1 Introduction

In this tutorial we will explore the ways of presenting simple kinetic data and calcualting kinetic parameters from that data. Starting with the ideal case, we will then demonstrate the effect of experimental errors on the determination of the kinetic parameters using each of the methods.

1.1 The ideal model

A simple kinetic reaction follows the formula $V = \frac{V_{max} \cdot [S]}{K_m + [S]}$

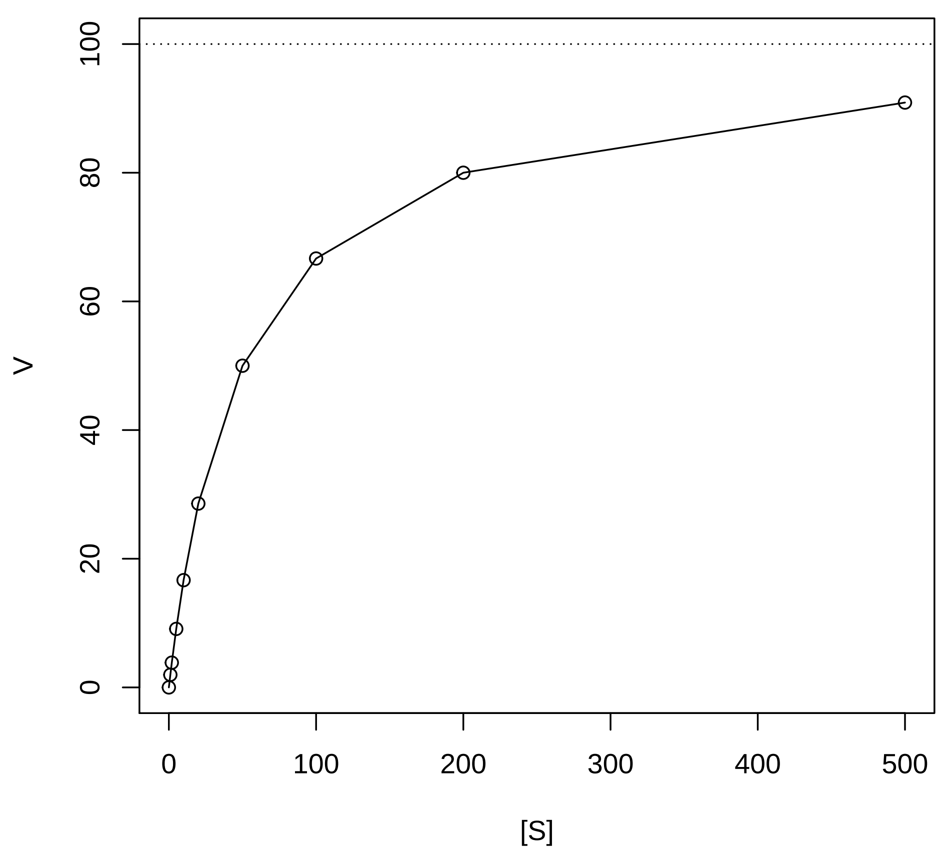
We can set the V_{max} and K_m to appropriate values and calculate the observed velocity V at each value of the substrate concentration $[S]$.

With V_{max} set to 100 and K_m to 50 we can calculate the effective rate for values of the substrate concentration from 0 to 500 using a standard serial dilution.

```
> vmax <- 100
> km <- 50
> substrate <- c(0,1, 2, 5, 10, 20, 50, 100, 200 ,500)
> velocity <- vmax*substrate/(substrate+km)
```

The velocity can then be plotted in several ways. The simplest way is to plot the rate against the substrate concentration. The dotted line indicates V_{max}

```
> plot(velocity~substrate, xlab="[S]", ylab="V", type='l',ylim=c(0,100))
> points(velocity~substrate)
> abline(h=100, lty=3)
```

