Module 3 - Graphing and analysing enzymology data using R

Paul Brennan

July 2017

### Welcome to R for Biochemists 101 Module 3

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

The material provided includes text to read, video demonstrations, an example R script and exercises. 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

### Learning objectives

1. Integrate our learning in a different context
2. Learn about facet\_wrap() function in ggplot2
3. Subset our data
4. Apply geom\_smooth()
5. Do Nonlinear Least Squares fit of enzymatic data

### The Experiment

The data from this experiment was provided by Dr Claire Bennett, University of Bath. Some of the data was published here: <http://www.jbc.org/content/285/44/33701.full> and her PhD thesis is published here <http://opus.bath.ac.uk/27220/> My thanks to Dr Bennett for sharing her data.

The experiment using two different enzymes - one wild type and one with a single amino acid substitution.

The velocity of the two enzymes was measured using fixed concentration of substrates.

The first step of data analysis is to plot and look at the data. Then we will add some lines and extract some calculations.

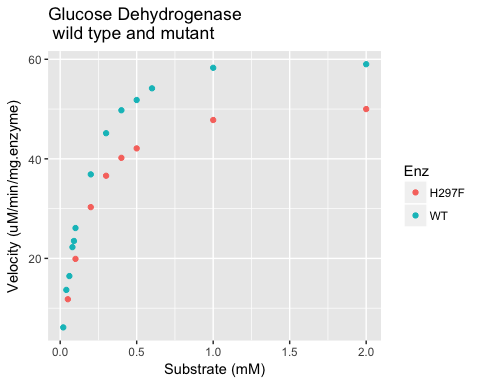
### Visualising and analysing enzymology data in R

This code is illustrated in the first code demo. #### Create the data in a dataframe

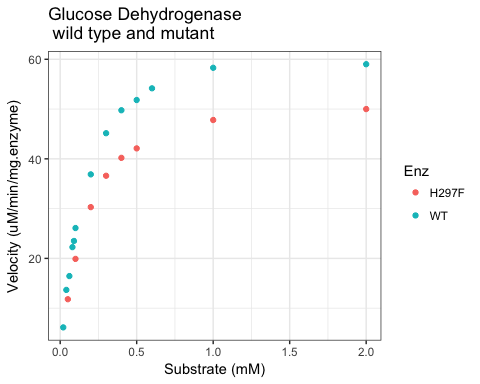
# This is the data  
Enz <- c("WT","WT","WT","WT","WT",  
 "WT","WT","WT","WT","WT",  
 "WT","WT","WT",  
 "H297F","H297F","H297F",  
 "H297F","H297F","H297F",  
 "H297F","H297F")  
S <- c(2.00, 1.00, 0.60, 0.50, 0.40,   
 0.30, 0.20, 0.10, 0.09, 0.08,   
 0.06, 0.04, 0.02,   
 0.05, 0.10, 0.20,   
 0.30, 0.40, 0.50,   
 1.00, 2.00)  
v <- c(59.01, 58.29, 54.17, 51.82, 49.76,   
 45.15, 36.88, 26.10, 23.50, 22.26,   
 16.45, 13.67, 6.14,   
 11.8, 19.9, 30.3,   
 36.6, 40.2, 42.1,   
 47.8, 50.0)  
  
# assemble the data into a data.frame  
enzdata <- as.data.frame(Enz)  
enzdata$S <- S  
enzdata$v <- v

Historically, enzymatic data was turned into lines using various transformations such as the Lineweaver-Burk plot but current computational technique advoate a non-linear fit of the data.

library(ggplot2)  
library(ggthemes)  
# plot the data with ggplot  
# most of the syntax seems relatively easy to understand...  
ggplot(data=enzdata, # give the ggplot() function the data  
 aes(x=S, # data.frame col with values for x-axis  
 y=v, # and the y-axis  
 colour = Enz)) + # colour by WT or H297F \*\*NOTE the "+"  
 geom\_point() + # key function that show points  
 xlab("Substrate (mM)") + # label x-axis  
 ylab("Velocity (uM/min/mg.enzyme)") + # label y-axis  
 ggtitle("Glucose Dehydrogenase \n wild type and mutant")



# we can create an object with the plot in it...   
enz\_plot <- ggplot(data=enzdata,   
 aes(x=S,   
 y=v,   
 colour = Enz)) +   
 geom\_point() +   
 xlab("Substrate (mM)") +   
 ylab("Velocity (uM/min/mg.enzyme)") +   
 ggtitle("Glucose Dehydrogenase \n wild type and mutant")  
  
