# Data Analysis
# Week 4: R Markdown

## 1 Introduction

A vital part of data analysis is communicating the results of an analysis. This usually takes the form or a report which summarizes the data analysis process and relevant findings. There is a helpful guide to writing statistical reports following a statistical analysis on Moodle here and you are expected to familiarize yourself with this resource, especially the sections on "Structure" and "Presentation".

This week's interactive tutorial will introduce you to a very efficient way of producing statistical reports and will show you how to produce reports for the two class tests and the group project in Data Analysis. Namely, we will take what we have learned in previous weeks and produce a report using **R Markdown**. The package `rmarkdown` allows reports to be created within R, thus allowing for R code and output to be easily embedded within a report. Hence, all of the R code and plots obtained from an analysis are contained within a single file.

Under Week 4 on Moodle there is an Example Report produced using R Markdown relating to fitting a regression model using one numerical explanatory variable that was done in Week 3. The corresponding R Markdown file (`.Rmd`) can also be downloaded and opened in R to see how the document was produced (press the `Knit` button to create the PDF version). It is advised to have this document open within RStudio while working through the remaining sections in order to see examples put into practice.

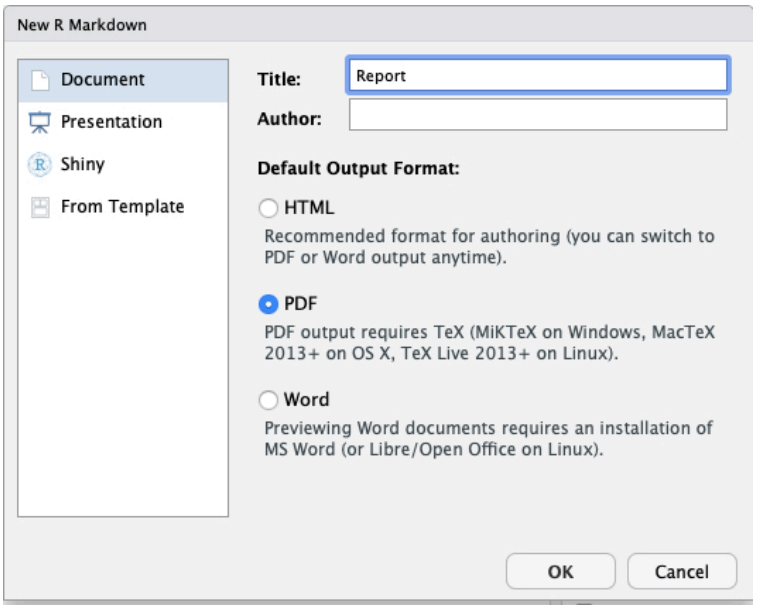
The following sections will take you through the different steps required to produce the Example Report on Moodle.

### 1.1 Creating a new R Markdown document from scratch

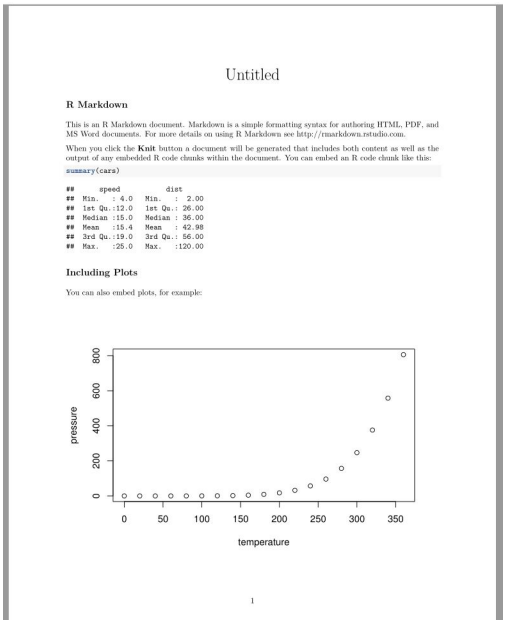
If you want to create a new R Markdown document from scratch within RStudio you can go to:

`File -> New File -> R Markdown...`

This will open the following window within RStudio:



From here select **Document** and the **Default Output Format** as **PDF**. Name your document `Week4DA` and select OK. This will open within RStudio a `.Rmd` file containing simple instructions on how to do some basic stuff using R Markdown. To see what the PDF of the default document looks like click on the `Knit` button at the top of the document window. A screen will appear showing the newly created `.pdf` document, the first page of which looks like this:



Untitled

**R Markdown**

This is an R Markdown document. Markdown is a simple formatting syntax for authoring HTML, PDF, and MS Word documents. For more details on using R Markdown see http://rmarkdown.rstudio.com.