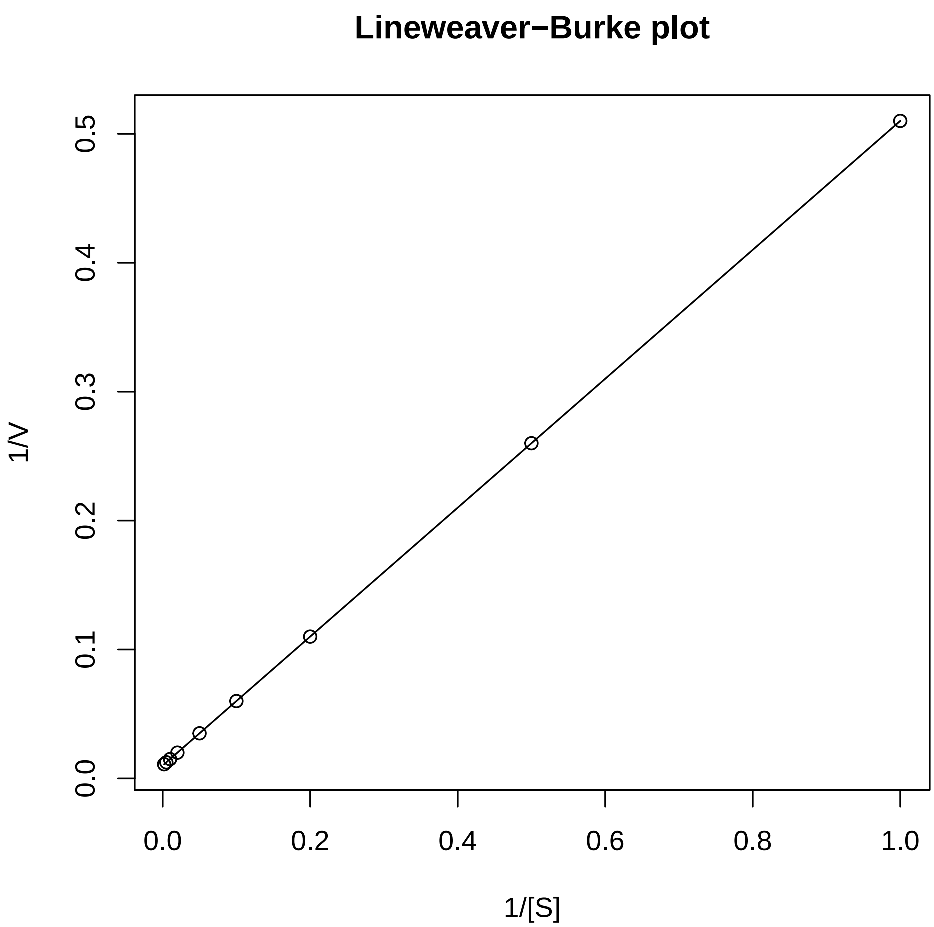


1.1.1 The Lineweaver-Burke plot

It is hard to estimate the Km or $Vmax$ from such plots so typically we transform the data to a straight line, fit the straight line and use this to determine the K_m and V_{max}.

