Introduction to markdown

Olivia Boyd

26/04/2021

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.

We can run code chunks in markdown or output code into a report format. Either way, R Markdown documents provide a covenient way to ensure our analyses are reproducible and easy to share amongst colleagues.

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. We first use a dataframe called cars which is included in baseR for the purpose of exploring functions. This dataframe has two variables, speed and dist. Using this dataframe, we show that you can embed an R code chunk like this:

```
##      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
```

Where the above summary was ouput using the r code chunk summary(cars). We can also incorporate specific values into text based on analyses or present objects. For example, lets say we would like to report on the mean of speed or distance a car went from the cars data included in baseR. We can calculate the mean by including mean(cars$speed), mean = 15.4, or mean(cars$dist), mean = 42.98.

One thing to note, # does not represent comments in R markdowns. Instead, varying levels of # allow you to include headings and further subheadings. This is because the language used in Rmarkdown is slightly different from Rscript.

Headings 1

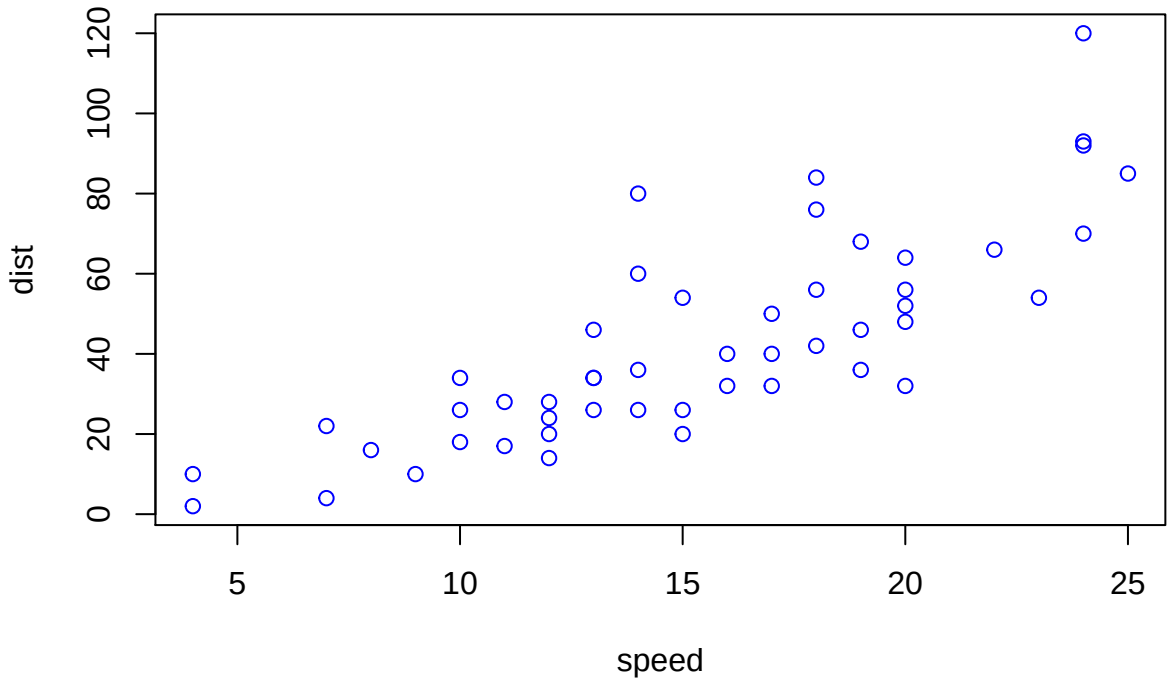
Headings 2

Headings 3

To type comments in markdown script, you can use: We will not see the previous three lines shown in the R markdown in the output pdf from using the Knit function.

Including Plots

You can also embed plots, for example:



Note that the **echo = FALSE** parameter was added to the code chunk to prevent printing of the R code that generated the plot. If we set **echo = TRUE**, then the R code chunk will also be included in the output, such as shown below.

