Introduction to Quarto

Contents

1   Introduction

This week we will take what we will produce a report using Quarto . Quarto is a multi-language system that allows reports and presentation to be created within R, thus allowing for R code and output to be easily embedded within a report, a presentation or even a Blog!

Hence, all of the R code and plots obtained from an analysis are contained within a single file.

Below you can download an example of a report produced with Quarto relating to fitting a regression model. You can toggle the `</>` `Code` button in the html file to see the source code that produced this file. It is advised to have this document open while working through the remaining sections in order to see examples put into practice.

The following sections will take you through the different steps required to produce this report in Quarto. For creating a new Quarto file from scratch see Section 2, otherwise move onto Section 3.

2 Creating your first Quarto document

If you want to create a new Quarto document from scratch within RStudio you can click on on the `New File` icon ⊕ or go to:

```
File -> New File -> Quarto  Document...
```

This will open the following window within RStudio:
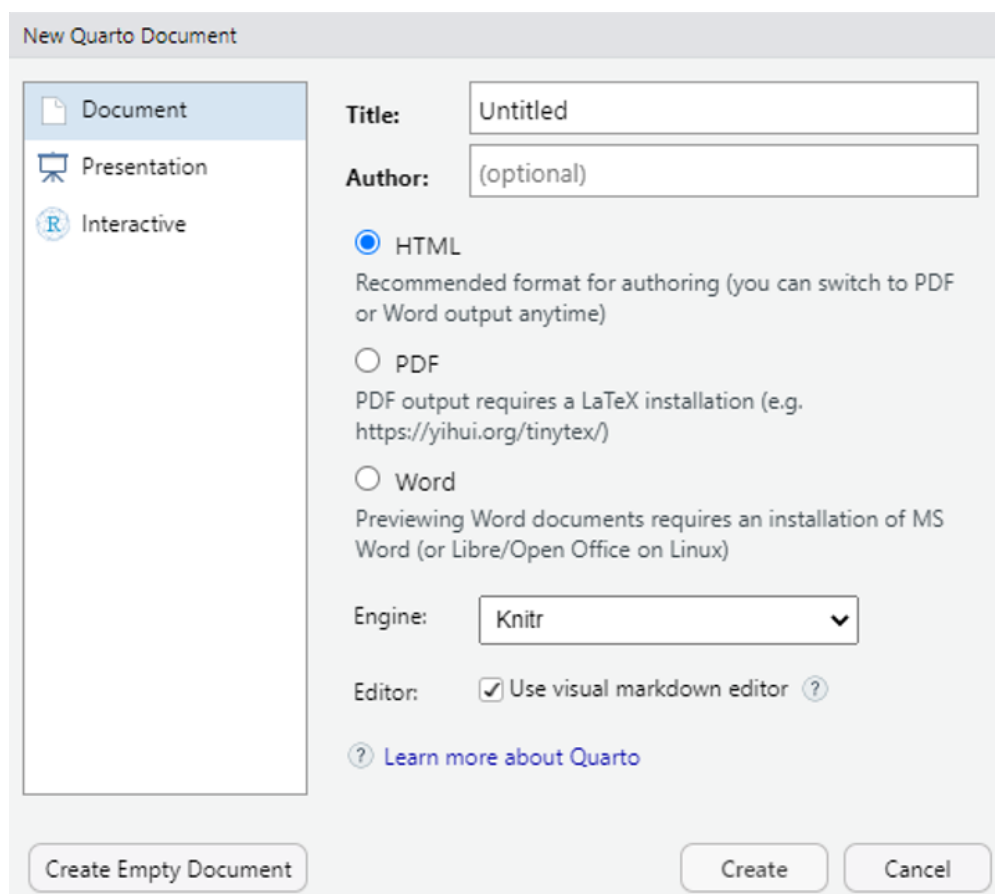
> ❗ **Important**
>
> If you are working on your personal computer be sure that you have the latest RStudio version installed.

From here, select **Document** and the **Default Output Format** as **HTML** (**PDF** output is also possible but you need the latest version of `MikTex` to be installed if you are a Windows

user). Give your document a title and select OK. This will open in RStudio an empty `.qmd` file.

3 Title

The title of the document can be found at the top of the `.qmd` file within its **preamble** (YAML heading), which is shown below. When titling your document, ensure the title is within inverted commas.

```
---
title: "Data Analysis: Example Report"
---
```

A Quarto document can be edited in the two modes of the RStudio editor: visual (on the left) and source (on the right). The visual mode offers a more friendly interface for editing you document (it includes a toolbar for quick edit options), while the source mode allows you to see and edit the underlying markdown text code.
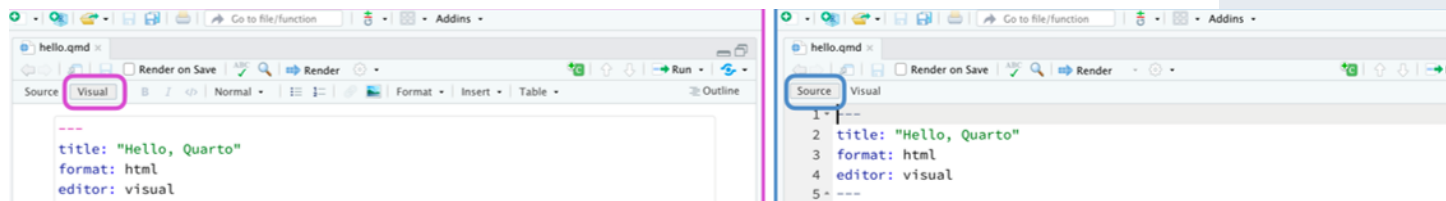
You can switch between these two modes at anytime. We will now cover different aspects of building a Quarto document and provide the instructions on how to do this using either the visual editor or the source mode.

4 Sections

The structure of you quarto document can be determine by the different sections contained in it.

