Module 1 - Learning R Fundamentals with a Protein Standard Curve

Paul Brennan

Created: July 2017; Updated Sep 2020

### Welcome to the first R for Biochemists 101 Training module.

Each module is designed to take approximately 60 minutes to complete.

The material provided includes text to read, video demonstrations, an example R script with exercise. We hope you enjoy the modules. We welcome feedback and comments.

### How to use this module

1. To learn from this module, cut and paste the R code into R-Studio and then run the code line by line.
2. See if you can make the script work
3. Watch the demonstration videos to see how the code works
4. Look at how the ‘Global Environment’ changes.
5. Change the code and test it
6. Try the exercises at the end
7. Extend your learning using the resources at the end

### Learning Objectives for Module 1

* Use R Studio
* Understand the concepts of objects and functions
* Download a package
* Create plot with ggplot2

### The Experiment - a Protein Assay

Our first experiment is a protein assay. Protein assays are protein measurements from cell or tissue extracts. They are a fundamental biochemical experimental technique. Protein assays are used to determine the amount of protein in a set of samples. Usually the samples have been extracted from cells or tissue. It is anticipated that further experiments will then be performed.

Protein assays depend on the incubation of a protein mixture with a dye that changes colour when bound to the protein. A measurement of the colour change is taken by a change in absorbance. There are various kinds depending on the types of samples being measured. A simple example is the Bradford protein assay which comes as a single solution available from many suppliers. It cannot be used for samples lysed with detergent so detergent compatible solutions are available too.

To do a protein assay, we use ‘standards’. These ‘standards’ are made from a protein solution of known concentration. Changes in absorbance can be measured in absorbance cuvettes or in 96 well plates. For the data in this example, we added a small volume of our sample with known protein concentrations into a 96 well plate. Then the Bradford dye that binds the proteins was added.

We read the absorbance of the resulting solution using an absorbance plate reader at a wavelength of 570nm. The darker the solution, the more light will be absorbed. Thus more protein gives a darker colour and an increase in absorbance. This kind of assay is used for determining the amount of protein in a solution and is a very common technique in most biochemical laboratories in academic institutions and companies.

We will to use R to vizualise and analyse our data. The first step is to learn about the fundamentals of R - functions, objects and packages.

### First video contains an introduction to R-Studio and getting started

### Fundamentals - functions and objects

To do this, we start with two fundamentals that we need to understand: \* functions \* objects

#### First ‘Fundamental’ is functions

Functions do things! You know it’s a function because it contains brackets. A function has arguments. The arguements are always inside the brackets.

c( ) is a function - usually called combine or concatenate.

#### Second ‘Fundamental’ is objects

Objects contain data. The data can be in a variety of types. We make object with functions. Bringing these two ‘Fundamentals’ together we use a function to create an object.

The “arrow” “<-” is a key part of R code. The arrow is like an equals (=) sign but is more definative. We use it to create or overwrite an object usually with name of the object on the left hand side and the function to create the object on the right hand side.

In this example, we use the combine function, c( ), to create the object prot\_conc which is a list of protein concentrations. Because we did the measurement in duplicate, we have included two lines of these concentrations.

Here is the code to create this object. Cut and paste it into R and try running the code. It should create an object called prot\_conc in the Global Environment.

prot\_conc <- c(0.000, 0.016, 0.031, 0.063, 0.125, 0.250, 0.500, 1.000,   
 0.000, 0.016, 0.031, 0.063, 0.125, 0.250, 0.500, 1.000)

If you look at the R-Studio environment window, you will see that something has appeared there. It’s the object called prot\_conc. This is a group of numbers (see the word “num”) and it contains 16 elements (in square brackets it says “[1:16]”. This type of object is called a ‘vector’.