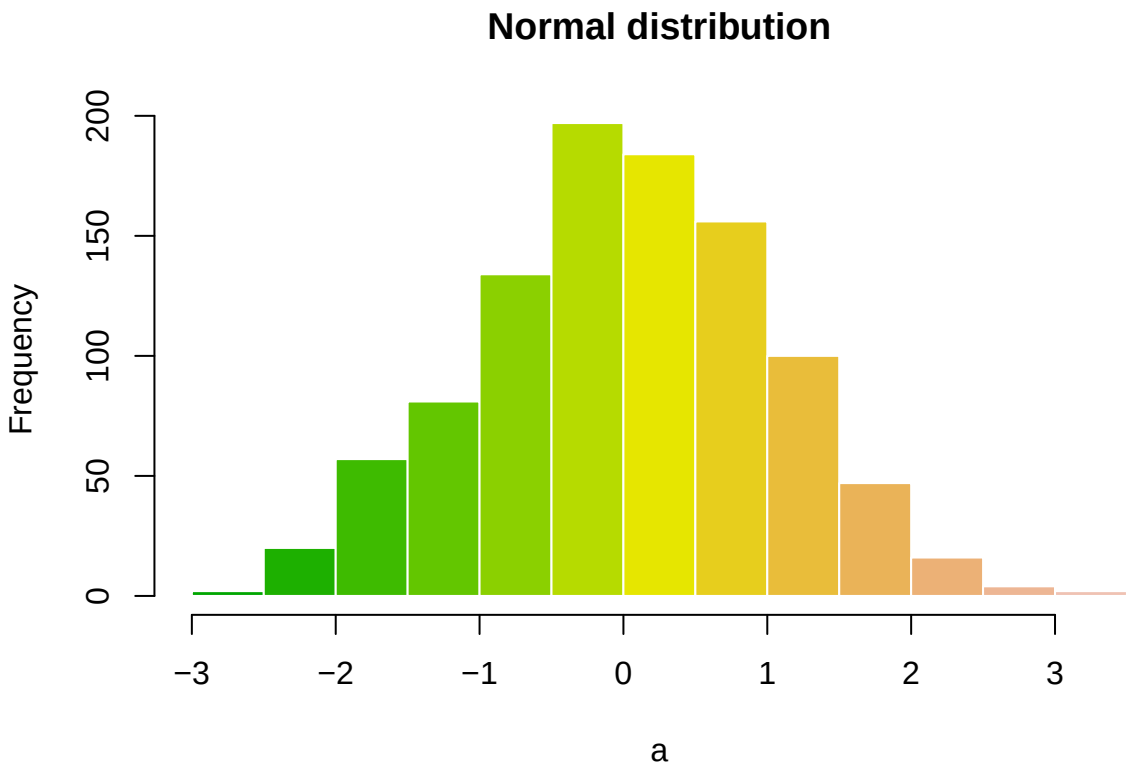
```
library(glue)
x = 3
a = "A"
print(glue('{x} and {a}'))
```

3 and A

Additionally, we should consider how we are organising our script here. Note we loaded a new library much later in the script. Ideally, we should include libraries at the start of the script to show what packages are included in our markdown/analyses. In general, when we are setting up our R markdown we should always consider if we are:

1. Organising R script logically
2. Using naming convention recommendations
3. Using descriptive naming of files
4. Writing readable code
5. Avoiding messy projects
6. Using relative file paths
7. Avoiding losing work (version control, back ups)

We next show the same figure from the presentation for clarity along with some additonal formatting things you can do on a page, including bullet points, text formats, other figures/analyses and tables.



We can output bullet points:

- 2 is the minimum distance driven
- 118 is the range between max and min distance driven
- 2, 120 is the min, max distance driven using the range() fn from baseR

You can also output tables quite easily using the kable package. This is nice as a viewing format in itself, and allows for high level of customisation in terms of the table output. Additionally, there are other quite useful packages for designing tables such as **xtable**, **stargazer**, **pander**, **tables**, and **ascii** which we don't discuss here but are all recommended.

Table 1: A knitr kable.

	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb
Mazda RX4	21.0	6	160	110	3.90	2.620	16.46	0	1	4	4
Mazda RX4 Wag	21.0	6	160	110	3.90	2.875	17.02	0	1	4	4
Datsun 710	22.8	4	108	93	3.85	2.320	18.61	1	1	4	1
Hornet 4 Drive	21.4	6	258	110	3.08	3.215	19.44	1	0	3	1
Hornet Sportabout	18.7	8	360	175	3.15	3.440	17.02	0	0	3	2

Perhaps now an **image** would be *nice*, let's say a picture of these cute ***penguins***:

Figure 1: Some penguins

This is where it is good to keep file paths and directories associated with a project consistent. Note here that because I have my project set to introR, and my image above is saved in this project directory, I can load the penguins.jpg image simply by typing that name in since my R session is currently set to look for files in the project directory. Similarly, if my colleague have their project directory set up and it includes this image, they can also run the markdown without having to change the file path name above.