# then we can apply a different theme to change the style of the plot.   
enz\_plot + theme\_bw()

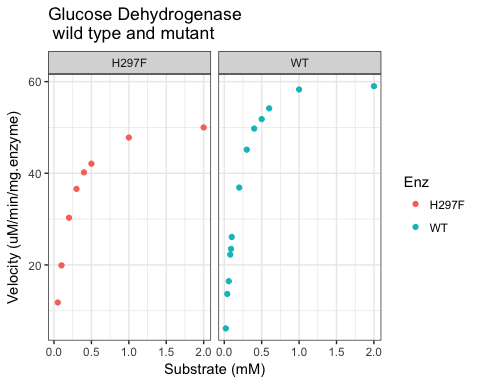


#### Using facet\_wrap to give us two plots

ggplot2 has a clever way of separating our plots which can be useful. The function is called facet\_wrap(). Using the tilda and a column in our data frame that has categories or factors, ggplot will separate the plots into those categories.

The facet\_wrap() function requires at least one argument. If you try to run it without an argument you will get an error message. This code will display the facet\_wrap() version of enz\_plot using Enz as a category.

enz\_plot + facet\_wrap(~Enz) + theme\_bw() # this function gives two plots



There is a title in each plot and the y-axis is the same for both plots.

Note the order of the plots. Our mutant enzyme plots first. ggplot2 plots the categories or factors in alphabetical order as a default.

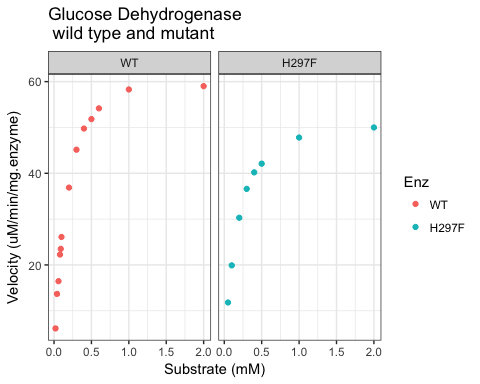
It is possible, and probably wise, to reformat our data so that the wild type enzyme is on the left. It requires the reformatting of our data.frame.

This code replaces the column enzdata$Enz with a new version with the factors in a customised order - i.e. with wild type, "WT" first.

# re-format the data in our dataframe  
enzdata$Enz <- factor(enzdata$Enz, levels = c("WT", "H297F"))

We need to create a new enz\_plot object using our reformatted data.

# make a new object with the reformatted in it...   
enz\_plot <- ggplot(data=enzdata,   
 aes(x=S,   
 y=v,   
 colour = Enz)) +   
 geom\_point() +   
 xlab("Substrate (mM)") +   
 ylab("Velocity (uM/min/mg.enzyme)") +   
 ggtitle("Glucose Dehydrogenase \n wild type and mutant")  
  
# then facet\_wrap() again   
enz\_plot + facet\_wrap(~Enz) + theme\_bw()



Now the wild type plot ("WT") is on the left.

### Exercise 1

You can test your coding skills a different data set.

Using this data, try assembling a data.frame and ploting the data yourself before you turn the page...

Sub <- c(0, 1, 2, 4, 8, 12, 16, 20, 30, 40)  
  
exp\_1\_vel <- c(0, 15.7, 29.42286, 45.64, 62.60615, 75.78118, 69.88, 75.256, 89.59429, 86.84)   
  
exp\_2\_vel <- c(0, 2.190476, 5.254545, 8.95, 15.628571, 20.8, 25.355556, 26.55, 32.44, 33.333333)

Suggestions:

1. it might be easier to plot each experiment separately first
2. the rbind() function will combine rows at the bottom of a data.frame.