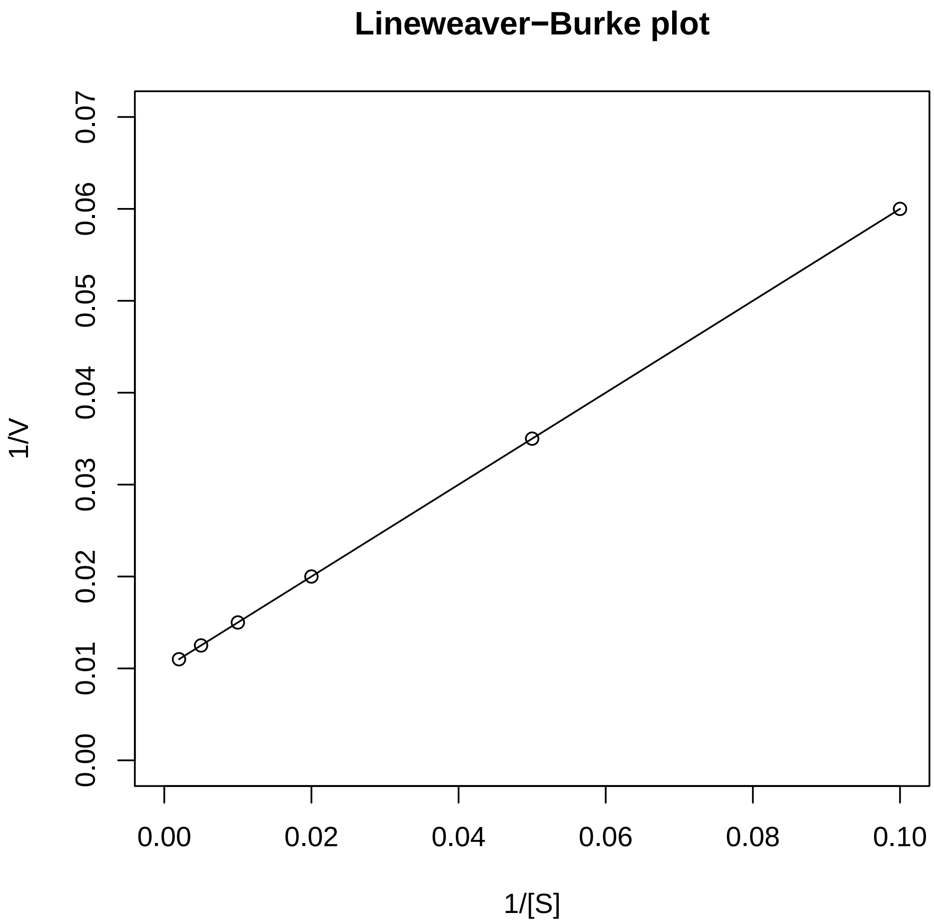
Traditionally students would use the Lineweaver-Burke plot. This is the reciprocalof the previous plot.

```
> plot(1/substrate[-1],1/velocity[-1], xlab="1/[S]", ylab="1/V", type='l',
+      main="Lineweaver-Burke plot")
> points(1/substrate[-1],1/velocity[-1])
```



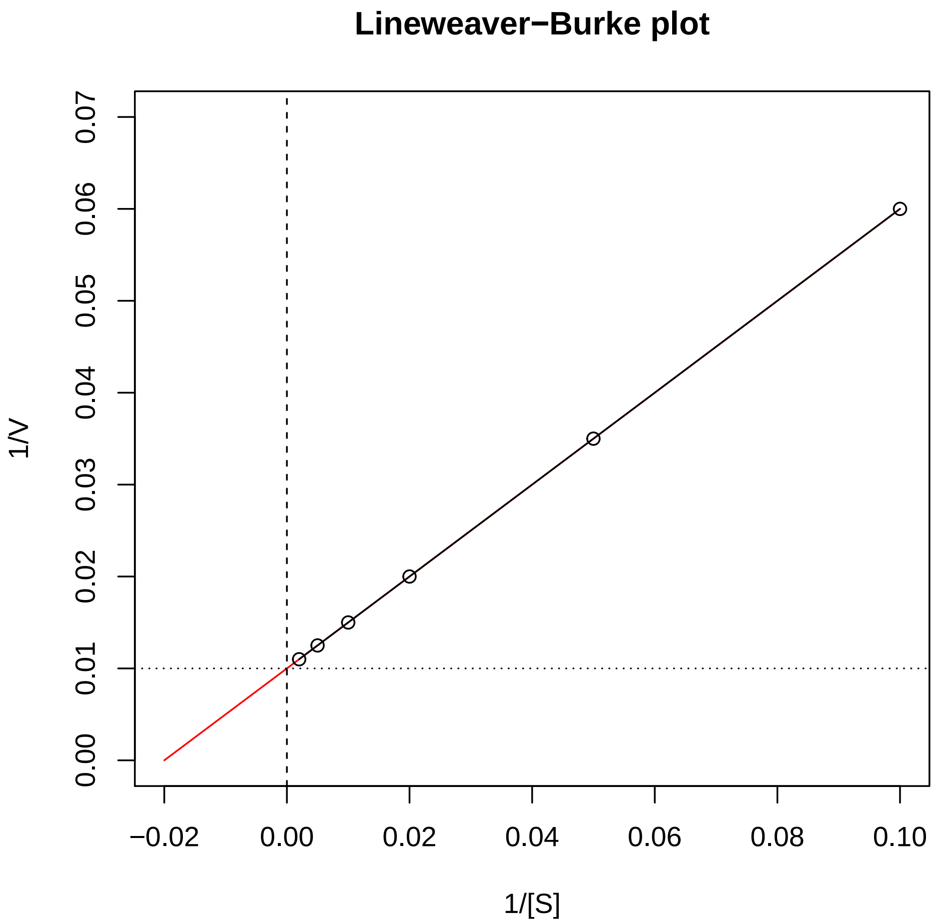
Zooming in on the Y axis:

```
> plot(1/substrate[5:10],1/velocity[5:10], xlab="1/[S]", ylab="1/V", type='l',
+      xlim=c(0,0.1), ylim=c(0,0.07),main="Lineweaver-Burke plot")
> points(1/substrate[5:10],1/velocity[5:10])
```



