create each type of slide. We will accept both Xaringan, Quarto, or RISE slides as correct. Give this file a meaningful name.
2. Create at least 4 slides. At least two slides must include a code chunk or cell (these can be really simple). Save the new document.
3. Render/knit/export the new document to get a `html` presentation file.

4. Put the new document and the rendered `.html` file under version control using Git, and push/upload the file to your GitHub repository for this lab
5. Activate GitHub pages and paste the link below. Remember where github looks for website materials, and make sure you put the files in the approriate directory. You can use any of the folder/branches to publish the slides, as long as the URL works.

YOUR ANSWER GOES HERE

Link to presentation

https://meirikson.github.io/DSCI_521_lab3_group05/q10_presentation.html

Link to repo

https://github.com/meirikson/DSCI_521_lab3_group05

(Challenging) Question 11 rubric={reasoning}

In a paragraph or two, compare and contrast the use of reproducible tools (e.g., R Markdown and Jupyter) and non-reproducible tools (Word, Powerpoint, Keynote, etc) for presentations and reports. Include advantages and disadvantages for each.

When choosing tools for presentations and reports, there is a fundamental distinction between reproducible tools (like R Markdown and Jupyter) and non-reproducible tools (like Word and PowerPoint). Reproducible tools allow users to combine executable code with narrative text in a single document. The primary advantage is complete reproducibility – when an error is discovered in calculations (as described in Activity 1 from the lecture), the code can be corrected and the entire document re-rendered, automatically updating all affected results throughout the report. This eliminates manual updates and ensures consistency across the document. As the lecture materials explain, these tools create dynamic documents where "the rendering of the plots, figures, and tables next to the text in a manuscript allows giving valuable context to the visualizations." However, these tools require programming knowledge and present a steeper learning curve for users.

Non-reproducible tools like Word, PowerPoint, and Keynote offer user-friendly interfaces and sophisticated design capabilities, providing extensive formatting options without requiring technical expertise. However, their major limitation lies in manual processes – correcting a calculation error would require users to manually locate and update every affected table, figure, and result throughout the document, a process that is time-consuming and prone to human error. Reproducible tools are most suitable for data-intensive projects that require regular updates and where accuracy is paramount, while non-reproducible tools remain appropriate for presentations where design flexibility and ease of use take precedence over computational reproducibility.

Free Point

Set `free_point` to `TRUE` with `free_point <- TRUE`

```
free_point <- TRUE
```

```
. = ottr::check("tests/free-point-autograde.R")
```

```
## All tests passed!
```

Submission

Make sure you have run all cells in your notebook in order before running the cell below, so that all images/graphs appear in the output. The cell below will generate a zip file for you to submit. **Please save before exporting!**

```
# Save your notebook first, then run this cell to export your submission.
#ottr::export("lab3.Rmd", pdf = TRUE)
```