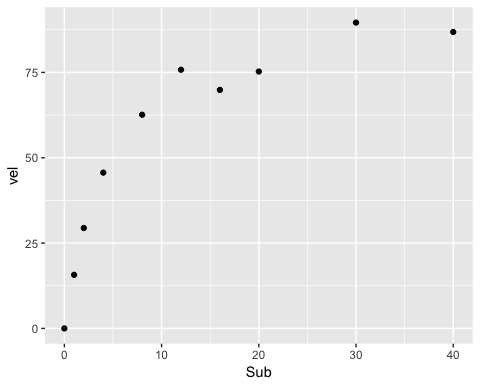
### Answers for Exercise 1

Create a data.frame

Sub <- c(0, 1, 2, 4, 8, 12, 16, 20, 30, 40)  
  
exp\_1\_vel <- c(0, 15.7, 29.42286, 45.64, 62.60615, 75.78118, 69.88, 75.256, 89.59429, 86.84)   
  
exp\_2\_vel <- c(0, 2.190476, 5.254545, 8.95, 15.628571, 20.8, 25.355556, 26.55, 32.44, 33.333333)  
  
data2 <- as.data.frame(Sub)  
data2$vel <- exp\_1\_vel  
data2$exp <- c("WT") # note this will fill the whole column

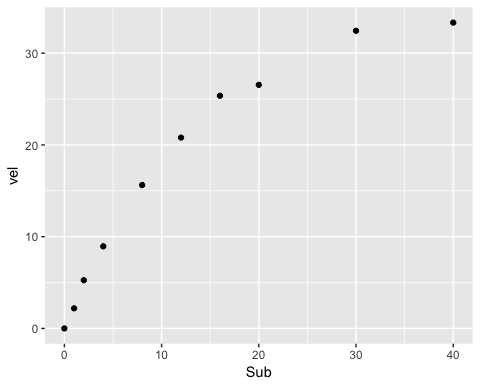
Plot first set of data

ggplot(data=data2,   
 aes(x=Sub,   
 y=vel)) +   
 geom\_point()



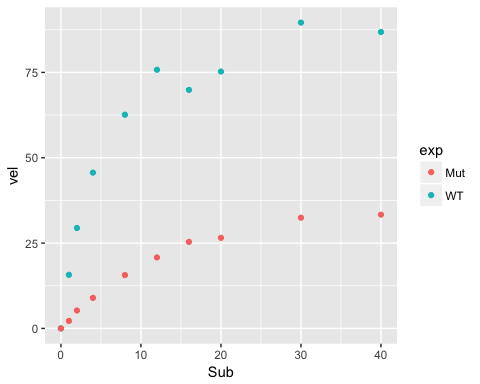
Repeat for second set of data

data3 <- as.data.frame(Sub)  
data3$vel <- exp\_2\_vel  
data3$exp <- c("Mut")   
  
ggplot(data=data3,   
 aes(x=Sub,   
 y=vel)) +   
 geom\_point()



Use the rowbind function rbind() to combine the data.frames and plot both together.

data\_both <- rbind(data2, data3)  
  
ggplot(data=data\_both,   
 aes(x=Sub,   
 y=vel,   
 colour = exp)) +   
 geom\_point()



Now try adding titles, creating the plot object, adding a theme you like and using facet\_wrap().

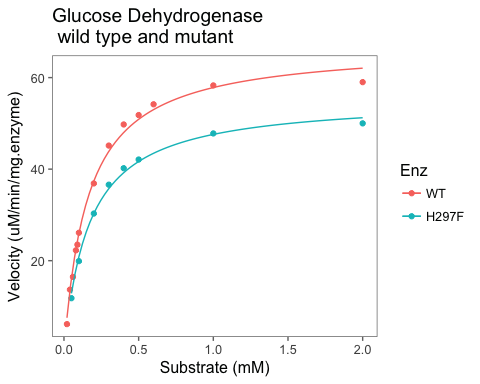
#### Showing a smooth curve on our plot

This code is illustrated in the second code demo.

I would like to explain the arguments within the geom\_smooth() function:

1. we subset the data - as we are fitting two separate lines
2. we have a method - we're using nls - non-linear least squares
3. we have method.args (method arguments) - this provides the formula and start values for the formula
4. we have the se - set to FALSE - essential for this line
5. we have a size for the line - you can change this...
6. we have applied the theme\_few() at the end.

# we can add the enzyme kinetic lines using the geom\_smooth() function  
enz\_plot + geom\_smooth(data = subset(enzdata, Enz=="WT"),  
 method = "nls",   
 method.args = list(formula = y ~ Vmax \* x / (Km + x),   
 start = list(Vmax = 50, Km = 0.2)),  
 se = F, size = 0.5) +   
 geom\_smooth(data = subset(enzdata, Enz=="H297F"),  
 method = "nls",   
 method.args = list(formula = y ~ Vmax \* x / (Km + x),   
 start = list(Vmax = 50, Km = 0.2)),  
 se = F, size = 0.5) +  
 theme\_few()

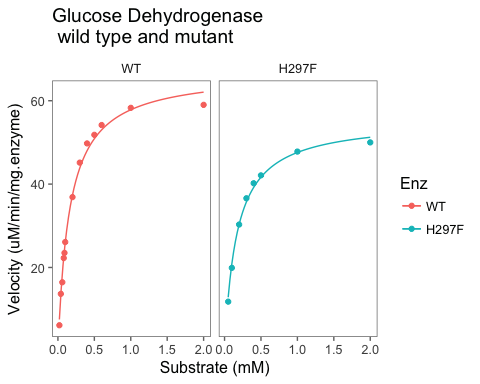


We can create an object called enz\_plot\_lines...