Extending theline to intercept the X-axis gives $1/V_{max}$ as the Y intercept and $-1/K_m$ as the X intercept.

```
> plot(c(-1/km, 1/substrate[5]), c(0, 1/velocity[5]), col=2, type='l', xlab="1/[S]",
+      ylab="1/V",xlim=c(-1/km,0.1), ylim=c(0,0.07),main="Lineweaver-Burke plot")
> lines(1/substrate[5:10],1/velocity[5:10] )
> points(1/substrate[5:10],1/velocity[5:10] )
> abline(v=0,lty=2)
> abline(h=1/vmax, lty=3)
```

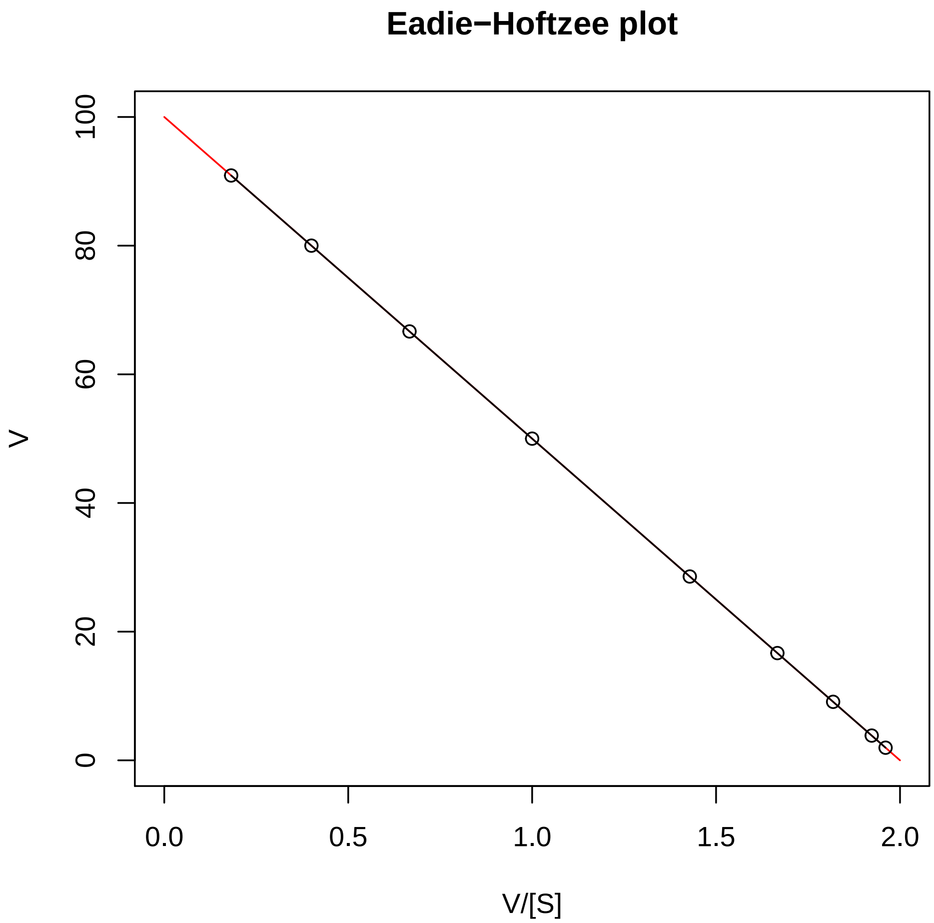


1.1.2 The Eadie-Hoftzee plot

Alternative transformations which are less susceptibel to errors are the Eadie-Hoftzee plot and the Haynes-Woolf plot.

Eadie-Hoftzee plots the velocity against the velocity divided by the substrate concentration. This gives a straight line of the form $v = -K_m \frac{v}{[S]} + V_{max}$ where v is the