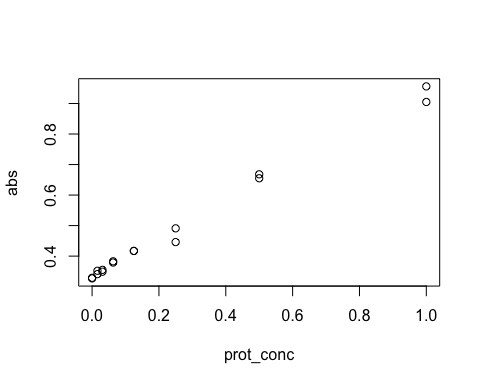
In our experiment, we also have a list of absorbances (abs). We also need to create an object with these.

abs <- c(0.329, 0.352, 0.349, 0.379, 0.417, 0.491, 0.668, 0.956,   
 0.327, 0.341, 0.355, 0.383, 0.417, 0.446, 0.655, 0.905)

The object ‘abs’ has now been created in our environment. This object also has 16 elements - the same size as prot\_conc. Is also a vector of numbers.

With two numeric vectors of the same length, like these two, we can plot these using the plot( ) function. We give the function two arguments - the values to use for the x axis and the values to use for the y-axis.

plot(prot\_conc, abs)



If you been following this, I hope you’ve made your first graph in R.

### Exercise 1

You can test your knowledge using this data. You will need to reformat the data so that it looks similar to the code we have previously used.

Protein concentrations: 0.0, 0.05, 0.1, 0.2, 0.4, 0.8, 1.6, 0.0, 0.05, 0.1, 0.2, 0.4, 0.8, 1.6

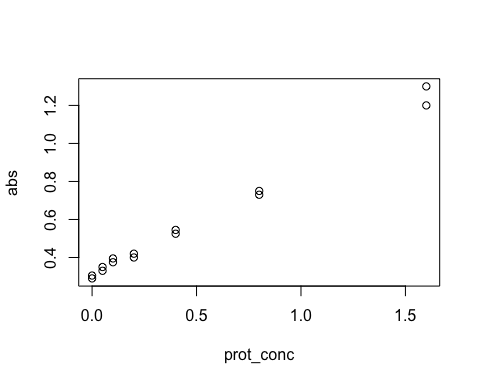
Absorbances: 0.29, 0.35, 0.375, 0.4, 0.525, 0.75, 1.2, 0.305, 0.33, 0.395, 0.42, 0.545, 0.73, 1.3

Try this yourself before you turn the page…

### Answers for Exercise 1

Here is the code

prot\_conc <- c(0.0, 0.05, 0.1, 0.2, 0.4, 0.8, 1.6, 0.0, 0.05, 0.1, 0.2, 0.4, 0.8, 1.6)  
abs <- c(0.29, 0.35, 0.375, 0.4, 0.525, 0.75, 1.2, 0.305, 0.33, 0.395, 0.42, 0.545, 0.73, 1.3)  
  
plot(prot\_conc, abs)



### Second video demonstrates ggplot2.

We’ve started with base plotting in R. It’s useful but I usually prefer to use another way. My preferred way is to use the package ggplot2.

### Third ‘Fundamental’ is packages

Base R is great and contains lots of useful functions. But what makes R even more useful is the huge number of packages that are available and are being developed by an active community across the world. One of the most useful packages for graphs is the package **ggplot2**. To use a package, you have to download or ‘install’ it.

# install.packages("ggplot2") # to run remove the comment mark  
# - don't want to run this everytime.

and then you have to activate it to make it useable in the R environment.

library(ggplot2)

ggplot2 works best with a different kind of object - a data.frame. This is a two dimensional array of text and numbers - a bit like an Excel spreadsheet. We need to convert our two vectors into a data.frame. We do this using the function as.data.frame( ). We do it first with one vector.

data <- as.data.frame(prot\_conc)  
  
# the str() function allows us to find out about the structure of the data.  
str(data)

## 'data.frame': 14 obs. of 1 variable:  
## $ prot\_conc: num 0 0.05 0.1 0.2 0.4 0.8 1.6 0 0.05 0.1 ...

If we look at the environment in R-Studio we see a new object has been created. This object is called ‘data’ and contains 16 observations (“obs.”) of 1 variable. We can add in the absorbance (“abs”) values into this object. The str() function tells us that this is a data.frame.