##   
enz\_plot\_lines <- enz\_plot +   
 geom\_smooth(data = subset(enzdata, Enz=="WT"),  
 method = "nls",   
 method.args = list(formula = y ~ Vmax \* x / (Km + x),   
 start = list(Vmax = 50, Km = 0.2)),  
 se = F, size = 0.5) +   
 geom\_smooth(data = subset(enzdata, Enz=="H297F"),  
 method = "nls",   
 method.args = list(formula = y ~ Vmax \* x / (Km + x),   
 start = list(Vmax = 50, Km = 0.2)),  
 se = F, size = 0.5) +  
 theme\_few()

... and we can facet\_wrap it again:

enz\_plot\_lines + facet\_wrap(~Enz)



### Exercise 2

You can test your coding skills a different data set.

Using the data from Exercise 1, try adding the geom\_smooth lines.

Answers on the next page.

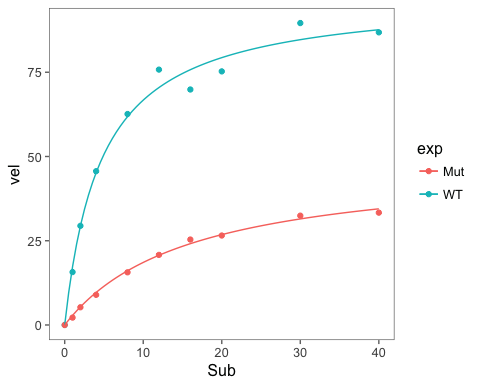
### Answers for Exercise 1

Create the plot object

enz\_plot\_2 <- ggplot(data=data\_both,   
 aes(x=Sub,   
 y=vel,   
 colour = exp)) +   
 geom\_point()

Plot with the lines.

enz\_plot\_2 + geom\_smooth(data = subset(data\_both, exp=="WT"),  
 method = "nls",   
 method.args = list(formula = y ~ Vmax \* x / (Km + x),   
 start = list(Vmax = 75, Km = 10)),  
 se = F, size = 0.5) +   
 geom\_smooth(data = subset(data\_both, exp=="Mut"),  
 method = "nls",   
 method.args = list(formula = y ~ Vmax \* x / (Km + x),   
 start = list(Vmax = 50, Km = 10)),  
 se = F, size = 0.5) +  
 theme\_few()



#### Extract Km and Vmax

This code is illustrated in the third code demo. The graphical output is an essential step and very useful. However, it's difficult to extract the numbers out of the plot. Thus, it's best to do this separately.

First, we create our equation which is based on the Machealis Menton enzymatic equation. We do that using the formula() function.

MMcurve<-formula(v~Vmax\*S/(Km+S))

Second, we generate a subset of our data which is from the "wild type" enzyme using the subset function.

WT <- subset(enzdata, Enz=="WT")

Third, we create the fit using the nls() function.

fitWT <- nls(MMcurve, WT, start=list(Vmax=50,Km=0.2))  
summary(fitWT)

##   
## Formula: v ~ Vmax \* S/(Km + S)  
##   
## Parameters:  
## Estimate Std. Error t value Pr(>|t|)   
## Vmax 66.959939 1.218157 54.97 8.91e-15 \*\*\*  
## Km 0.157924 0.009002 17.54 2.17e-09 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 1.457 on 11 degrees of freedom  
##   
## Number of iterations to convergence: 6   
## Achieved convergence tolerance: 1.218e-06

Finally, we extract our Km and Vmax constants from the fit.

Vmax\_WT <- summary(fitWT)$coefficients[1]  
Km\_WT <- summary(fitWT)$coefficients[2]  
paste("Vmax: ", round(Vmax\_WT, 1))

## [1] "Vmax: 67"

paste("Km: ", round(Km\_WT, 3))

## [1] "Km: 0.158"

### Exercise 3

Please adapt the code above to: 1. Generate a subset of our data for H297F mutant enzyme using the subset function. 2. Fit the data using the nls() function. 3. Extract the Km and Vmax constants for the H297F mutant.

