Using R as a Research Tool.
Part 2: R Markdown and basic statistics.

Dr Susan Johnston: Susan.Johnston@ed.ac.uk

Demonstrators: Gergana Daskalova, John Godlee.
Hat-Tips to Kyle Dexter, The Coding Club and R4all.

November 23, 2017

1 Introduction

This practical will follow on from the previous practical in data manipulation and visualisation, exploring how to write reports in **R** Markdown and how to conduct simple statistical tests in **R**. By the end of the practical, you should be able to:

- Write, embed and render code and results into an HTML document.
- Carry out basic statistics and visualisations, including:
  - Linear regression with `lm()`
  - Chi-squared test with `chisq.test()`
  - 2-sample t-test with `t.test()`

2 Writing reports with R Markdown.

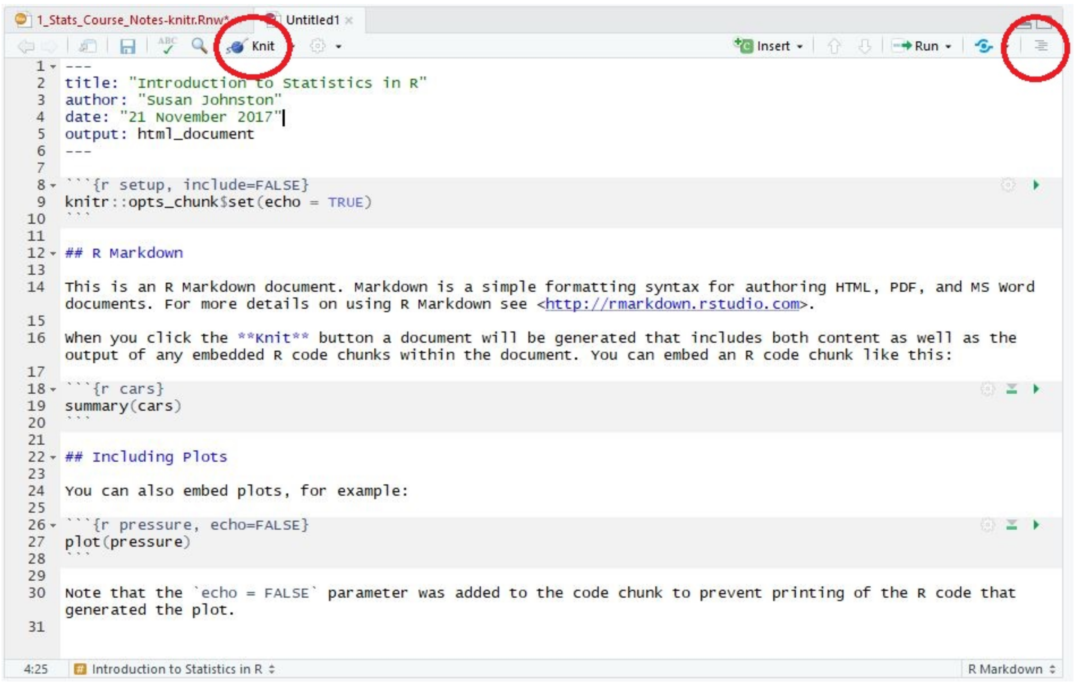
R Markdown is a tool for writing, reproducible reports in **R**. It can be used to produce documents with embedded code and figures in HTML, Word and PDF format, and can also be used to create webpages and slideshows. The current version of the **R** Markdown Cheatsheet from RStudio has been attached to these course notes.

2.1 Creating an R Markdown Document.

Open RStudio and create a new markdown document by going to **File > New File > R Markdown...**. In the window, name your document, select **HTML** and click **OK**. RStudio should automatically create a template as in Figure 1 (if not - it is saved in the file `R_Markdown_Template.Rmd`).

To render the document, click the button that says **Knit** (you may have to save it first). As you can see, it produces a formatted HTML document with embedded figures.

Another thing to note is the **Show document outline** in the top right corner of the window, which shows the document structure.


1_Stats_Course_Notes-knitr.Rnw*
Untitled1
Knit
Insert
Run
4:25
Introduction to Statistics in R
R Markdown

~~~
---
title: "Introduction to Statistics in R"
author: "Susan Johnston"
date: "21 November 2017"
output: html_document
---

```{r setup, include=FALSE}
knitr::opts_chunk$set(echo = TRUE)
```

## 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:

```{r cars}
summary(cars)
```

## Including Plots

You can also embed plots, for example:

```{r pressure, echo=FALSE}
plot(pressure)
```

Note that the `echo = FALSE` parameter was added to the code chunk to prevent printing of the R code that
generated the plot.
~~~

Figure 1: R Markdown Template.