y-axis, $\frac{v}{[S]}$ is the x-axis, the intercept is V_{max} and the gradient is $-K_m$. Again, the line can be extrapolated to give the intercept as indicated by the red line.

```
> plot(c(0,vmax/km),c(vmax,0), col=2, xlab="V/[S]", ylab="V", type='l', xlim=c(0,2),
+      ylim=c(0,100), main="Eadie-Hoftzee plot")
> lines(velocity[-1]/substrate[-1],velocity[-1] )
> points(velocity[-1]/substrate[-1],velocity[-1] )
```

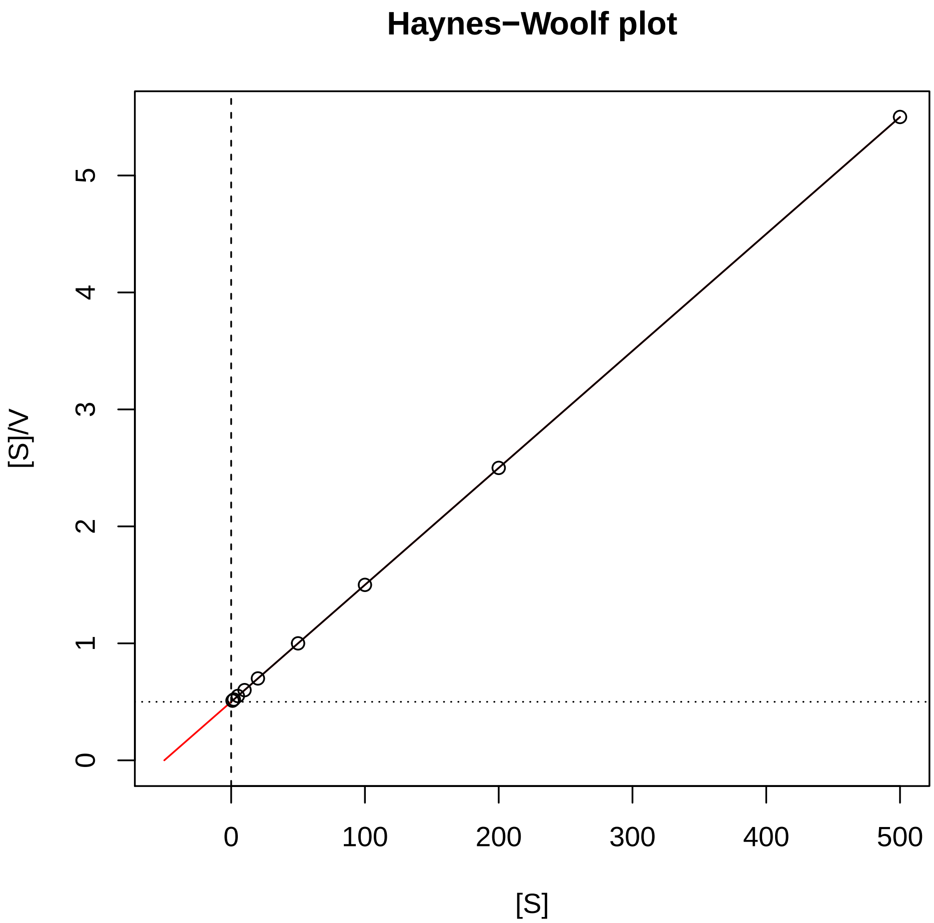


1.1.3 The Haynes-Woolf plot

The Haynes-Woolf plot is possibly the most accurate of the linear transformations. Dividing the substrate concentration by the velocity, and plotting this against the substrate concentration gives a plot which looks superficially like a Lineweaver-Burke plot but does not fall prey to the same reciprocal error scaling.