Under the `source` mode, sections within a Quarto document are created using #. For example, # Introduction will create a section titled **Introduction**. Alternatively, if you use the `visual` mode, you can include a new section by selecting a proper heading under the block-format display of the toolbar as follows:
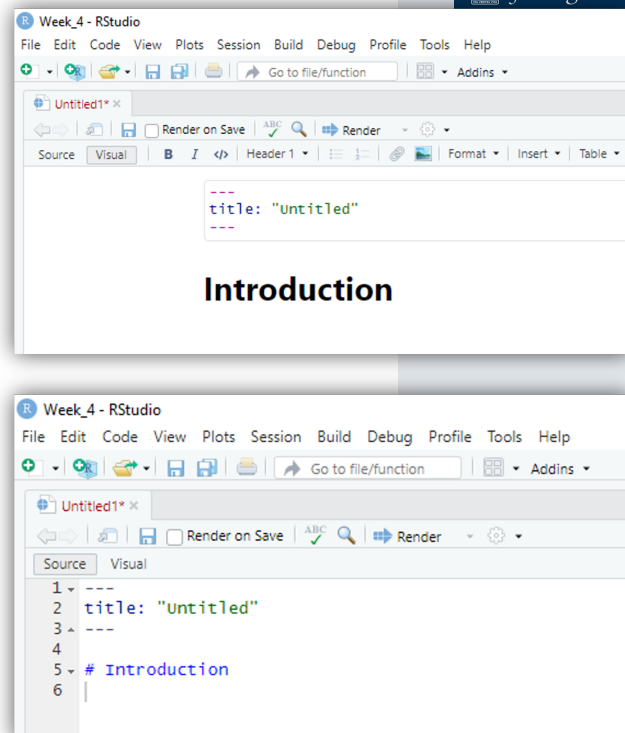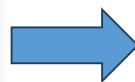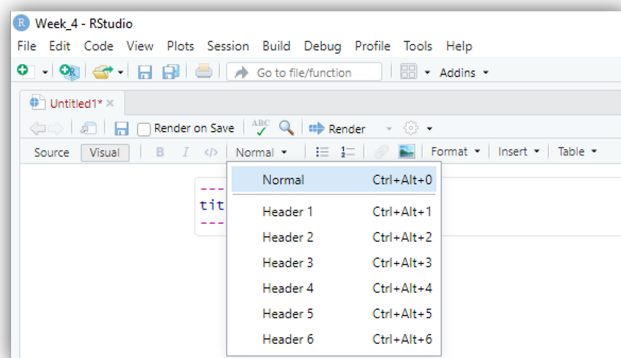
If you want section to be numbered you can set `number_sections: true` within the preamble of the document:

```
---
title: "Data Analysis: Example Report"
number-sections: true
---
```

Task 1

Create a new quarto document by clicking on the `New File` icon ⊕ , change the title to *"Data Analysis: Example Report"* and add numbered-level 1 headings for each of the following project sections:

- Introduction

- Exploratory data analysis

- Formal data analysis

- Conclusions

- References

Then, click the 🔜 `Render` button and compile your Quarto document to make sure that your markdown formatting produces the expected output in your HTML document.

Tip: Try typing / symbol on your main document while using the visual mode and then use the drop menu to add a new section to your document (you can type the letter `h` to search for the different header types while the drop menu is being displayed).

See solution

Your quarto document should look like this if you go to the `source mode`:

```
---
title: "Data Analysis: Example Report"
number-sections: true
---

# Introduction

# Exploratory data analysis

# Formal data analysis

# Conclusions

# References
```

Each section can be assigned labels so that they can be referred to within the text.

5 `visual` **mode**

For the `visual` model you can click on the edit attributes settings ⋯ (right hand side of the section heading) and write the label you want on the ID box of the pop-up window:
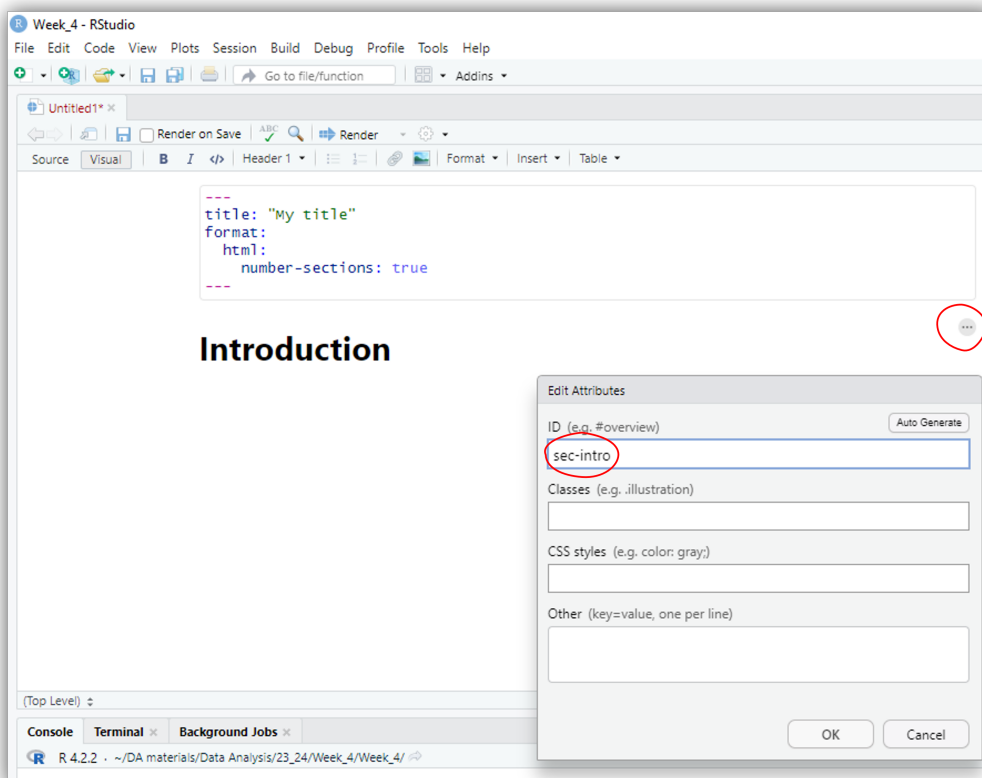
6 `source` **mode**

To reference a section, add a #sec- identifier to any heading. For example, to give our **Introduction** section a label we simply add the label {#sec-intro} to the section title as

follows:

```
# Introduction {#sec-intro}
```

where `sec-intro` is the label chosen for this particular section.

> **!** Important
>
> It is a good idea to label your sections appropriately so that it is easy to refer to them later. Note that it is important to add the `sec-` prefix in order for cross-referencing to work properly.