2.2 Formatting an R Markdown Document.

It is possible to carry out basic formatting, tables, headers, as well as adding tables, lists and external figures. See Part 3 of the R Markdown CheatSheet from RStudio, which is attached to the back of this document.

Exercise 1.
1. Create a new R Markdown Script with the following header:

```
--
title: “Introduction to Statistics in R”
author: “Your Name”
date: “28 November 2017”
output:  html_document
--
```

Add some text to it and familiarise yourself with how to make *italic*, **bold** and superscript text. Add headers for “Introduction”, “Linear regression”, “Chi-squared test” and “Two-sample t-test“..

2.3 Embedding code in R Markdown.

Code can be embedded into the document in two ways:

Code Chunks:

~~~
```{r}
head(iris)
```
~~~

Code chunks can be named, e.g. ` ```{r iris} `

Inline code:

```
Two plus two equals `r 2 + 2`.
```

Which will print “Two plus two equals 4.”.

There are also a number of display options for each chunk in section 5 of the cheat sheet. For example,

~~~
```{r echo = F, results = "hide", fig.width = 4, fig.height = 3}
head(iris)

library(ggplot2)
ggplot(iris, aes(x = Sepal.Length, y = Sepal.Width)) + geom_point()
```
~~~

will show the code output and the figure (dimensions specified with `fig.width` and `fig.height`), but not the code itself (`echo = F`) or the console output (i.e. `head(iris)`; `results = "hide"`).

2.4 Points to note.

- Rendering the document with **Knit**, that R is opening and creating a new working environment. Each code chunk that is run is saved in the workspace - therefore, libraries and new objects created in a chunk will remain in the workspace for all subsequent chunks.
- It's good to put a chunk at the start loading the libraries and data.
- Code in **R** Markdown chunks can be run in the console as in the previous practical. The default option is to show the code output inline in the document. Some people like this, others don't: to switch this function on and off, go to **Global options > R Markdown > Show output inline for all Markdown documents** and select your preferred setup.
- **R** Markdown documents are not flexible to errors - the code must be error free, or it will not render.
- Want to learn more? More detailed information on this can be found at the Coding Club tutorial at `https://ourcodingclub.github.io/2016/11/24/rmarkdown-1.html`.

For the rest of the practical, it is expected that you create an R Markdown document that contains text, code, graphs and inline reporting of results.

3 Basic statistics.

Question > visualisation > statistic > render

4 χ^2 contingency table

A χ^2 contingency table analyses count data, and looks at the association between two or more categorical variables. In this example, we will examine the differences in the frequency of red and black ladybirds (*Adalia bipunctata*) in rural and industrial habitats. Our question is: are dark morphs more likely to reside in dark (industrial) backgrounds? The null hypothesis is that there is no association between ladybird colour morph and habitat type.

Load the data file `ladybirds.csv` into R using `read.csv()` and examine it using `glipse()` from the `dplyr` package.

```
library(dplyr)

ladybirds <- read.csv("data/ladybirds.csv", header = T)
glimpse(ladybirds)
```

```
Observations: 20
Variables: 4
$ Habitat      <fctr> Rural, Rural, Rural, Rural, Rural, Rural, Rural, Rura...
$ Site         <fctr> R1, R2, R3, R4, R5, R1, R2, R3, R4, R5, U1, U2, U3, U...
$ morph_colour <fctr> black, black, black, black, black, red, red, red, red...
$ number       <int> 10, 3, 4, 7, 6, 15, 18, 9, 12, 16, 32, 25, 25, 17, 16,...
```

There are multiple lines for each category, with the column number giving the count details. We ultimately want four numbers, corresponding to the 2×2 categories: red industrial, black industrial, red rural and black rural.

