R Plotting and Aesthetics.

Brady Johnston

26/08/2019

# Plotting Biological Data with R

Hopefully from this tutorial you’ll be a bit more comfortable interacting with R, and be able to make simple plots from some biological data.

To begin with, create a new R script and save it in the tutorial directory.

## Importing you data.

To start working we need to import our data. It is in a .csv file in 3 columns: protein concentration and duplicate observations of the absorbance.

To read it in, we use the command read.csv() supplying it with a string that is the path to the file.

data <- read.csv("./abs.csv")  
print(data)

## prot abs1 ab2  
## 1 0.000 0.329 0.327  
## 2 0.016 0.352 0.341  
## 3 0.031 0.349 0.355  
## 4 0.063 0.379 0.383  
## 5 0.125 0.417 0.417  
## 6 0.250 0.491 0.446  
## 7 0.500 0.668 0.655  
## 8 1.000 0.956 0.905

## Machine-readable data

While this is how we would normally table our data collection, in reality we’re cutting corners. Machines don’t like cutting corners and it’s better to have data in a ‘long’ format where each observation (absorbance measurement) has its own concentration associated with it.

data[,1:2]

## prot abs1  
## 1 0.000 0.329  
## 2 0.016 0.352  
## 3 0.031 0.349  
## 4 0.063 0.379  
## 5 0.125 0.417  
## 6 0.250 0.491  
## 7 0.500 0.668  
## 8 1.000 0.956

abs1 <- data[,1:2]  
colnames(abs1) <- c("prot", "abs")  
abs2 <- data[,c(1,3)]  
colnames(abs2) <- c("prot", "abs")  
combined <- abs1  
combined <- rbind(combined,abs2)  
combined

## prot abs  
## 1 0.000 0.329  
## 2 0.016 0.352  
## 3 0.031 0.349  
## 4 0.063 0.379  
## 5 0.125 0.417  
## 6 0.250 0.491  
## 7 0.500 0.668  
## 8 1.000 0.956  
## 9 0.000 0.327  
## 10 0.016 0.341  
## 11 0.031 0.355  
## 12 0.063 0.383  
## 13 0.125 0.417  
## 14 0.250 0.446  
## 15 0.500 0.655  
## 16 1.000 0.905

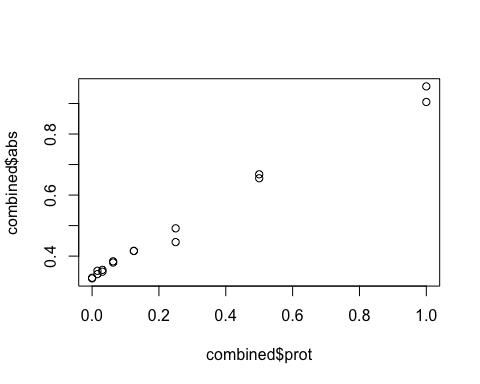
## Why though?

While it seems like it was a lot of work for nothing (you probably could have done this in excel in less time), when you start dealing with considerably larger dadta sets it starts coming in handy. Also if you set it up properly in excel then you can automate it later using scripts.

## Time to start plotting some data.

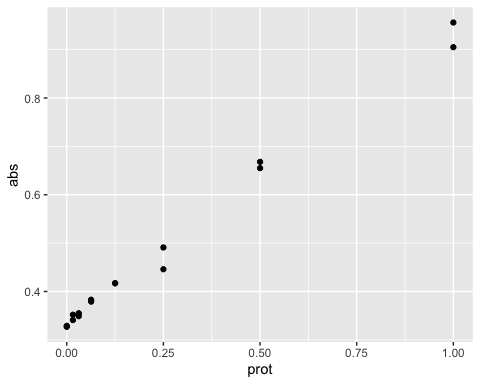
A simple plot is easy enough.

plot(combined$prot, combined$abs)



This tutorial is after all about *aesthetics* so we’re going to be using a package called ggplot2.

library(ggplot2)  
#ggplot(dataframe, aes(x data, y data)) + plottype  
plt <- ggplot(combined, aes(prot, abs)) + geom\_point()  
plt



Now that’s looking a bit nicer. We want to plot a line of best fit, get the equation and plot our unknows.

line <- lm(combined$abs~combined$prot)  
summary(line)

##   
## Call:  
## lm(formula = combined$abs ~ combined$prot)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -0.040756 -0.006025 0.004625 0.007011 0.030494   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) 0.336005 0.005801 57.92 < 2e-16 \*\*\*  
## combined$prot 0.603002 0.014211 42.43 3.42e-16 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 0.01843 on 14 degrees of freedom  
## Multiple R-squared: 0.9923, Adjusted R-squared: 0.9917   
## F-statistic: 1801 on 1 and 14 DF, p-value: 3.421e-16

Get the details that you require from the linear model (intercept, slope).

int <- summary(line)$coefficients[1]  
slope <- summary(line)$coefficients[2]

With this linear model information, we can calculate our unknowns.

