Introduction to R Markdown

Dr Brendan Palmer

17th Marh 2019

That first bit was a YAML header (optional). You can control “whole document” settings by tweaking the YAML header. With YAML, indentation is important, so just keep it simple first as you get up and running.

Oh, and this bit right here is just plain text as you would see in any text editor. This next bit is how we include our code within the document. It starts with three back-ticks, a pair of curly braces, “r” and then ends with three back ticks. The code lies within.

Example 1:

# This can be written just like any other R script  
  
# You can create new vectors  
x <- 1:100  
y <- summary(x)  
y

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 1.00 25.75 50.50 50.50 75.25 100.00

Example 2: In that first code chunk, we just did a simple operation. If we were to present this to a colleague as part of our presentation/analysis, we can remove the code and display only the output as follows……

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 1.00 25.75 50.50 50.50 75.25 100.00

Example 3: Of course, maybe we just want to report discrete values.

When we’re writing, little elements of R code can be incorporated into the document with back ticks.

The mean of a sequence of numbers from 1 to 100 is 50.5 or 50.5 depending on how you like writing the background code.

The benefit of all of this is that the numerical information in the document is coming directly from the data, thereby eliminating errors arising from copy/paste. Even more importantly, if a value is questioned, it can be readily checked, verified or corrected.

# Formatting

## Headers

That hash symbol (#) indicates the header levels. Two hash symbols together ## give us our second header level. Because it has a specific purpose, I had to escap it with the backslash (\).

# This is header level 1

## This is header level 2

### This is header level 3 and so on

## Text formatting

*This text is in italics.* *And so is this text.*

**This text is in bold.** **And so is this text.**

~~This text is rendered with strikethrough.~~

Code is surrounded by the back ticks which changes its appearence in the finished document. This can be useful in the finished document. Today for example we have used the gather() and separate() function from the tidyr package.

FYI, This is a paragraph.

Now I’m in paragraph 2. I’m still in paragraph 2 too! You’ll get what I mean once you look at the ‘knitted’ document

I’m in paragraph three now!

## Lists

### Bulleted list item

* Item 1
* Item 2
  + 2a
  + 2b

or

* Item
* Another item

or

* Item
* Another item

### Numbered list items

1. Item one
2. Item two
3. Item three

## Links

[b.palmer@ucc.ie](mailto:b.palmer@ucc.ie)

<http://tidyverse.org>

[Click me to go to tidyverse central!](http://tidyverse.org/)

[Can place links to locations on your own system](/Users/bpalmer/).

## Code chunks

These can be added by clicking on the green insert button or by typing….

As you can see if you click on the green button, there are a number of languages supported within R Markdown

The chunks can be given names, but two chunks can’t have the same name. This has three advantages. 1. You can more easliy navigate to specific code chunks 1. Graphics produced by the chunks will have useful names 1. You can set up networks of cached chunks to avoid re-running expensive computation on every run.

#### Chunk options

* eval = FALSE - prevents the code from being evaluated
* include = FALSE - runs the code but doesn’t show it in the final document
* echo = FALSE - prevents the code but not the results from appearing
* message = FALSE - prevents messages from appearing in the finished file
* results = "hide" - hides the printed output
* error = TRUE - causes the render to continue even if the code returns an error

For examples of chunks open ~project-structure-March-19/docs/code\_chunks.Rmd

## -- Attaching packages --------------------------------------------------- tidyverse 1.2.1 --

## v ggplot2 3.1.0 v purrr 0.2.5  
## v tibble 2.0.1 v dplyr 0.7.8  
## v tidyr 0.8.2 v stringr 1.3.1  
## v readr 1.3.1 v forcats 0.3.0

## -- Conflicts ------------------------------------------------------ tidyverse\_conflicts() --  
## x dplyr::filter() masks stats::filter()  
## x dplyr::lag() masks stats::lag()

## Tables

By default, R Markdown prints data tables as you’d see them in the console

mtcars[1:5, 1:10]

## mpg cyl disp hp drat wt qsec vs am gear  
## Mazda RX4 21.0 6 160 110 3.90 2.620 16.46 0 1 4  
## Mazda RX4 Wag 21.0 6 160 110 3.90 2.875 17.02 0 1 4  
## Datsun 710 22.8 4 108 93 3.85 2.320 18.61 1 1 4  
## Hornet 4 Drive 21.4 6 258 110 3.08 3.215 19.44 1 0 3  
## Hornet Sportabout 18.7 8 360 175 3.15 3.440 17.02 0 0 3

You can also get them displayed with additional formatting. The knitr package comes with the kable() function. It also has a digits argument to control rounding.