The section can now be referred to within the text of the document using the `@sec-` command. That is

```
@sec-intro ...
```

will produce

```
Section 1 ...
```

where the 1 is a clickable hyperlink that will take you to the beginning of that section within the document. If you are using the `visual` mode, you can also easily include cross-references by typing `@` and then scroll through the different hyperlinks you created, or by clicking on `Insert   Cross Reference`

Note that for section labels to appear on the references list, you must save your document first (click on `File    Save as…` and save your document with a proper name). Then all the labels (i.e., labels of sections, figures, equations, etc.) will appear on the Insert Cross Reference pop up window. Then simply click on `Insert` to add the selected reference to your document:
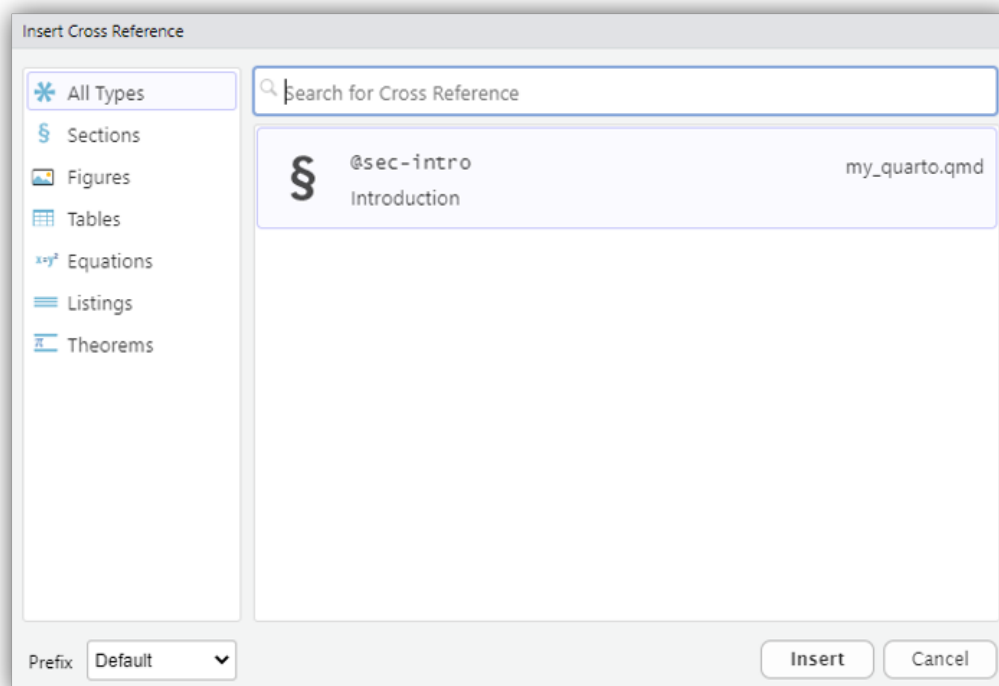
> **ℹ Note**
>
> Subsections can be added to a document in a similar fashion using either `##` (such that `## Subsection {#sec-sub}` will create a subsection with the label `sec-sub` and title **Subsection**) or by clicking on the `Block format` `Header X` menu on the toolbar on the `visual` mode.

> **Task 2**
>
> Add the following level 2 headings (subsections) underneath the `Formal data analysis` heading:
>
> - Statistical model
>
> - Results
>
> - Model diagnostics
>
> Then, add appropriate labels for each section. Render 🔁 your Quarto document and check the HTML output.
> see solution
> Your quarto document should look like this if you go to the `source mode`:
>
> ```
> ---
> title: "Data Analysis: Example Report"
> number-sections: true
> ---
>
> # Introduction {#sec-intro}
>
> # Exploratory data analysis {#sec-eda}
>
> ## Statistical model {#sec-stats}
>
> ## Results {#sec-results}
>
> ## Model diagnostics {#sec-diagnostics}
>
> # Formal data analysis {#sec-fda}
>
> # Conclusions {#sec-con}
>
> # References
> ```

7 Adding references

References can be included by clicking on `Insert` `@Citation`. This will pop-up a new window where you can look up for a reference, add the citation for an R package that is installed in your local computer or add a citation based on the DOI code of a paper or book:

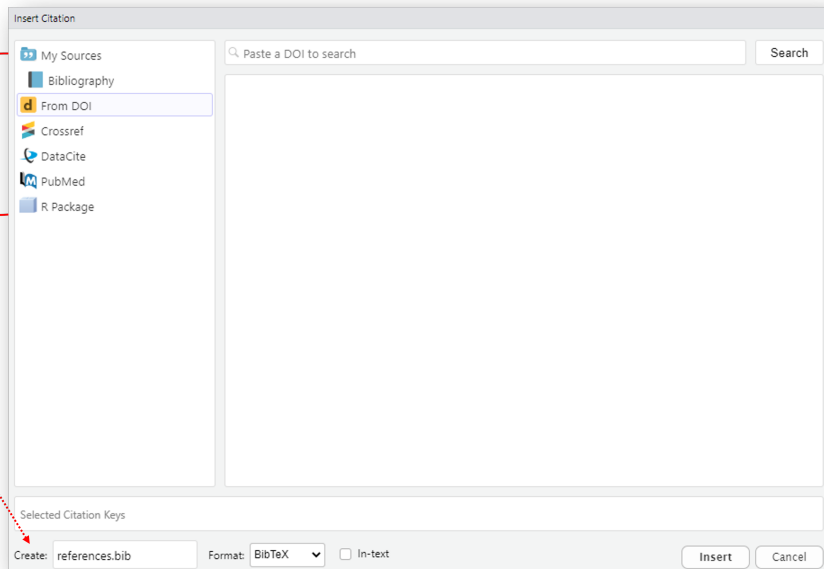The reference you add will be saved to a `*.bib` that you can access at the location where your `.qmd` file is stored. Once a reference is added to your document, the YAML heading will display the option `bibliography: references.bib` which is the source file where your references are being managed.

Task 3

1. Copy and paste the following text under the `Introduction` section:

```
There are hundreds of species of flowering plants, with one such genus known as the iris.

Section 2 consists of an exploratory analysis of the iris data and explores the potential
```

2. Then, change the species names (setosa, versicolor, and virginica) to *italics* format and add a citation [CITE] for the iris data set using the following doi: `10.24432/C56C76`.
3. Lastly, change the text referring to to Section 2, 3 and 4 with clickable hyperlinks to each of these sections. Render ➡ your Quarto document and check the HTML output.

see solution

The introduction section of your quarto document should look like this if you go to the `source mode`:

```
---
title: "Data Analysis: Example Report"
number-sections: true
---

# Introduction {#sec-intro}

There are hundreds of species of flowering plants, with one such genus known as the iris.

@sec-eda consists of an exploratory analysis of the iris data and explores the potential r
```

---

8 Embedding R code

8.1 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.

R Code can be evaluated directly on the Quarto document or in the R console (the latter would be similar to run your code from an R script). To select where you want your code to be evaluated click on the setting options ⚙ (next to the Render button ⇨) and choose between `Chunk Output Inline` (default) if you want your code to be evaluated within the Quarto document or `Chunk Output in Console` to evaluate your code directly on the R console. If you select the latter, the following options will be added to the document preamble:

```
editor_options:
  chunk_output_type: inline
```

To add an R Code Chunk you can simply click on the symbol (or using keyboard shortcut `cmd+alt+i` or `ctrl+alt+i` for windows users) on either `visual` or `source` modes.