The graph forms a straight line according to the equation $\frac{[S]}{v} = \frac{1}{V_{max}}[S] + \frac{K_m}{V_{max}}$. Like the Lineweaver-Burke plot, the line can be extended through the y-axis to intercept the x-axis at $-K_m$, the y-axis intercept being at $\frac{K_m}{V_{max}}$.

```
> plot(c(-km,substrate[10]),c(0,substrate[10]/velocity[10]), col=2, xlab="[S]",
+      ylab="[S]/V", type='l',  main="Haynes-Woolf plot")
> lines(substrate[-1],substrate[-1]/velocity[-1] )
> points(substrate[-1],substrate[-1]/velocity[-1] )
> abline(v=0, lty=2)
> abline(h=km/vmax,lty=3)
```



With modern computers, fitting the experimental data directly is a good idea as the transformations amplify errors in the data.

1.2 Introducing experimental errors

Errors can be introduced into an experiment in many ways and fall into several categories. We will include fixed errors (which may be found in for example machine reading variation) and proprotional errors such as those found in pipetting or errors in fixed time measurement when measuring rates.

We will implement these using the jitter() command in a function that will return a set of experimental results with error added.

```
> fixederror <- velocity-jitter(velocity, factor=2)
> variableerror <-velocity*(velocity-jitter(velocity, factor=0.2))
```

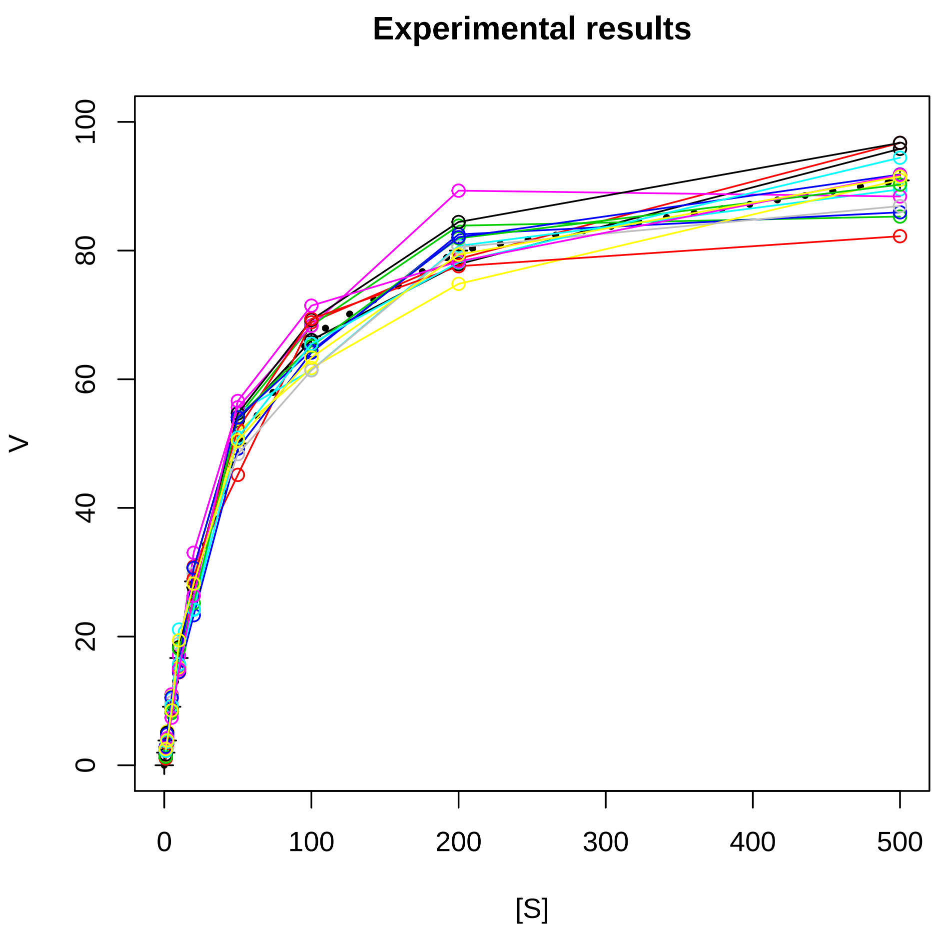
We can combine these errors and build a table of experimental values. Each time jitter() is called it produces a new random distribution of errors. These values will be saved in a table of substrate (column 1) and velocity measurement (column 2) with the experiment number in columm 3. The velocity calculations will be performed in a function doexperiment which takes as parameters V_{max}, K_m, and the substrate concentrations.

```
> doexperiment <- function (vm, k, subs){
+   # add some proportional error to the substrate for concentration errors.
+   errsubs <- subs+subs*(subs-jitter(subs))
+   # calculate the velocity that should be observed
+   vobs <- vm*(errsubs/(k+errsubs))
```

```
+   # add measurement (fixed) and time errors (proportional) to the observed value
+   errvobs <-jitter(vobs, factor=2)+vobs*(vobs-jitter(vobs, factor=0.2))
+   # return the experimental values
+   errvobs
+ }

> experiment <-cbind(substrate[-1],doexperiment(vmax,km,substrate[-1]),1)
> for (e in 2:15){
+   experiment <- rbind(experiment,
+                       cbind(substrate[-1], doexperiment(vmax,km,substrate[-1]),e))
+ }
```