A nicer version of the same table

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | mpg | cyl | disp | hp | drat | wt | qsec | vs | am | gear | carb |
| Mazda RX4 | 21.0 | 6 | 160 | 110 | 3.9 | 2.6 | 16.5 | 0 | 1 | 4 | 4 |
| Mazda RX4 Wag | 21.0 | 6 | 160 | 110 | 3.9 | 2.9 | 17.0 | 0 | 1 | 4 | 4 |
| Datsun 710 | 22.8 | 4 | 108 | 93 | 3.8 | 2.3 | 18.6 | 1 | 1 | 4 | 1 |
| Hornet 4 Drive | 21.4 | 6 | 258 | 110 | 3.1 | 3.2 | 19.4 | 1 | 0 | 3 | 1 |
| Hornet Sportabout | 18.7 | 8 | 360 | 175 | 3.1 | 3.4 | 17.0 | 0 | 0 | 3 | 2 |

### Statistical outputs

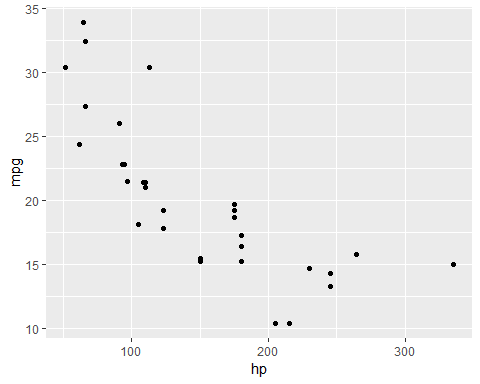
It’s also possible to format the outputs of your statistcial test for quick reporting.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| term | estimate | std.error | statistic | p.value |
| (Intercept) | 36.908 | 2.191 | 16.847 | 0.000 |
| hp | -0.019 | 0.015 | -1.275 | 0.213 |
| cyl | -2.265 | 0.576 | -3.933 | 0.000 |

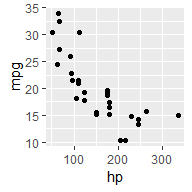
There is a bit of cleaning up required here but you get the idea.

## Figures

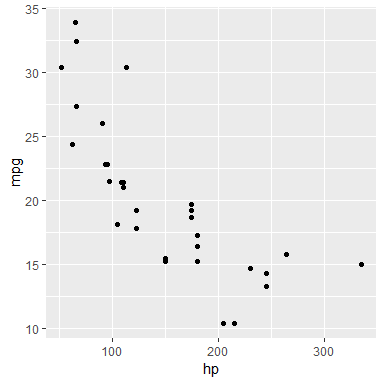
Figures will just be included using default settings.



*Fig. 1. Default*



*Fig. 2. Width = 2 by 2 in*



*Fig. 3 Width = 4 x 4 in*

Figure size can be set for the whole document within the YAML header, a global chunk option or within the individual chunk. Note that the default numbers is set to inches.

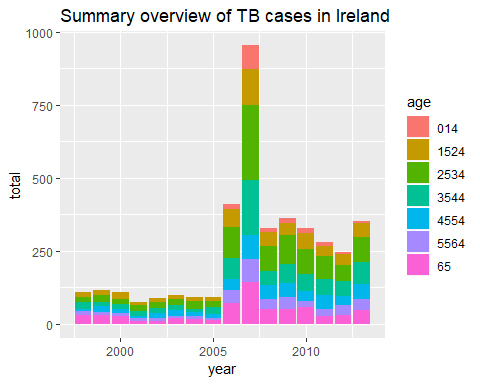
We can also generate the figures as we would in our R script, or call on figure object within the R script.

source("scripts/03\_pm\_clean\_data.R")

## Parsed with column specification:  
## cols(  
## .default = col\_double(),  
## country = col\_character(),  
## iso2 = col\_character(),  
## iso3 = col\_character()  
## )

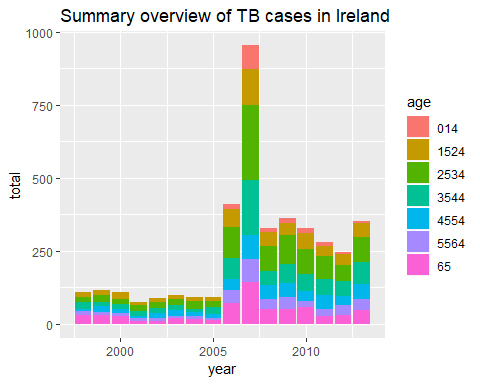
## See spec(...) for full column specifications.