R code chunks are identified with `{r}` with multiple (optional) chunk options which we can access by typing `#|` at the beginning of the line.

9 `visual` mode view

If you are on visual mode the R chunks will appear as follow:
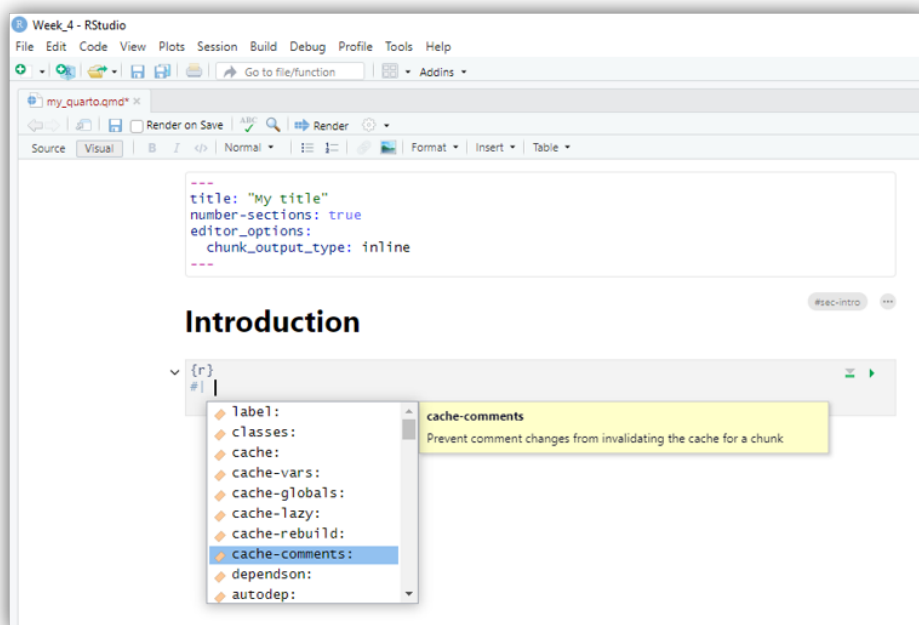
10 `source` **mode view**

If you are in the source mode the R chunks will appear as follow:

```
```{r}
#|


```
```

Some of the most common arguments you will use in your R Chunks are:

- **echo**: include the R code within the code chunk in the document (true/false);
- **eval**: evaluate the R code within the code chunk (true/false);
- **warning**: suppress warnings from R (true/false); and
- **message**: suppress messages from R (true/false).

For example, let's say we wanted to select the `Sepal.Length`, `Sepal.Width` and `Species` variables from the `iris` data set (available by default from the `datasets` library)to be used later, we can do that using the following code chunk:

```r
{r}
#| echo: false
library(tidyverse)

iris_subset <- iris %>%
    dplyr::select(Sepal.Width, Sepal.Length, Species)
```

Setting `echo: false` will tell the R to evaluate the code while hiding the R chunk source code from the main document. In this example, we will store the subsetted data set as the object `iris_subset` so that it can be used in chunks downstream. If you want to embed the source code within the Quarto document then you would simply set `echo: true`.

> **!** Important
>
> R chunk options are case sensitive! so `echo: TRUE` or `echo: FALSE` won't work! You can use R Studio autocomplete feature the see the available options by pressing the tab key.

Note that by default, and unless otherwise specified, chunk codes in Quarto will be set to `echo: true`. Alternatively, you can edit the Quarto YAML preamble and change the global output options to show or hide all R chunks. You can do this within the `execute` options as follows (you can manually override this by changing the `echo` option in each individual chunk) :

```
---
title: "Data Analysis: Example Report"
number-sections: true
execute:
  echo: false
---
```

> **❗ Important**
>
> You need to be very careful with the indentation if you decide to change the global settings of Quarto preamble!

Lets look at another example:

```
#| message: false
#| warning: false
library(ggplot2)
```

In this case, we set `warning` and `message` to `false` to indicate that we want to suppress any warnings or messages (e.g., the ones that you usually get when loading R packages). Another useful option is `eval`. If we set `eval: false` then we indicate that we don't want the chunk output to be included in the rendered document (this is useful for example if we just want to show our R code without rendering its output). For example the following chunk will only show the code for creating `ggplot` object without actually evaluating it:

```
#| eval: false
ggplot(iris_subset, aes(x = Sepal.Length,
                 y = Sepal.Width,
                 color = Species)) +
  geom_point()
```

> **ℹ Note**
>
> Note that we can still run the code in each individual chunk (even if we set `eval: false`) by clicking the run chunk bottom ▶ . The code will be evaluated in either your `.qmd` file or in the console depending on the Chunk output settings (this mean that you can test your code in R but its output won't be rendered in the final document). **This means you don't need a separate R script to test your code!**

Additional arguments can be passed to code chunks other than those displayed above. The most useful ones other than those relate to figure sizing and positioning and are discussed in the upcoming sections.

> **Task 4**
>
> 1. Add a new chunk at the beginning of your document that loads the following R packages:
>
> ```
> library(ggplot2)
> library(tidyverse)
> library(kableExtra)
> library(performance)
> ```
>
> Make sure that the source code and messages/warnings outputs from loading these libraries are hidden from your rendered document.
>
> 2. Then, add another chunk beneath the exploratory analysis section where you create a subset of the `iris` data set containing the variables `Sepal.Width`, `Sepal.Length` and `Species`. Make sure that the source code for this chunk is also hidden in the final document.

Then, click the ➡️ Render button and compile your Quarto document to make sure that your markdown formatting produces the expected output in your HTML document.

See solution

The two chunks you have created should look as follows:

```
#| echo: false
#| warning: false
#| message: false

library(ggplot2)
library(tidyverse)
library(kableExtra)
library(performance)
```

```
#| echo: false
  iris_subset <- iris %>%
    dplyr::select(Sepal.Width, Sepal.Length, Species)
```

10.1   Inline code

R code can be included within text by enclosing the code with `` `r ` ``. This allows for expressions to be evaluated by R and not be hardwired by the user. For example, if you wanted to convey the number of observations within `iris_subset` then we can enclose `nrow(iris_subset)` within `` `r ` `` to obtain the number of observations, rather than hardwiring 150 into the text. This can help to prevent potential human error when presenting information. It can also help with consistency and ease-of-use, since `n = nrow(iris_subset)` could be stored as an R object and referred to whenever necessary within the text using inline R code.