When you click the **Knit** button a document will be generated that includes both content as well as the output of any embedded R code chunks within the document. You can embed an R code chunk like this:

```
summary(cars)
```

```
##      speed           dist       
##  Min.   : 4.0   Min.   :  2.00  
##  1st Qu.:12.0   1st Qu.: 26.00  
##  Median :15.0   Median : 36.00  
##  Mean   :15.4   Mean   : 42.98  
##  3rd Qu.:19.0   3rd Qu.: 56.00  
##  Max.   :25.0   Max.   :120.00
```

**Including Plots**

You can also embed plots, for example:



1

Thes tasks in this weeks tutorial require you to modify this default document by coping over the code in the following sections and compiling and viewing the `.pdf` file each time.

---

**Further information**: Additional information on getting started with Markdown can be found here.

# 2 Title

The title of the document can be found at the top of the `.Rmd` file within its preamble, which is shown below. When titling your document, ensure the title is within inverted commas.

```
---
title: "Example Report"
author: "Annon."
output:
  pdf_document:
          latex_engine: pdflatex
          number_sections: yes
fig_caption: yes
---
```

---

**Task** Give your document a suitable title and your name by copying and modifying the above code into `Week4DA.Rmd`. `Knit` the `.Rmd` file and notice what is produced in the `Week4DA.pdf` file.

# 3 Sections

Sections within an R Markdown document are created using `#`. For example, `# Introduction` will create a section titled **Introduction** with automatic numbering, i.e.

### 1 Introduction

**Note**: the section is numbered as we have set `number_sections: yes` within the preamble of the document. If you do not wish to have numbered sections, then set `number_sections: no` in the preamble. I would recommend using numbered sections as it makes it easier to refer to them within the text.

Each section can be assigned labels so that they can be referred to within the text. 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 name chosen for this particular section. It is a good idea to label your sections appropriately so that it is easy to refer to them later. The section can now be referred to within the text of the document using the `\ref` command. That is

```
Section \ref{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.

## 3.1 Subsections

Subsections can be added to a document in a similar fashion using `##` such that `## Subsection {#sec:sub}` will create a subsection with the label `sec:sub` and title **Subsection**:

### 1.1 Subsection

---

**Task** Create some suitable subsections in your document by copying and modifying the above code into `Week4DA.Rmd`. `Knit` the `.Rmd` file and notice what is produced in the `Week4DA.pdf` file.

# 4 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.

Using R code within an R Markdown document is done within the following environment:

```
```{r label, echo=FALSE, eval=TRUE, warning=FALSE, message=FALSE}

```
```

where each code chunk is given its own `label`. The additional arguments are:

- **echo**: include the R code within the code chunk in the document (TRUE/FALSE, default=TRUE);
- **eval**: evaluate the R code within the code chunk (TRUE/FALSE, default=TRUE);
- **warning**: suppress warnings from R (TRUE/FALSE, default=TRUE); and

- **message**: suppress messages from R (TRUE/FALSE, default=TRUE).

Often the first use of a code chunk in a R Markdown file is to load packages so that functions and datasets are available for analysis. For example, the following code chunk will load these packages:

- `ggplot2` - to access the `tidyverse` visualization functions
- `dplyr` - to access the `tidyverse` data manipulation functions
- `skimr` - to access the `tidyverse` data summary functions
- `moderndive` - to access the student feedback data `evals` that we looked at in Week 3
- `datasets` - to access the `iris` data frame