The latter line is a loop that gives us many replicates as we want. In this case we repeat 15 times.

```
> plot(substrate, velocity, type='l', lty=3, lwd=4, ylim=c(0,100),
+      main='Experimental results', xlab='[S]', ylab='V' )
> points(substrate, velocity, pch=3)
> for (p in 1:15) {
+   points(experiment[experiment[,3]==p, 1],
+          experiment[experiment[,3]==p, 2], col=p )
+   lines(experiment[experiment[,3]==p, 1],
+         experiment[experiment[,3]==p, 2], col=p )
+ }
```



1.2.1 Fitting the data

To fit the data we can use the drc package. (Note: these commands do not run automatically. You will need to run the install once and the library command each time you start R.)

```
> install.packages("drc")
```

```
> library(drc)
```

```
> library(drc)
```

The drm method can be used to fit the data to a 2-parameter Michaelis-Menten equation. It takes a data frame as argument and fits velocity against substrate concentration. First we can test hwo well it works against ideal data:

```
> ideal<-as.data.frame(cbind(substrate,velocity))
> # create the data frame
> colnames(ideal) <- c('substrate', 'velocity')
> # set the column names
> fit<-drm(velocity~substrate, data=ideal, fct=MM.2())
> #fit the data to the equation
```

The results are found in fit$coefficients. The first value is the fit to V_{max} and the second is K_m.

```
> fit$coefficients
```

```
d:(Intercept) e:(Intercept)
    100.00001      50.00002
```

The ideal data has, as expected given a perfect fit.

