Commenting Code: A Dem's Perspective #905

T

Thomas Elton STAFF

5 months ago in Projects – P3

Hi,

Inspired by a private post on Ed the other day, I wanted to share my thoughts on how to go about clean coding:

ggplot The Client Report demonstrates proficiency with ggplot. There is best practise with regards to clean coding, documentation, care to aesthetics, and labelling. Note: 1. 'Clean coding' means care to how the code is written. 'Documentation' means words in the code explaining the main steps. 2. 1901 (or 1001) students may choose to demonstrate coding proficiency in other ways than ggplot. 3. 'Aesthetics' means the overall impression of the output - ie Is it easy to read and everything fits (eg the choice of colours). 4. 'Labelling' means everything that makes a plot easer to read.	**2 pts** **Excellent** The coding could be given to a Colleague as an exemplar of best practise.	**1 pts** **Good** Most criteria.	**0 pts** **Poor or No attempt**	2 pts

What Even is a Comment?

If you are scratching your head on what I mean by this mysterious comment thing (we did mention it in week 1, but I understand that was ages ago), in R, you can leave comment in your code using the hash character:

```
# Outputs the first 10 factorial numbers
for (i in c(1:10)) {
    x <- factorial(i)
    print(x)
}

# NB: This is a for loop which we haven't learnt about yet.
# Please don't worry about it!
# What you should find cool is that even though you might not know
```

```
# how the for loop above works, you still understand what it does
# from the comment, which I think is pretty useful! :)
```

All of the text following the hash is ignored when your R code is interpreted.

> Fun fact, it is called a hash, and not a hash tag! We say hashtag on Insta or Twitter because we use a # followed by a tag, like #Data1001IsLit

What's the Point of Leaving Comments in Code?

One of the things that we have been trying to emphasise since day 1 is the importance of having reproducible reports. Whilst yes, technically without code comments your report is still reproducible (provided you have code folding) because someone could just re-run your code, in data science, transparency is key.

Having comments explaining what your code does (and how it does it) makes your methodology so much more transparent and easy to follow. It also allows other data scientists to have an easier time understanding what you have done, which is really important in the peer review process.

This is part of **communicating** your methodology, which is linked with learning outcome 10 for this unit:

> **LO10**. perform data exploration **in a team**, and **communicate** the findings via oral presentations and **reproducible reports**, with interrogation.

Note, although LO10 explicitly mentioned communicating findings, I would still argue that communicating your methodology is intrinsically tied to your findings, as without having a methodology, you wouldn't have findings to communicate! For example, is it actually have for you to communicate your linear regression model if you didn't check for the linearity assumption as part of your methodology?

Comments are also really important when working **in a team**, as this will allow your team mates to have an easier time understanding what the code you've written achieves.

It's also helpful for you when you come back to code after long period of time, because it helps you get back up to speed with what your code is trying to do!

Isn't Our Code Self Explanatory? Why Bother Commenting It?

I understand that you might be thinking that a lot of your code is self explanatory and doesn't require commenting.

The reality is, you have only been learning R for 10 weeks now, and so for the most part, you may be using only a few lines of code here and there. However, commenting your code

is still important!

First point)

We are at uni, and so we are easing you into R. Writing comments now for simpler programs is definitely a move in the right direction, and will prepare you for more complex coding which will come in second and third year!

For example, here is an excerpt from my DATA3888 assignment which was due a month ago:

```
# Loop through all directories.
for (i in 1:length(directories)) {

  # Get all .wav files within the directory.
  files = dir(directories[i])
  for (j in 1:length(files)) {

    # Extract the file we are dealing with this loop iteration.
    file = files[j]

    # Get actual value from file name
    actual = strsplit(file, split = "_")[[1]][1]

    # Get prediction using classifier
    wave_data = readWave(paste(directories[i], "/", file, sep = ""))
    prediction = streaming_classifier(wave_data)
```

Although this is only a brief excerpt, hopefully you can see how the comments aid the otherwise confusing code.

Second Point)

Although some code might seem quite obvious (like hopefully we would all be able to understand what a single line doing a linear regression is achieving), we have already started to seen more complex code examples...

cough, box model, cough, cough...

From lab 8, comments really aid understanding in more complex code examples:

```
# define oyster population
N <- 6000
n <- 800
yes <- 250 # oysters with pearls in sample
```

```
no <- n - yes # oysters without pearls in sample

# scaling up
pop_yes <- yes/n * N
pop_no <- no/n * N

# box model
box <- c(rep(1, pop_yes), rep(0, pop_no))

# expected value
ev <- mean(box)

# pop sd
library(multicon)
sd <- popsd(box)
sd

# standard error
se <- sd/sqrt(n)
se
```

How Should I Go About Commenting?

I know it's taken a while to get here, but now that we have finally arrived, how should you go about commenting?

Now, I would like to preface that this is the commenting style that I personally use, however, your tutor may have a difference preference! This is totally fine. In the workplace, you would have a style guide which tells you how you should comment so that all of your colleagues are commenting in a consistent manner.

Anyway, if given free choice, I keep the following in mind:

1) KIS - **K**eep **I**t **S**imple.

When you are writing your comments, get straight to the point. We are not after long elaborate paragraphs. These become jarring to read, and in my opinion, actually detract from the "clean coding" style that the marking rubric is after.

2) Not every line of code needs to be commented

When I was a DATA1x01 student, I wanted to leave no place where I could lose a mark, so I commented everything! In hindsight, this was not good! My problem with this is that having heaps and heaps of comments (where non are warranted) detracts from your main 'useful' comments.

For example, if I was to use the following code (again, an example from a previous assignment), I believe one single comment like I have done is sufficient.

```
# Create bar graph of distribution of values
bar_chart = accuracies_df %>% ggplot() +
 aes(x = accuracy_1) +
 geom_bar(fill="steelblue") +
 theme_minimal() +
 labs(x = "Levenhshtein Distance",
      y = "Count",
      title = "Levenhshtein Distance Between Actual and Predicted Classifications")
```

This is as opposed to:

```
# Create bar graph of distribution of values
bar_chart = accuracies_df %>% ggplot() +
 aes(x = accuracy_1) + # Set x to the accuracy_1 column
 geom_bar(fill="steelblue") + # Create a bar graph using the steel blue fill.
 theme_minimal() + # Use the minimal theme
 labs(x = "Levenhshtein Distance", # Set x axis label
      y = "Count", # Set y axis label
      title = "Levenhshtein Distance Between Actual and Predicted Classifications") # Se
```

I don't personally feel that the comments in the second example add anything helpful, especially lines 6 to 8!

3) Use common sense!

I know this might sound a bit strange, but at the end of the day, you are the one communicating your code. If you think an extra comment here or there would be useful and aid understanding, then please add it!

Bonus: Good Use of Variable Names

Part of "clean coding", and definitely a good habit to get into, is ensuring that you make use of good and self-explanatory variable names in your code.

For example, if you have a column measuring people's height, call that column height! Or, if you are using the public toilets data in project 3, why not call your data frame toilet_data?

Not only do good variable names help when someone (like a marker...) reads you code, it is also helpful for yourself when it comes to debugging. If R tells you that there is an error with the toilets_data data frame, you know exactly what variable we are having issue with!

Anyway, these are just some of my thoughts and feelings!