# Plot A - early on in the project  
who\_ire <- who\_tb\_data %>%  
 filter(country == "Ireland") %>%  
   
# Basic barplot with data by age shown  
ggplot() +  
 geom\_bar(mapping = aes(x = year, y = total, fill = age),  
 stat = "identity") +  
 labs(title = "Summary overview of TB cases in Ireland")  
  
who\_ire

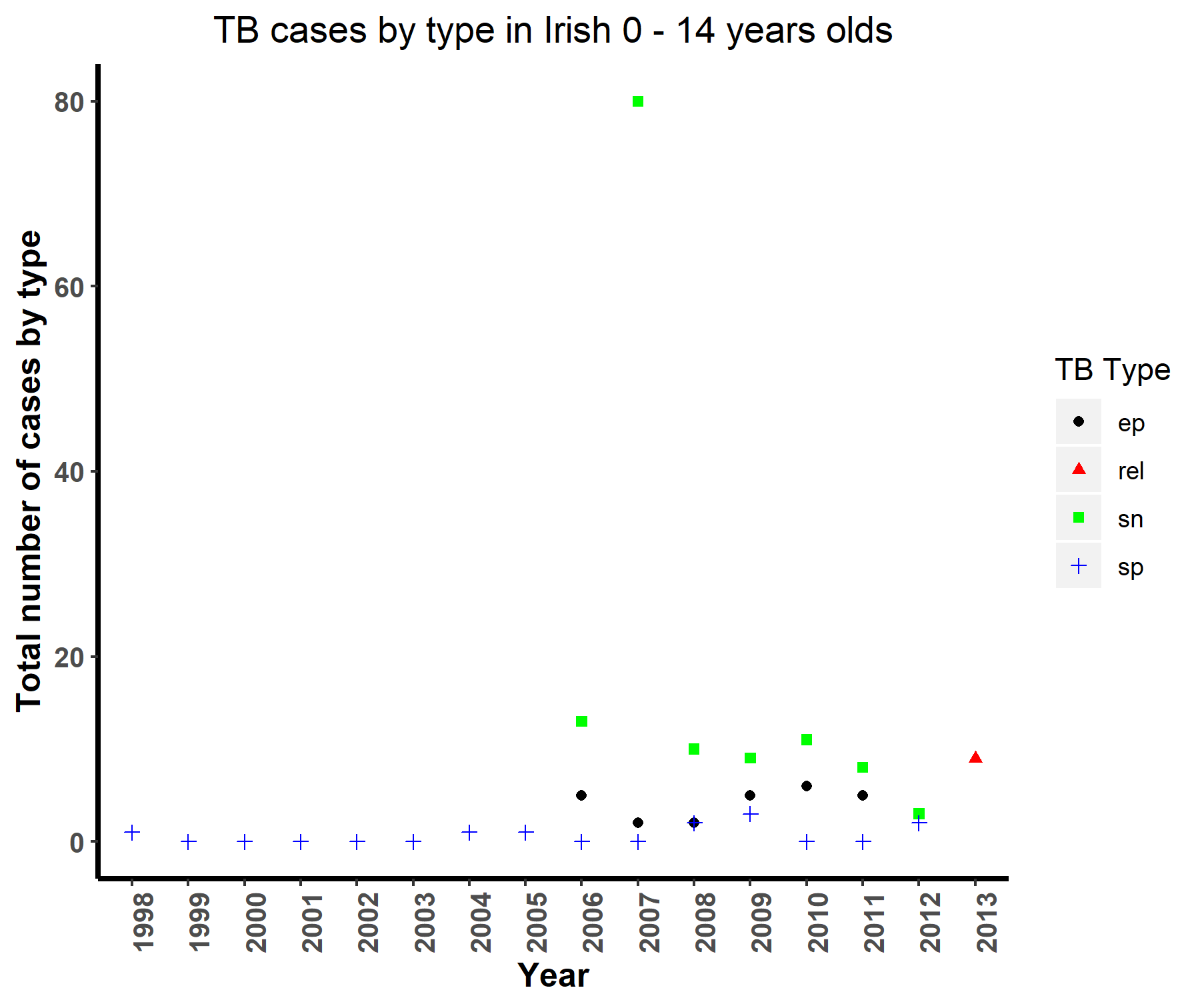


## Parsed with column specification:  
## cols(  
## .default = col\_double(),  
## country = col\_character(),  
## iso2 = col\_character(),  
## iso3 = col\_character()  
## )

## See spec(...) for full column specifications.



Personally, I prefer this second option as the document is less cluttered and changes are confined to the scripts, but you can also call on the image you stored in your plots folder.



Here’s one I made earlier.

## Caching

Normally each knit of a document starts from a clean slate. This great for reproducibility, but painful if some computations take a long time to run. You can avoid this by setting cached = TRUE. This will save the output of the chunk to a named file on disk. On susequent runs knitr will check to see if the code has changed, and if it hasn’t, will reuse the cached results.

## Bibliographies and citations

Pandoc can automatically generate citations and a bibliography in a number of styles.

This is just as an FYI. We won’t be able to cover it here.