However, if this wasn't the case, I would have to type the whole image path out which is quite long (see note below), increases chances of errors in file path name specification and makes it hard to share the markdown easily. By using the project/single directory for all files linked to the markdown, as long as your colleague has the same directory set up they can run this script easily without worrying about file path names or file locations.

You can also change the size of the image:

Figure 2: Some smaller penguins

You can alternatively include figures from another script that you have run but might not be part of the present analyses, as long as they are saved in an accessible format and preferably in your project directory for easy access. Or alternatively, you can link images from webpages.

Further analyses

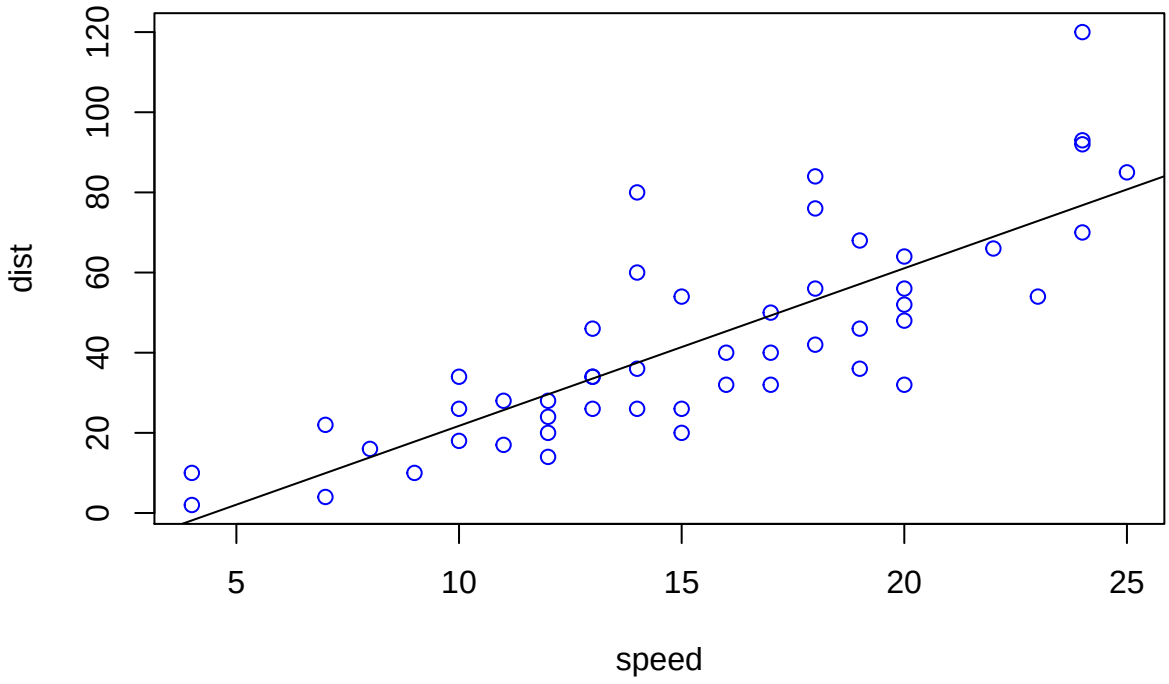
Finally, you could for example run a regression on our cars dataframe and output a nice plot from this using three different methods:

1. plot()
2. ggplot() and stat_smooth()
3. Function combining ggplot() and lm()

Note the ggplot() plotting function requires ggplot2 library to be loaded, which we have done at the start of our script! Additionally, although the numbered points above are written as each line starting with 1. in the markdown script, they still return sequential numeric values 1-3 in the output report.