This can be done using the `dplyr` functions `group_by` amd `summarise()`.

```
> totals <- group_by(ladybirds, Habitat, morph_colour)
> totals <- summarise(totals, total.number = sum(number))
> totals

# A tibble: 4 x 3
# Groups:   Habitat [?]
     Habitat morph_colour total.number
      <fctr>       <fctr>        <int>
1 Industrial        black          115
2 Industrial          red           85
3      Rural        black           30
4      Rural          red           70
```

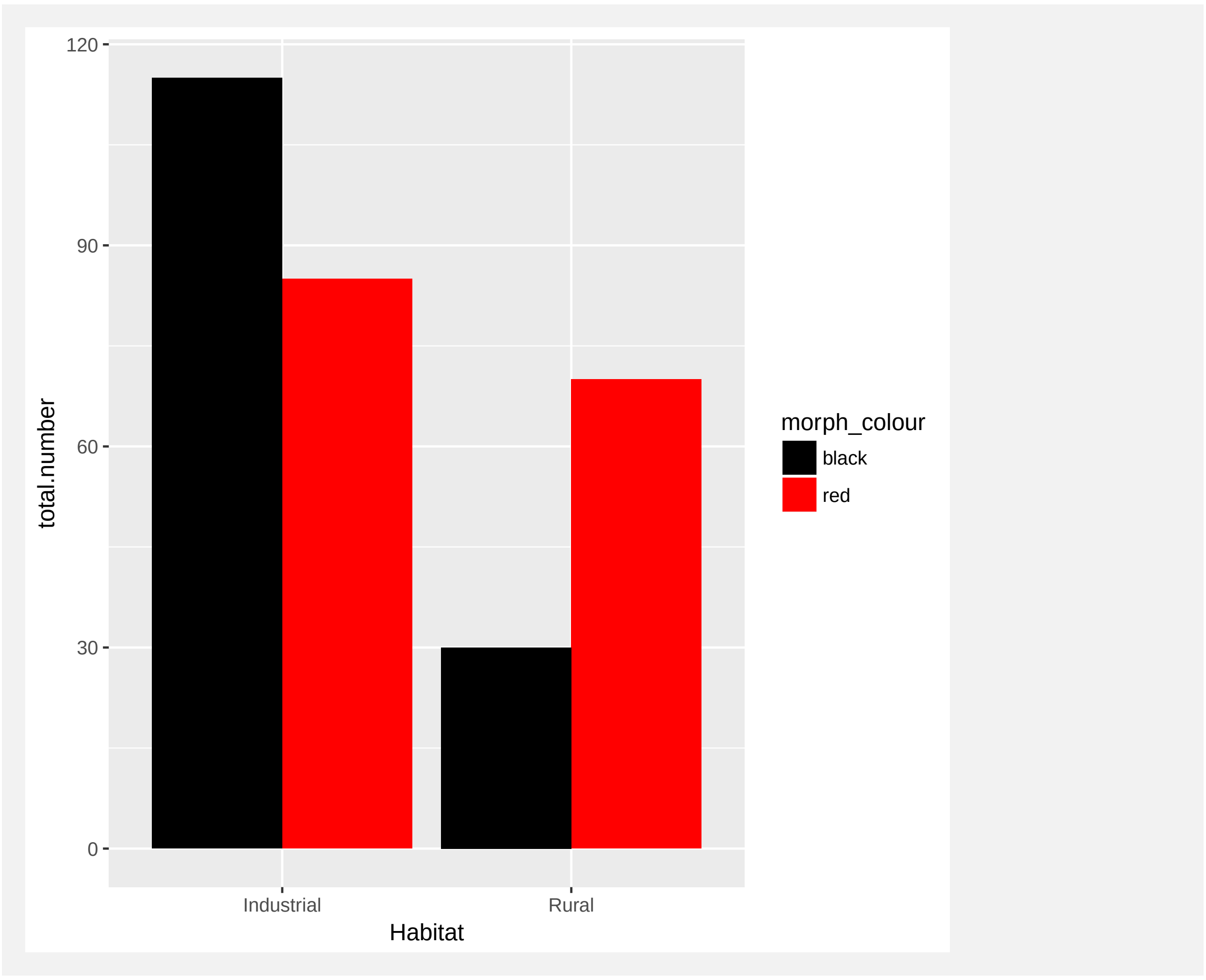
4.1 Plot the data

One visualisation for this type of data is a bar chart using `geom_col()` in `ggplot2`.

```
> library(ggplot2)
> ggplot(totals, aes(x = Habitat, y = total.number, fill = morph_colour)) +
+   geom_col()
```

There are two edits we can make to this to improve the visualisation: to add `geom_col(position = "dodge")` to place bars side by side, and also by changing the colours of the bars to black and red to match the colour of the morphs in real life (using `scale_fill_manual()`):

```
> ggplot(totals, aes(x = Habitat, y = total.number, fill = morph_colour)) +
+   geom_col(position = "dodge") +
+   scale_fill_manual(values = c(black = "black", red = "red"))
```



4.2 Run the test with `chisq.test`.