11 Tables

There are several ways to produce tables in Quarto. Here, a couple of different approaches will be presented. The first approach uses the `kable()` function from the `kableExtra` package and essentially receives a data frame as input and then puts a wrapper around the tables produced in R in order to make them more visually appealing within the Quarto document. Here we will just cover the basics, but if you want to learn more about creating eye-catching tables with `kableExtra` visit here.

Let's say we wanted to create a table of the mean and std. dev of the sepal Length and width for each species in the `iris_subset` data. We can create the table using the `kable()` function as follows:

```
  iris_summary <- iris_subset %>%
    summarise('Sepal.Width.Mean' = mean(Sepal.Width),
              'Sepal.Width.sd' = sd(Sepal.Width),
              'Sepal.Length.Mean' = mean(Sepal.Length),
              'Sepal.Length.sd' = sd(Sepal.Length),
```

```
              .by = Species)
  iris_summary %>%
    kable()
```

Species	Sepal.Width.Mean	Sepal.Width.sd	Sepal.Length.Mean	Sepal.Length.sd
setosa	3.428	0.3790644	5.006	0.3524897
versicolor	2.770	0.3137983	5.936	0.5161711
virginica	2.974	0.3224966	6.588	0.6358796

Lets customize this table a bit by (1) adding a table caption and a cross-referencing number (2) modifying the labels of the columns and (3) rounding the number of decimal places to two:

```{r}
#| label: tbl-iris
#| tbl-cap:  Mean and standard deviation (sd) sepal width and length by species of iris.

iris_summary %>%
  kable(col.names = c("Species",
                      "Sepal mean width (cm)",
                      "Sepal width std.dev (cm)",
                      "Sepal mean length (cm)",
                      "Sepal width std.dev (cm)"),
        digits = 2)
```

Table 2: Mean and standard deviation (sd) sepal width and length by species of iris.

Species	Sepal mean width (cm)	Sepal width std.dev (cm)	Sepal mean length (cm)	Sepal width std.dev (cm)
setosa	3.43	0.38	5.01	0.35
versicolor	2.77	0.31	5.94	0.52
virginica	2.97	0.32	6.59	0.64

The first thing we've done is to add a table label using `label: tbl-iris` in the chunk options.This will allow us to add cross-reference in the text (remember to use the `tbl-` prefix to make them cross-referenceable). For example we can write `@tbl-iris` in our document to reference and create an hyperlink directed to the Table 2 (you can also use the tool bar `Insert  Cross Reference` to do this). Then, the chunk option `tbl-cap:` allow us to add caption to the table. To make the table more appealing we can make the table more appealing by setting the number of decimals to 2 by setting `digits = 2`. Lastly, we modify the original column names via the `col.names` argument.

> **Task 5**
>
> Add two separate chunks in your exploratory analysis that produce the following tables:
>
> 1. A table displaying the mean, median and standard deviation for sepal width and length for each of the three different species of iris.
>
> 2. A table displaying the Pearson correlation between sepal width and length by

species.

Round each value to 2 decimals and use appropriate column headings for each table.
Then add a sensible table caption and a cross-referencing number. Comment on this
output in the main body of your document while using clickable hyperlinks whenever
referencing each table.
Before you render your document make sure that the source code for these chunks
is hidden.
Take hint
Use the `summarise` function in conjunction with the `cor()` function to compute the
Pearson correlation between sepal length and width for each species.
See solution
Your chunks should look something like this:

```
#| echo: false
#| label: tbl-summary
#| tbl-cap:  Mean, median and standard deviation (sd) sepal width and length by species of

iris_summary <- iris_subset %>%
    summarise('Sepal.Width.Mean' = mean(Sepal.Width),
              'Sepal.Width.Medain' = median(Sepal.Width),
              'Sepal.Width.sd' = sd(Sepal.Width),
              'Sepal.Length.Mean' = mean(Sepal.Length),
              'Sepal.Length.Medain' = median(Sepal.Length),
              'Sepal.Length.sd' = sd(Sepal.Length),
              .by = Species)

iris_summary %>%
  kable(col.names = c("Species",
                      "Sepal mean width (cm)",
                      "Sepalmedian width (cm)",
                      "Sepal width std.dev (cm)",
                      "Sepal mean length (cm)",
                      "Sepal median length (cm)",
                      "Sepal width std.dev (cm)"),
        digits = 2)
```

```
#| echo: false
#| label: tbl-cor
#| tbl-cap: Correlation between sepal width and length by species.

Cors <- iris_subset %>%
        summarise('Correlation' = cor(Sepal.Width, Sepal.Length),
                  .by = Species)
Cors %>%
  kable(digits=2)
```
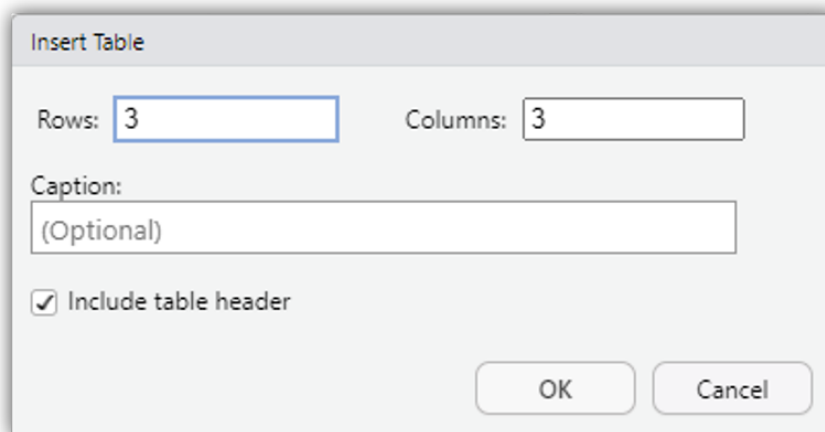
Then, `@tbl-summary` and `@tbl-cor` should be used in the text to reference the cor-
responding table.

11.1 Tables 'by hand'

Tables can also be produced 'by hand' in Quarto. Here is an example of how we can recreate the table above corresponding to the first 5 rows of the `iris` data.

12 `visual` **mode view**

On visual mode, this table can be created by clicking on the `Table` option in the toolbar (next to `Insert` while using the `visual` mode) and selecting the number of rows and columns. Here you can also add a caption to the table.

We can manually fill each entry to produce the following table:

Table 3: The fist 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

13 `source` **mode view**

On `source` mode, the raw markdown code that produces this table is the following:

```
| 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 |
: The fist 5 rows of the iris data. {#tbl-iris}
```

In here, 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 :---:. For these sorts of tables, you can add a caption below the table and then include a #tbl-label in braces at the end of the caption for cross-referencing.