````
```{r loadpackages, echo=FALSE, eval=TRUE, warning=FALSE, message=FALSE}
library(ggplot2)
library(dplyr)
library(moderndive)
library(gapminder)
library(skimr)
library(mvtnorm)
library(gridExtra)
library(kableExtra)
library(tidyr)
```
````

Then, let's say we wanted to select the `score` and `bty_avg` variables from the `evals` data set to be used later, we can do that using the following code chunk:

````
```{r evals, echo = FALSE, eval = TRUE, warning = FALSE}
evals.scores <- evals %>%
  select(score, bty_avg)
```
````

This will evaluate the R code and store the subsetted data set as the object `evals.scores` so that it can be used later. If you want to embed the code within the Markdown document then you would simply set `echo = TRUE` and it will include the R code as follows:

```
evals.scores <- evals %>%
  dplyr::select(score, bty_avg)
```

**Note:** It is usually optional whether the package is specified in front of the function, as in `dplyr::select` here (compare to the previous code chunk above were `select` appears without the package name preceding it - both chunks of code do exactly the same thing), but if you have problems running a particular function, try including the `package name` and `::` before the function. Also, you can always check what functions or datasets are in a particular package using `help(package="packagename")` or refer to the cheat sheets.

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 **Figures** section later.

---

**Task** Create some code chunks in your document by copying and modifying the above code into `Week4DA.Rmd`. `Knit` the `.Rmd` file and notice what is produced in the `Week4DA.pdf` file.

### 4.1 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 `evals.scores` then we can enclose `nrow(evals.scores)` within `` `r ` `` to obtain the number of observations, rather than hard wiring "463" 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 the number of observations could be stored as an R object, e.g. using `` `r n<-nrow(evals.scores)` `` , and referred to whenever necessary within the text using `` `r n` ``.

---

**Further information**: Additional details on all of the code chunk options can be found here and here.

## 5 Tables of data

There are several ways to produce tables in Markdown. Here, a couple of different approaches will be presented. The first approach uses the `kable` function from the `knitr` package (see also the `kableExtra` package) and essentially puts a wrapper around the table produced in R in order to make it more visually appealing within the R Markdown document.

Let's say we wanted to create a table of the first 5 rows of the `iris` data from the `datasets` library. We can create the table using the `kable` function as follows:

```
```{r table}
kable(head(iris, n = 5), caption = '\\label{tab:iris} The first 5 rows of the iris data.')
```
```

This produces the following table in the `.pdf` document:

Table 1: The first 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 |

Notice that within the `caption` argument of the `kable` function there is `\\label{tab:iris}`. This is how you label tables in order to refer to the within the text. For example,

```
Table \ref{tab:iris} displays the first 5 rows of the iris data...
```

will produce

```
Table 1 displays the first 5 rows of the iris data...
```

where, like for sections, the "1" will be a hyperlink directed to the table.

See also the Example Report Markdown file for an example of using the `kable_styling` function for changing the size and positioning of a table. For example, its often useful to include `kable_styling(font_size = 10, latex_options = 'HOLD_position')` as the final term in the "pipe" to control the size of the font used to construct the table and to ensure the table appears at that location in the document (otherwise it may be placed at the bottom or top of the page).

---

**Further information**: Additional details on using `kable` to produce tables can be found here and here.

## 5.1 Tables of summaries

Often we need to report summary statistics for different subsets in a set of data. For instance we can construct a table of summaries for the sepal length of the different species in the `iris` data using:

```
iris %>%
  group_by(Species) %>%
  summarise(n=n(),Mean=round(mean(Sepal.Length),digits=1),
            St.Dev=round(sd(Sepal.Length),digits=1),
            Min=min(Sepal.Length), Q1 = quantile(Sepal.Length,0.25),
            Median=median(Sepal.Length),
            Q3 = quantile(Sepal.Length,0.75), Max=max(Sepal.Length)) %>%
  kable(caption = '\\label{tab:summaries} Summary statistics
        on the sepal length by species of irises.') %>%
  kable_styling(font_size = 10, latex_options = "hold_position")
```

Table 1: Summary statistics on the sepal length by species of irises.

| Species | n | Mean | St.Dev | Min | Q1 | Median | Q3 | Max |
|---|---|---|---|---|---|---|---|---|
| setosa | 50 | 5.0 | 0.4 | 4.3 | 4.800 | 5.0 | 5.2 | 5.8 |
| versicolor | 50 | 5.9 | 0.5 | 4.9 | 5.600 | 5.9 | 6.3 | 7.0 |
| virginica | 50 | 6.6 | 0.6 | 4.9 | 6.225 | 6.5 | 6.9 | 7.9 |

---

**Task** Include a table of summaries of `Petal.Width` for the different iris species in your document by copying and modifying the above code into `Week4DA.Rmd`. `Knit` the `.Rmd` file and notice what is produced in the `Week4DA.pdf` file.

## 5.2 Tables of model estimates

Often we also need to report the results of fitting a model to our data. For instance if we modeled the sepal length on the different species in the `iris` data by:

$$\widehat{\text{Sepal.Length}}(x) = \widehat{\alpha} + \widehat{\beta}_{\text{versicolor}} \cdot \mathbb{I}_{\text{versicolor}}(x) + \widehat{\beta}_{\text{virginica}} \cdot \mathbb{I}_{\text{virginica}}(x)$$

Then we could report the estimated values of $\widehat{\alpha}$, $\widehat{\beta}_{\text{versicolor}}$ and $\widehat{\beta}_{\text{virginica}}$ by constructing the following table:

````
```{r fittedmodel}
model <- lm(Sepal.Length ~ Species, data = iris)
get_regression_table(model) %>%
  dplyr::select(term,estimate) %>%
  #Note that it seems necessary to include "dplyr::" here!!
  kable(caption = '\\label{tab:reg} Estimates of the parameters from the fitted linear regression model
  kable_styling(latex_options = 'HOLD_position')
```
````

```
model <- lm(Sepal.Length ~ Species, data = iris)
get_regression_table(model) %>%
  dplyr::select(term,estimate) %>%  #Note that it seems necessary to include dplyr:: here!!
  kable(caption = '\\label{tab:reg} Estimates of the parameters from the fitted
        linear regression model.') %>%
  kable_styling(latex_options = 'HOLD_position')
```

Table 2: Estimates of the parameters from the fitted linear regression model.

| term | estimate |
|---|---|
| intercept | 5.006 |
| Speciesversicolor | 0.930 |
| Speciesvirginica | 1.582 |

### 5.3 Tables 'by hand'

Tables can also be produced "by hand"" in Markdown. For example, the table above corresponding to the first 5 rows of the `iris` data can be produced by hand by typing the following text (without any other text) into a `.Rmd` file:

```
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

Table: The fist 5 rows of the iris data.
```

This produces 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 |

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 `:---:`. A limitation of creating tables by hand is that you cannot give them a label to refer to them later within the text (which will also need to be done by hand).

---

**Further information**: Additional details on creating tables by hand can be found here.

# 6 Figures

Including plots within an R Markdown document is straightforward. The R code for the plot is simply included within a code chunk including additional arguments for plot size and positioning. For example, this code chunk in the `.Rmd` file will produce the scatterplot of teaching and beauty scores below:

```
```{r scatplot, echo = FALSE, eval = TRUE, out.width = '70%', fig.align = "center",
warning = FALSE, fig.cap = "\\label{fig:scat} Relationship between teaching and
beauty scores. The best-fitting line has been superimposed."}
ggplot(evals.scores, aes(x = bty_avg, y = score)) +
  geom_point() +
  labs(x = "Beauty Score", y = "Teaching Score") +
  geom_smooth(method = "lm", se = FALSE)
```
```

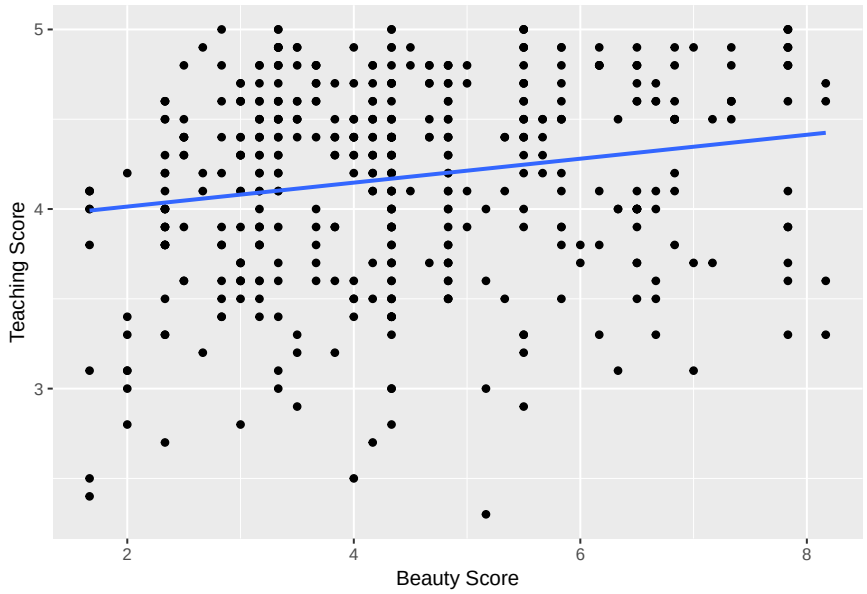



Figure 1: Relationship between teaching and beauty scores. The best-fitting line has been superimposed.

(**NB:** The "Figure 1:" label doesn't appear here before the caption, but it does in the `.pdf` file - see `ExampleReport.pdf` generated by `ExampleReport.Rmd`)

Here, we have `echo = FALSE` and `eval = TRUE` as we only want to put the plot into the document and not the accompanying R code. Labelling is done the same way as with tables, that is, the label `\\label{fig:scat}` is placed within the figure caption, which is given by the argument `fig.cap`. For size and positioning of the figure we can include:

- `out.width`: a percentage of the actual size of the produced plot;
- `fig.width`: an integer value denoting the width of the figure;
- `fig.align`: the alignment of the figure within the body of the document; and
- `fig.pos`: can be used to force the positioning of the figure within the document.

For examples of using each of the above arguments see the Example Report Markdown file `ExampleReport.Rmd`.

**Further information**: Additional notes on figures can be found here.

---

**Task** Include a plot showing the relationship between the teaching score and the age of the professors in your document by copying and modifying the above code into `Week4DA.Rmd`. `Knit` the `.Rmd` file and notice what is produced in the `Week4DA.pdf` file.

# 7 Mathematics

Mathematics and statistical equations can be presented nicely within an R Markdown document using LaTeX notation. For example, the following equation referring to a linear regression model:

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

is produced using the following code:

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

That is, we use:

- `$$` signs to produce mathematics which is centred on a new line, and a single `$` to include mathematics within a sentence or paragraph;
- `_` and `^` are used for subscripts and superscripts, respectively;
- Greek letters are obtained using `\` and the letters name, i.e. `\alpha` gives $\alpha$; and
- tildes (`~`) are used to put spacing between notation.

For additional tricks inserting mathematics into documents see the Example Report Markdown file `ExampleReport.Rmd`.

**Further information**: Additional details on including mathematics into Markdown documents can be found here and here.

---

**Task** Include an equation describing the relationship between the teaching score and the age of the professors in your document by copying and modifying the above code into `Week4DA.Rmd`. `Knit` the `.Rmd` file and notice what is produced in the `Week4DA.pdf` file.

# 8 Further Tasks

The following tasks are here to help you prepare for the class test in Week 5. For the class test you will be given a data set to analyse and produce a report using R Markdown given a basic template. Using the *Basic Template* from Moodle, write a report based on the following data sets.

1. Return to Week 3 and the section on simple linear regression with one categorical explanatory variable. Produce a report based on life expectancy across various countries around the world from the `gapminder` data set.
2. There are several categorical variables in the `evals` data set on the student evaluations of a large sample of professors, for example the `ethnicity` of the professors ("minority" or "not minority") and the `rank` of the professors (i.e. low-status, mid-status or high-status). Choose one of these categorical variables and prepare a report describing the differences between the different categories of the average evaluation `score` of the professors.