Then we add the other vector - abs. Now our object has 16 observations of 2 variables. This is 16 rows and 2 columns.

data$abs <- abs

We can look at this object.

data

### Make a plot with ggplot2

**Step 1: add the data and then the ‘aesthetics’**

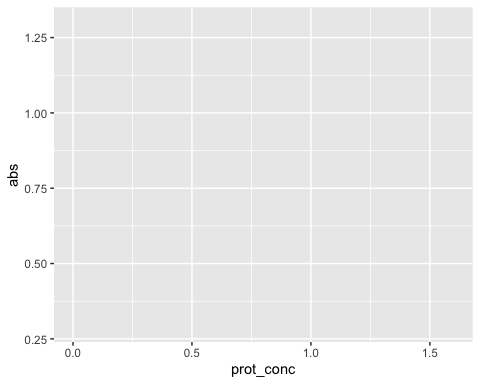
The key asthetics in this case are the data for x and y axis.

p <- ggplot(data=data, # specify the data frame with data  
 aes(x=prot\_conc, y=abs)) # specify x and y for the graph

This creates another object called ‘p’. This object is a list. A list is a one dimensional object that can contain different types of data.

If you want to see p – you can. You can see any object by typing the name of the object into the Console.

p

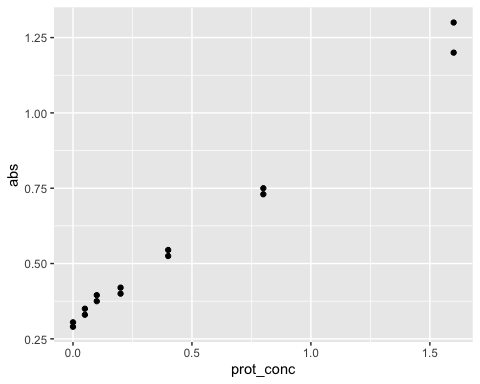


It’s a blank plot because we haven’t given any information on the type of plot we want to make.

**Step 2: add a type of graph**

The plus sign is particular to ggplot and allows us modify the object p with more layers. In this case we want to add a layer that contains the data represented as points. This gives us a scatter plot or dot plot.

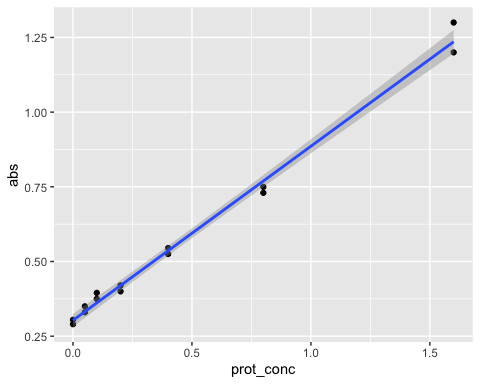
p <- p + geom\_point()  
  
p # show the graph again



**Step 3: show the graph with a line on it.**

We use the function stat\_smooth( ) with the arguments “method = lm” and “formula = y ~ x” inside the brackets. lm refers to linear model. formula tells us to compare x and y. The function calculates a least mean squares linear model and draws it on the graph. The default colour is blue and the grey area shows the 95% confidence interval.

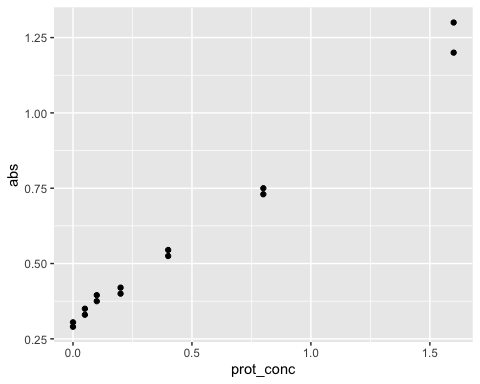
p + stat\_smooth(method = "lm", formula = y ~ x)



It’s important to note the difference between altering the object (using <-) and just altering the object when showing the graph (using + but not <-). Only the <- will modify the object. If we use the plus (+) sign without the arrow we only make the graph to show it. We don’t modify the object.