For example, the table above corresponding to the first 5 rows of the `iris` data can be produced by hand by clicking on the `Table` option in the toolbar (next to `Insert` while using the `visual` mode) and selecting the number of rows and columns.

> **ⓘ Note**
>
> You can read more about authoring Quarto tables here.

13.1 Summary of Regression Models as HTML Tables

In this section we briefly introduce the package `sjPlot` to create HTML-tables that will be displayed nicely in your Quarto document. We will focus on the `tab_model()` function, which receives as argument a single or multiple models fitted with the `lm` or `glm` engine (other models, like hurdle- or zero-inflated models, also work with this function).

> **❗ Important**
>
> HTML is the only output-format, you can't (directly) create a LaTeX or PDF output from `tab_model()` and related table-functions.

For example, say we want to create a nice html table of a linear model fitted to the `iris_subset` data set:

```
library(sjPlot)
int.model <- lm(Sepal.Width ~ Sepal.Length * Species, data = iris_subset)
```

Then we can simply add the following chunk to display the Table 4 below:

```
#| tbl-cap: "Summaries of the linear model fitted to the iries  data set."
#| label: tbl-lm_iris
tab_model(int.model)
```

Table 4: Summaries of the linear model fitted to the iries data set.

| | | Sepal Width | |
Predictors	Estimates	CI	p
(Intercept)	-0.57	-1.66 – 0.53	0.306
Sepal Length	0.80	0.58 – 1.02	**<0.001**
Species [versicolor]	1.44	0.03 – 2.85	**0.045**
Species [virginica]	2.02	0.66 – 3.37	**0.004**
Sepal Length × Species [versicolor]	-0.48	-0.74 – -0.21	**<0.001**
Sepal Length × Species [virginica]	-0.57	-0.82 – -0.32	**<0.001**
Observations	150		

Table 4: Summaries of the linear model fitted to the iries data set.

R^2 / R^2 adjusted	0.623 / 0.610

`tab_model()` has some argument that allow to show or hide specific columns from the output:

- `show.ci` to show/hide the column with confidence intervals.

- `show.se` to show/hide the column with standard errors.

- `show.p` to show/hide the column with p-values.

- `show.r2` to show/hide R^2 values.

- `show.obs` to show/hide the number of observation.

Other useful arguments are `collapse.ci` and `collapse.se` which collapse the columns for confidence intervals and standard errors (i.e., one column together with the estimates). You can use the `pred.labels` argument to change the names of the coefficients in the *Predictors* column (note that the length of `pred.labels` must exactly match the amount of predictors in the *Predictor* column).

Lastly, `tab_model()` can also be used to print multiple models at once, which are then printed side-by-side, e.g., `tab_model(model_1, model_2)`. You ca use the `dv.labels` to change the names of the model columns, this way you can give sensible name to each model.

> **i** Note
>
> You can read more about `sjPlot` html regression model tables here.

14 Figures

14.1 Embedding external images

Including plots and figures within a Quarto document is straightforward. To include an external figure you can use the `visual` mode tool bar and click on `Insert Figure/Image...` and then browse to the path where you image is, or alternatively write the *url* from which the image should be obtained. The pop-up windows allows you also to write a caption and also to modify the image alignment with respect the main text.
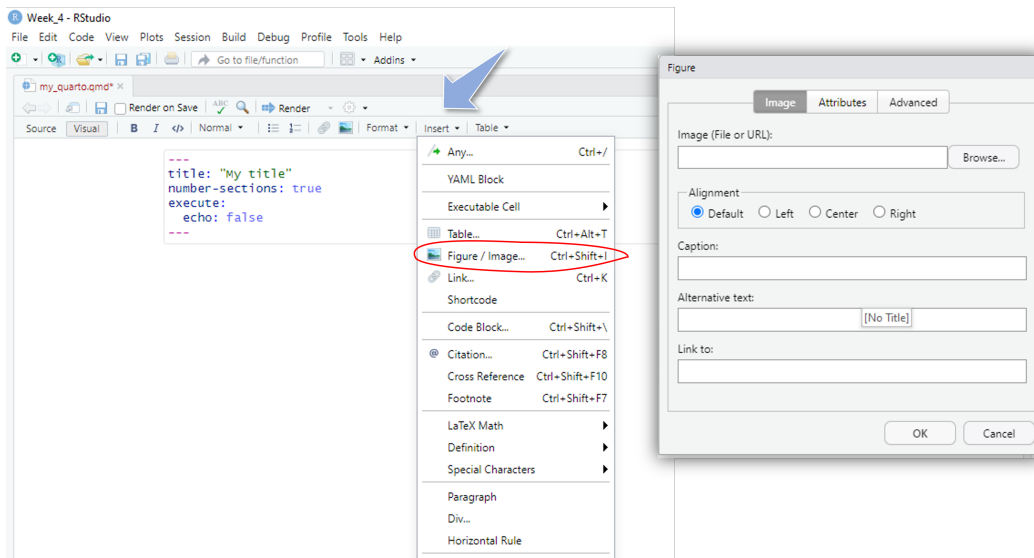
If you want to add cross-referencing you can add the `fig-` prefix in the attributes tab of the pop-up windows:
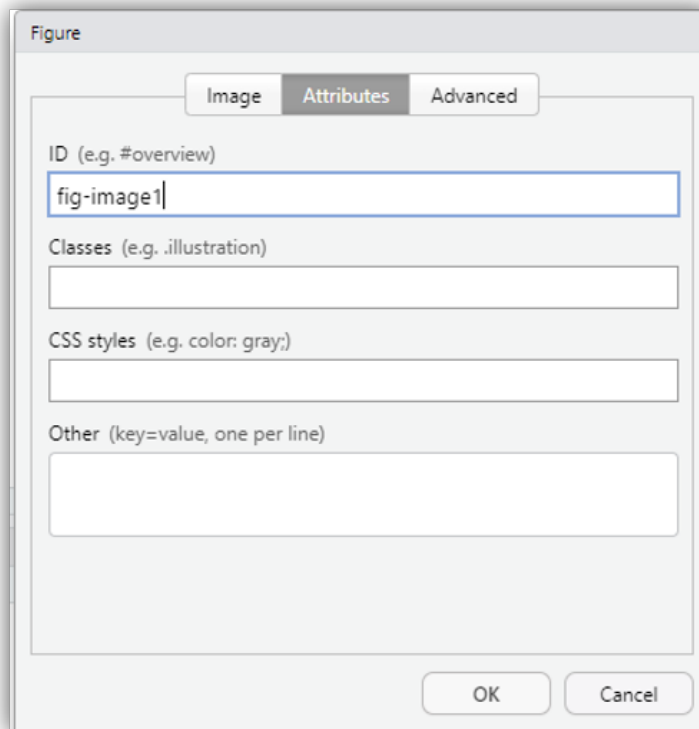
The source code would then look something like:

```
![](images/myimage.png){#fig-image1}
```