Try to work this out before you turn the page to see the answers.

### Answers for Exercise 3

To create a separate data.frame with mutated enzyme data. Use the subset() function.

mutant <- subset(enzdata, Enz=="H297F")

To fit the Michaelis Menton curve and show the summary statistics.

fit\_mut <- nls(MMcurve, mutant, start=list(Vmax=50,Km=0.2))  
summary(fit\_mut)

##   
## Formula: v ~ Vmax \* S/(Km + S)  
##   
## Parameters:  
## Estimate Std. Error t value Pr(>|t|)   
## Vmax 55.478915 0.946051 58.64 1.65e-09 \*\*\*  
## Km 0.165818 0.009766 16.98 2.67e-06 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 0.9594 on 6 degrees of freedom  
##   
## Number of iterations to convergence: 5   
## Achieved convergence tolerance: 1.906e-06

To extract and show the Km and Vmax of the mutated enzyme.

Vmax\_mut <- summary(fit\_mut)$coefficients[1]  
Km\_mut <- summary(fit\_mut)$coefficients[2]  
paste("Vmax of H297F mutant: ", round(Vmax\_mut, 1))

## [1] "Vmax of H297F mutant: 55.5"

paste("Km of H297F mutant: ", round(Km\_mut, 3))

## [1] "Km of H297F mutant: 0.166"

### Exercise 4

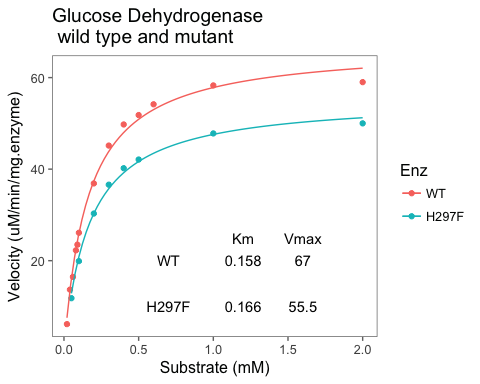
Using what you learned in Module 2, can you add the Km and Vmax of the wild type and mutated enzyme to the enz\_plot\_lines object? It would require use of the annotate() function.

Try to work this out before you turn the page to see the answers.

### Answers for Exercise 4

Here is the code that you would need for this. It's possible to make loops to do this if you wanted.

enz\_plot\_lines +   
 annotate(geom="text", x = 1.2, y = 25, label = "Km") +  
 annotate(geom="text", x = 1.6, y = 25, label = "Vmax") +  
 annotate(geom="text", x = 0.7, y = 20, label = "WT") +  
 annotate(geom="text", x = 0.7, y = 10, label = "H297F") +  
  
 annotate(geom="text", x = 1.2, y = 20, label = round(Km\_WT, 3)) +  
 annotate(geom="text", x = 1.6, y = 20, label = round(Vmax\_WT, 1)) +  
 annotate(geom="text", x = 1.2, y = 10, label = round(Km\_mut, 3)) +  
 annotate(geom="text", x = 1.6, y = 10, label = round(Vmax\_mut, 1))



### Review what we have learned

* We have graphed an enzyme kinetics plot using ggplot2.
  + geom\_point( )
  + enz\_plot + facet\_wrap(~Enz)
* We have used geom\_smooth() with a subset of our data and the nls method
  + enz\_plot + geom\_smooth(data = subset(enzdata, Enz=="WT"), method = "nls", method.args = list(formula = y ~ Vmax \* x / (Km + x), start = list(Vmax = 50, Km = 0.2)), se = F, size = 0.5)
* We have used the nls() function to calculate Vmax and Km values
  + summary(fitWT)$coefficients[1] gives us the Vmax value
  + summary(fitWT)$coefficients[2] gives us the Km value

### Resources

* Try a script on R for Biochemists that creates an LD50 plot <http://rforbiochemists.blogspot.ie/2015/06/using-ggplot-to-draw-ld50-graph.html>
* Documentation about the geom\_smooth( ) function <http://ggplot2.tidyverse.org/reference/geom_smooth.html>
* Documentation about the facet\_wrap( ) function <http://ggplot2.tidyverse.org/reference/facet_wrap.html>
* Use help in R

?nls # more information inlcuding examples on the linear model function