First we plot our linear regression fit for the model (dist ~ speed) using plot():

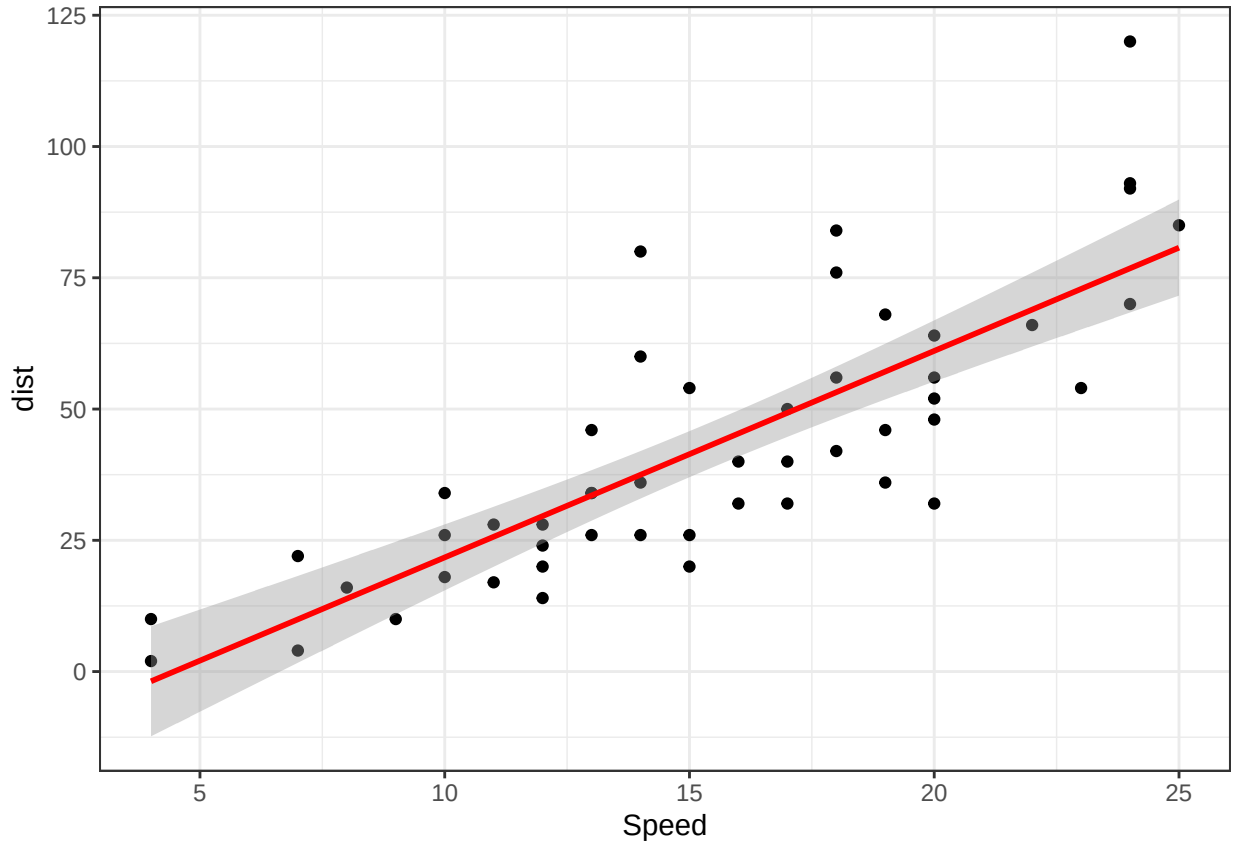
```
# This is where we use the gglpot2 package, but I included it at the top of the script!
# All your packages should be loaded into your R session before you start your analyses.

cars_reg = lm(dist ~ speed, data = cars)
plot(cars, col = "blue")
abline(cars_reg)
```



Second, we use ggplot() and the stat_smooth() function where we can specify the type of regression we would like the line to be fit to:

```
ggplot(cars, aes(x = speed, y = dist)) +
  geom_point() +
  stat_smooth(method = "lm", col = "red") + theme_bw() + labs(x = "Speed", Y = "Distance")
```

```
## `geom_smooth()` using formula 'y ~ x'
```