14.2   Embedding R generated figures

Very often we would like to use the R plots generated from your analysis instead of using external figures. To achieve this, the R code for the plot is simply included within a code chunk including additional arguments for plot size and positioning. For example, to include the scatterplot of sepal width and length by species:
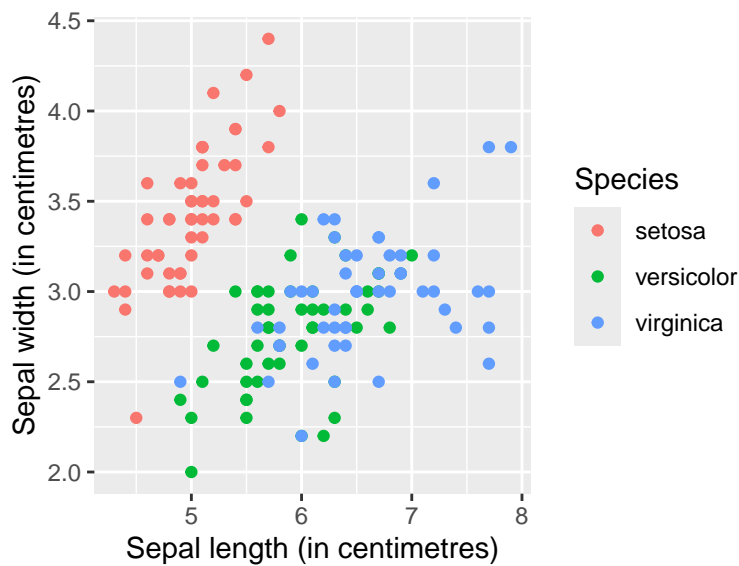
15 R ggplot

Figure 1: Relationship between sepal width and sepal length by species.

16 R chunk code

```
#| label: fig-scatterplot1
#| fig-cap: Relationship between sepal width and sepal length by species.
#| fig-width: 4
#| fig-height: 3
#| fig-align: center
#| message: false


ggplot(iris_subset, aes(x = Sepal.Length, y = Sepal.Width, color = Species)) +
  geom_point() +
  labs(x = "Sepal length (in centimetres)", y = "Sepal width (in centimetres)", color = "Spe
```

Here, we set the chunk label to include the `fig-` prefix so we can cross-reference using the
@ prefix. For example `@fig-scatterplot1` will create a hyperlink to Figure 1. Then we
added a figure caption using the `fig-cap` option. For size and positioning of the figure we
can include:

- `fig.width`: an integer value denoting the width of the figure;
- `fig.height`: an integer value denoting the height of the figure;
- `fig.align`: the alignment of the figure within the body of the document

> **Question**
>
> What other argument would you need to include if you wish to show the chunk plot
> output only?
>
> - (A) eval: false

- (B) echo: true

- (C) echo: false

- (D) eval: true

Task 6

Create a new chunk in your exploratory analysis section that displays a scatter plot showing the relationship between sepal width and length by species. Re-size and align the plot accordingly, add a caption and a cross-reference label. Finally write some comments in the main body while cross-referencing the plot you created.
see solution
Your new chunk should look something like this:

```
#| echo: false
#| fig-cap: Relationship between sepal width and sepal length by species.
#| label: fig-scatter
#| fig-align: center
#| fig-width: 4.5
#| fig-height: 3.5

ggplot(iris_subset, aes(x = Sepal.Length,
                y = Sepal.Width,
                color = Species)) +
  geom_point() +
  labs(x = "Sepal length (in centimetres)",
       y = "Sepal width (in centimetres)",
       color = "Species")
```

Then, `@fig-scatter` should be used in the text description to refer to this plot.

i Note

You can read more about authoring Quarto figures here.

17 Mathematics

Mathematical and statistical equations can be presented nicely within a Quarto document. For example, the following equation referring to a linear regression model:

$$y_i = \alpha + \beta x_i + \epsilon_i, \quad \epsilon_i \sim N(0, \sigma^2),$$

can be specified using the following syntax:

```
$$
y_i = \alpha + \beta x_i + \epsilon_i, ~~~~~ \``epsilon_i``\sim N(0, \sigma^2), $$
```

That is, we use:

- $$ signs to produce mathematics which is centred, and a single $ to include mathematics within a sentence or paragraph; in visual mode you can go to Insert

- `_` and `^` are used for subscripts and superscripts, respectively;
- Greek letters are obtained using \ and the letters name, e.g., `\alpha` gives α.
- tildes (~) are used to put spacing between notation.

If you have a categorical variable then you would write it using an indicator function. For example, if we have gender as a categorical variable, where females are the baseline category, then we could write our model as follows:

$$y_i = \alpha + \beta_{\mathrm{Male}} \cdot \mathbb{I}_{\mathrm{Male}}(x),$$

```
$$
y_i = \alpha + \beta_{\mathrm{Male}} \cdot \mathbb{I}_{\mathrm{Male}}(x),
$$
```

where $\mathbb{I}_{\mathrm{Male}}(x)$ (`$\mathbb{I}_{\mathrm{Male}}(x)$`) is an indicator function such that

$$\mathbb{I}_{\mathrm{Male}}(x) = \begin{cases} 1 \text{ if the gender of the } x\text{th observation is Male}, \\ 0 \text{ otherwise} \end{cases}$$

We can use the `\begin{cases}...\end{cases}` functions from the `amsmath` LaTeX package (available already in Quarto) to replicate this expression:

```
$$
\mathbb{I}_{\mathrm{Male}}(x) =
\begin{cases}
1 ~ \text{if the gender of the }x\text{th observation is Male},\\
0 ~ \text{otherwise}
\end{cases}
$$
```

Note that we used `\text()` to bypass any text format embedded within a mathematical expression. If you want to display an arrange of equations you could use the aligned environment:

```
$$
\begin{aligned}
y_i &\sim \mathrm{Normal}(\mu_i,\sigma^2) \\
\mu_i &= \beta_0 + \beta_1 x_i
\end{aligned}
$$
```

When rendered, this will produce:

$$\begin{aligned} y_i &\sim \mathrm{Normal}(\mu_i, \sigma^2) \\ \mu_i &= \beta_0 + \beta_1 x_i \end{aligned}$$

here, & aligns columns (like tab stops), while \\ starts a new row (like a line break). Display equations can be cross referenced by clicking on the equation options ⋯ (visual mode) and the entering #eq-label (eq is the equation suffix and this cannot be changed.)