If we show the object p again:

p



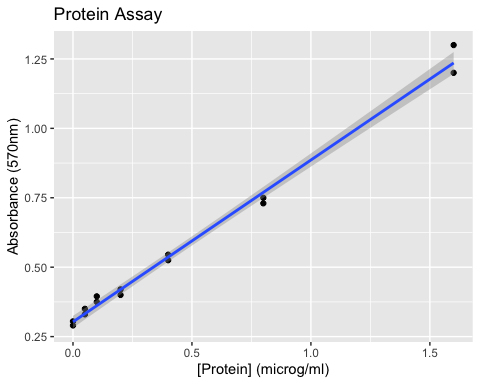
There is no line.

ggplot uses what is called the “grammer of graphics”. A key concept is the addition of layers to the plot. Examples of layers are the one generated by the geom\_point() function which is a layer containing plots.

**Step 4: Add extra layers to customize your plot**

We can also add a layer with a label for the x-axis, the y-axis and the title of the graph with the following code:

p + stat\_smooth(method = "lm", formula = y ~ x) +  
 xlab("[Protein] (microg/ml)") + # label x-axis  
 ylab("Absorbance (570nm)") + # label y-axis  
 ggtitle("Protein Assay") # add a title



**Step 5: Save your plot to show and share**

Finally to save this plot, you can use the ggsave() function.

p + ggsave("ProteinStandardCurve.pdf")

### Exercise 2

Play with the code: \* Try changing the colour the points:

p + geom\_point(colour = "blue")

* Try to change the size of the points

p + geom\_point(size = 5, colour = "red")

### 

### Exercise 3

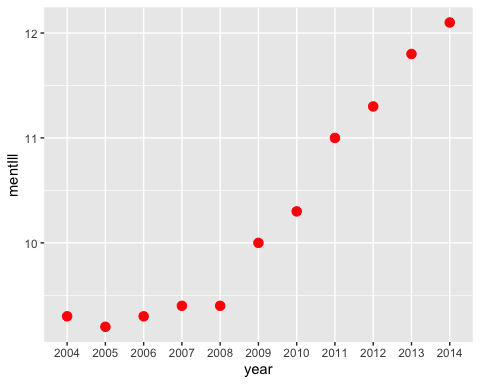
Here is some more data. Try to graph the data with it.

year <- c("2004", "2005", "2006", "2007", "2008",   
 "2009", "2010", "2011", "2012", "2013", "2014")  
  
mentIll <- c(9.3, 9.2, 9.3, 9.4, 9.4, 10.0, 10.3, 11.0, 11.3, 11.8, 12.1)

Try making the plot before you look at the code on the next page.

### Answers for Exercise 3

data2 <- data.frame(year, mentIll)  
p2 <- ggplot(data2, # a data.frame with the data  
 aes(x=year, y=mentIll)) + # columns of data.frame  
 geom\_point(colour = "red", size = 3) # type of plot   
p2 # show the plot



### Review what we have learned

* We have used some functions - they do things
  + c( ) – the combine function
  + plot( ) – the base R plot function
  + install.packages(“ggplot2”) – downloads a package
  + library(“ggplot2”) – activates the package so that we can use it.
  + then we used a selection of functions from ggplot2
* We have created some objects - they hold data
  + vectors – one dimensional objects with single type of object (number in this case)
  + data frames – two dimensional objects that can contain different types of data
  + lists – one dimensional object that can contain differenty types of data
* We have used a package (ggplot2) - packages give us extra functionality

### Resources

* You could move onto Module 2 - Extracting data and making a customised graph.
* Find out more about R-Studio at the R-Studio website
* Try out a script from the R for Biochemists website
  + [Exploring diseases in Wales] (<http://rforbiochemists.blogspot.co.uk/2015/10/exploring-diseases-in-wales-for-sql.html>)
* Learn how to search for help within R

?plot # more information inlcuding examples on the plot function

* Explore some of the following links
  + ggplot2 book written by the creator - (<http://ggplot2.org/book/>)
  + ggplot documentation pages - (<http://docs.ggplot2.org/current/>)
* Do a search in your browser – Stackoverflow is a very useful resource. For example here is demonstration of how to make a boxplot -

(<http://stackoverflow.com/questions/32046242/boxplots-in-ggplot2-r>)