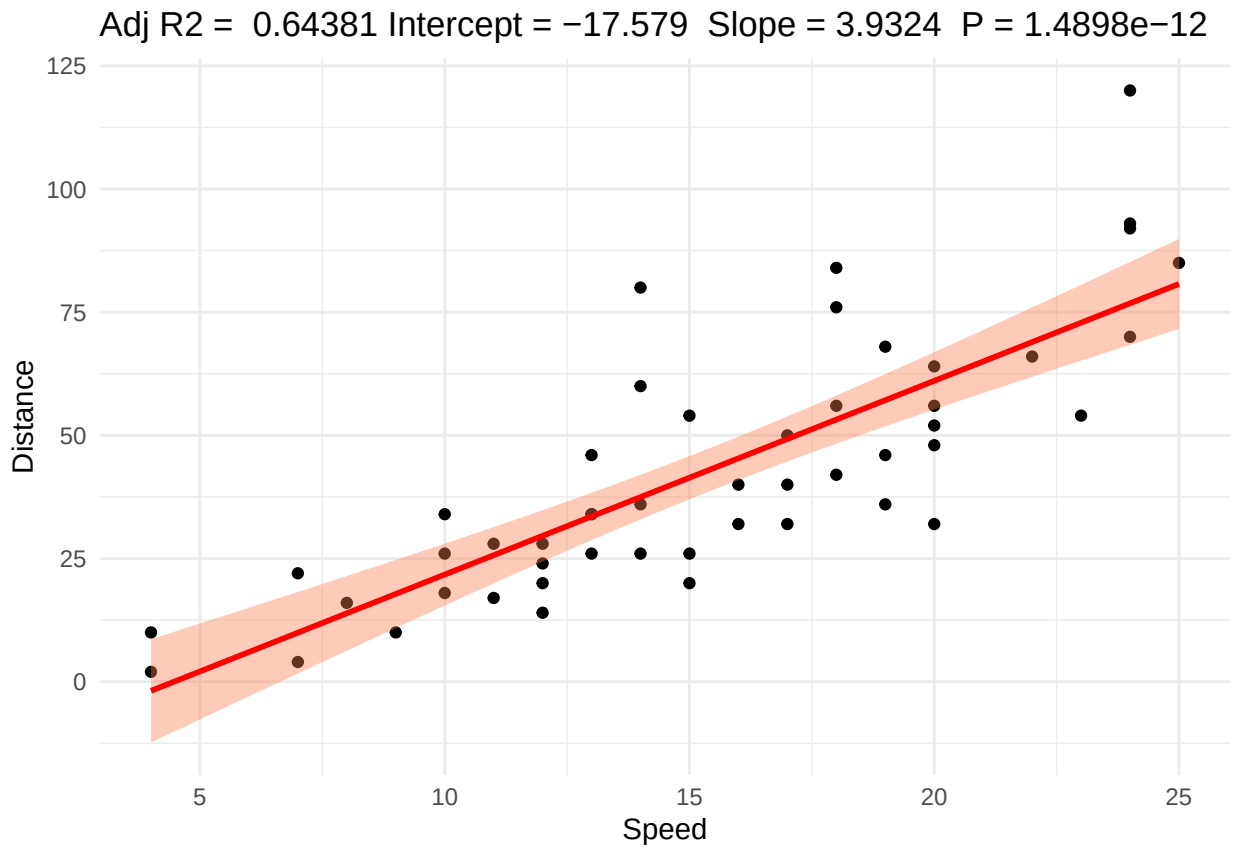


Notice here with this method the stat_smooth() funciton includes the addition of confidence intervals around the logistic regression fit. Look at ?stat_smooth() to see how to specify whether you want these confidence intervals included or not, as well as other regression fits that can be implemented in the method portion of the stat_smooth() function call.

Third we use a function that we have written ourselves, intitled ggplotRegression() which incorporates ggplot() and lm() function into one, allowing us to additionally easily specify at the top of the figure outputs of interest:

```
fit1 <- lm(dist ~ speed, data = cars) # run model
ggplotRegression(fit1) # use our written function to plot the model
```

```
## `geom_smooth()` using formula 'y ~ x'
```



We could alternatively write this as:

```
ggplotRegression(lm(dist ~ speed, data = cars))
```

which combines lines in the cars_regression_function_plot code chunk above into one succinct line of code instead. However, if you were planning to use the model fit for further analyses in your markdown, you would more likely want to store this in an object of its own (e.g. fit1 in the code chunk above).

Depending on your computer type, you might have to install tinytex package or another one of the latex language packages prior to being able to output a PDF from an R markdown script. This is because latex language requires a language processor to be installed in R before it will read/understand latex (which is the underlying language used to write PDFs).