Task 7

Try to write the LaTeX code for the correlation coefficient between two random variables x and y:

$$\rho(x,y) = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}}$$

Take hint

You can use the `dfrac{}{}` command to write a fraction in display mode. The `\bar{}` can be used to place a bar over a character, and `\sqrt{}` to draw a square root symbol. See here for a detailed LaTeX Math Symbols cheat sheet.

See Solution

```
$$
\rho (x,y) = \dfrac{\sum_{i=1}^n (x_i -\bar{x})(y_i-\bar{y})}{\sqrt{\sum_{i=1}^n (x_i-\bar
$$
```

Task 8

In the formal analysis section, underneath the Statistical model sub-section, write the expression of a saturated model that explores the relationship sepal width and sepal length for each iris species (i.e. separate regression lines for each species - we will cover this on the next few weeks). The equation you should be written as:

$$y_i = \alpha + \beta_{\text{length}}\cdot\text{length} + \beta_{\text{species}}\cdot\mathbb{I}_{\text{species}}(x) + \beta_{\text{length, species}}\cdot\text{length}\cdot\mathbb{I}_{\text{species}}(x) + \epsilon_i$$

Add an appropriate cross referable label for this equation and then refer to in in the text.

see solution

The display math for the full model would look something like this:

```
$$
y_{i} = \alpha + \beta_{\mbox{length}} \cdot \mbox{length} + \beta_{\mbox{species}} \cdot
$$
```

Or if you want to use a slightly more complex notation:

```
$$
\begin{aligned}
y_{i} &= \alpha + \beta_1 \cdot x_{1i} + \beta_2 \cdot x_{2i} + \beta_3 \cdot x_{1i} \cdot
&= \alpha + \beta_{\mbox{length}} \cdot \mbox{length} + \beta_{\mbox{species}} \cdot \math
\end{aligned}
$$
```

When compile it should appeared as:

-

$$y_i = \alpha + \beta_1 \cdot x_{1i} + \beta_2 \cdot x_{2i} + \beta_3 \cdot x_{1i} \cdot x_{2i} + \epsilon_i, \quad \epsilon_i \sim N(0, \sigma^2), \quad i = 1, \ldots, 150$$
$$= \alpha + \beta_{\text{length}} \cdot \text{length} + \beta_{\text{species}} \cdot \mathbb{I}_{\text{species}}(x) + \beta_{\text{length, species}} \cdot \text{length} \cdot \mathbb{I}_{\text{species}}(x) + \epsilon_i,$$

To add the description of each parameter you can click on the bullet list icon ▤ and use the appropiate math symbol, e.g.

- α is the intercept of the regression line for the baseline species (setosa);

which gets compiled as:

- α is the intercept of the regression line for the baseline species (setosa);

Lastly, make sure equation has a cross reference label such as `#eq-inter_model` (see example report)

> **ℹ Note**
>
> For additional tricks inserting mathematics into documents see here. For a complete tutorial on using LaTeX please see here.

18 Rendering

You can render a Quarto document or a presentation (see next section) by clicking the Render button ➡ .

> **ℹ Note**
>
> Note that if you are working on a Quarto document you can also compile a PDF version of it by clicking on ➡ `Render`   📄`Render PDF`. However, if you are working on a Quarto presentation while using `revealjs` format then you need to render the presentation first in html and then export it as pdf by following these instructions.

When you render a Quarto document, the R code in you chunks is executed by the knitr R-package and a new markdown (.md) document is then created. This file includes all of your code and its output. pandoc then processes this markdown to create the finished format. *The Render button encapsulates these actions and executes them in the right order for you.*

Note that HTML rendered documents will typically have a number of external dependencies (e.g. images, CSS style sheets, JavaScript, etc.) which are contained in an external `_files` directory alongside your document. If you share your HTML rendered file without the external files, it will lose all of its previous formatting and style. Alternatively, you can create a self-contained HTML document using the the `embed-resources` option in quarto as follows:

```
---
title: "Data Analysis: Example Report"
format:
  html:
    embed-resources: true
---
```

This will create standalone HTML file with no external dependencies that you can share with others (note that indentation here is important as you are calling arguments within the format output) .

> ❗ Important
>
> For your upcoming class test you want to make sure this option is set to `true` so your work can be displayed properly by a browser.

Task 9

Fit the full model describing the relationship between Sepal length and Sepal Width for each the iris species and then compared this to a parallel lines model (i.e., `lm(Sepal.Width ~ Sepal.Length + Species, data = iris_subset)`). Produce a table that compares these two models, add appropriate cross-referencable labels and captions for the table.

take hint

You can use the `tab_model()` function to compare the two nested models and display the results in one single output:

```
full.model <- lm(Sepal.Width ~ Sepal.Length * Species, data = iris_subset)
par.model <- lm(Sepal.Width ~ Sepal.Length + Species, data = iris_subset)
tab_model(full.model,par.model,
          collapse.ci = T,
          show.aic = T,
          show.obs = F,
          dv.labels = c("Interaction model","Pararell lines model"))
```

See solution

R chunk

```
#| echo: false
#| message: false
#| warning: false
#| label: tbl-regtable
#| tbl-cap: Estimates of the regression model coefficient with 95% confidence intervals

int.model <- lm(Sepal.Width ~ Sepal.Length * Species, data = iris_subset)
par.model <- lm(Sepal.Width ~ Sepal.Length + Species, data = iris_subset)
tab_model(int.model,par.model,
          show.aic = T,
          show.obs = F,
          collapse.ci = T,
          dv.labels = c("Interaction model","Pararell lines model"))
```

> **Task 10**
>
> Check your results and your final model assumption. You can use the `check_model` function from the `performance` package we loaded before to conduct a residual analysis. Finalise with some conclusions (you can use the Example report as reference).
>
> > **i** Note
> >
> > Don worry if you are not familiar with some of these regression modelling topics. We will cover these in the next couple of sessions.

19 More about Quarto

Congratulations! You have reached the end of today's session. If you want to learn more about quarto please visit https://quarto.org/. The website contains a wide variety of topics, tutorials, and practical examples to deepen your understanding.

Also, check out the additional material provided on creating eye-catching presentations with Quarto. This will help you to design professional slides with dynamic visuals, streamline your workflow by integrating code, text, and outputs seamlessly. Impress your audience with polished, publication-quality formats!