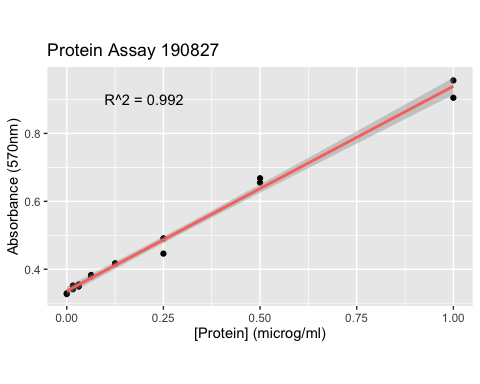
rsquare <- round(summary(line)$r.squared, digits = 3)  
  
abs.unks <- c(0.544, 0.568, 0.705)  
prot.unks <- (abs.unks - int) / slope  
  
calcdata <- as.data.frame(abs.unks)  
prot.unks

## [1] 0.3449324 0.3847333 0.6119300

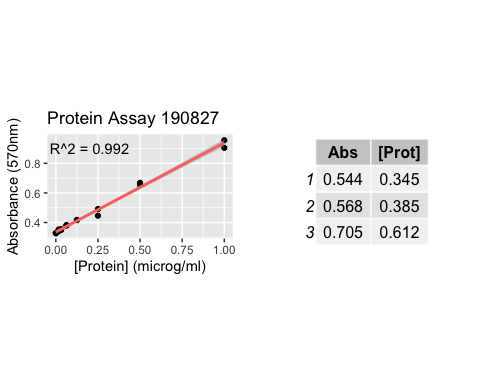
calcdata$prot <- round(prot.unks, digits = 3)  
calcdata

## abs.unks prot  
## 1 0.544 0.345  
## 2 0.568 0.385  
## 3 0.705 0.612

p <- ggplot(combined, # specify the data frame with data  
 aes(prot,abs)) + # specify the x and y for the graph  
 geom\_point() + # make a scatter plot  
 stat\_smooth(method = "lm", aes(colour = "red")) + # add a linear model line  
 xlab("[Protein] (microg/ml)") + # label x-axis  
 ylab("Absorbance (570nm)") + # label y-axis  
 ggtitle("Protein Assay 190827") + # Add a title  
 theme(legend.position = "", aspect.ratio = 9/16) + # remove unnecessary legend  
 annotate("text", 0.2,0.9, label = paste("R^2 =",rsquare)) #annotate with R^2 value  
 p



library("gridExtra")  
library("grid")  
colnames(calcdata) <- c("Abs", "[Prot]")  
t1<-tableGrob(calcdata)  
rm(calcdata)  
grid.arrange(p,t1, nrow=1)



# Bringing it together

All of the previous can be summarised in a script that looks like the following. Copy and paste this script into a new R document, change the ./abs.csv to another file and edit the abs.unks to c(0.44,0.53,0.68) then just run the scipt. Now you have a script that will analyse your data and produce great graphics in a matter of seconds.

You can even make this executable from the command line and take inputs, but that’s for another tutorial.

library("gridExtra")  
library("grid")  
library("ggplot2")  
  
data <- read.csv("./abs.csv")  
  
abs1 <- data[,1:2]  
colnames(abs1) <- c("prot", "abs")  
abs2 <- data[,c(1,3)]  
colnames(abs2) <- c("prot", "abs")  
combined <- abs1  
combined <- rbind(combined,abs2)  
  
line <- lm(combined$abs~combined$prot)  
  
int <- summary(line)$coefficients[1]  
slope <- summary(line)$coefficients[2]  
rsquare <- round(summary(line)$r.squared, digits = 3)  
  
abs.unks <- c(0.544, 0.568, 0.705)  
prot.unks <- (abs.unks - int) / slope  
  
calcdata <- as.data.frame(abs.unks)  
calcdata$prot <- round(prot.unks, digits = 3)  
  
p <- ggplot(combined, # specify the data frame with data  
 aes(prot,abs)) + # specify the x and y for the graph  
 geom\_point() + # make a scatter plot  
 stat\_smooth(method = "lm", aes(colour = "red")) + # add a linear model line  
 xlab("[Protein] (microg/ml)") + # label x-axis  
 ylab("Absorbance (570nm)") + # label y-axis  
 ggtitle("Protein Assay 190827") + # Add a title  
 theme(legend.position = "", aspect.ratio = 9/16) + # remove unnecessary legend  
 annotate("text", 0.2,0.9, label = paste("R^2 =",rsquare)) #annotate with R^2 value  
  
  
colnames(calcdata) <- c("Abs", "[Prot]")  
t1<-tableGrob(calcdata)  
rm(calcdata)  
grid.arrange(p,t1, nrow=1)