We can fit the data for each experiment and see how the results compare to the real result.

```
> # first convert our experimental data to a data frame
> experiment<-as.data.frame(experiment)
> colnames(experiment) <- c('substrate','velocity', 'expr')
> data<-subset(experiment, expr==1)
> fit<-drm(velocity ~substrate, data= data, fct=MM.2())
> mmres <- as.data.frame(cbind(fit$coefficients[1],fit$coefficients[2]))
> mmres
```

```
                    V1       V2
d:(Intercept) 102.9464 52.40318
```

This is not as good a fit. We can complete the rest of the fits with a loop and see how they match to the ideal fit.

```
> for(e in 2:15) {
+   data<-subset(experiment, expr==e)
+ fit<-drm(velocity ~substrate, data= data, fct=MM.2())
+   mmres <- rbind(mmres, fit$coefficients)
+ }
> colnames(mmres) <-c('Vmax', 'Km')
> mean(mmres[,'Vmax'])
```

```
[1] 99.82326
```

```
> sd(mmres[,'Vmax'])
```

```
[1] 3.87629
```

```
> mean(mmres[,'Km'])
```

```
[1] 49.04345
```

```
> sd(mmres[,'Km'])
```

```
[1] 5.412747
```

1.2.2 Fitting to the Lineweaver-Burke plot

To fit the Lineweaver-Burke plot (and any of the other linear plots) we first need to calculate the x and y values for each datapoint.

```
> x <- 1/substrate[-1]
> y <- 1/velocity[-1]
```

These can be fitted to a straight line with the lm function.

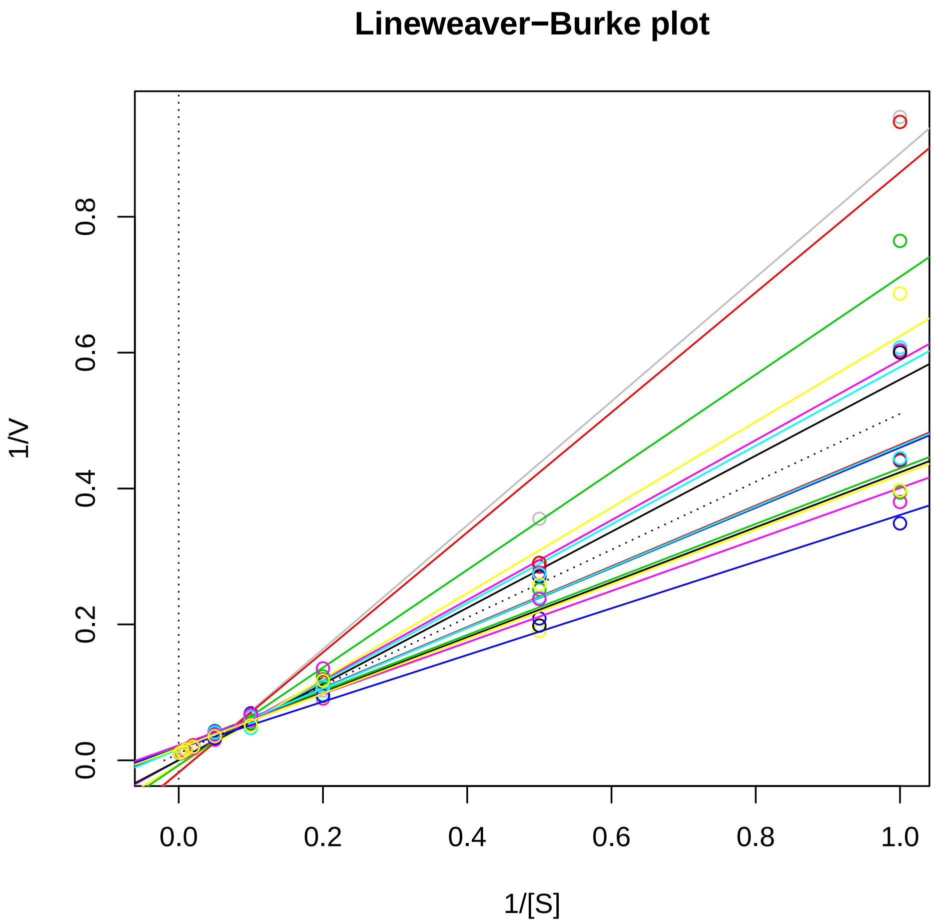
```
> fit <- lm(y~x)
> fit

Call:
lm(formula = y ~ x)

Coefficients:
(Intercept)            x
       0.01         0.50
```

The coefficients are the intercept and gradient. V_{max} is the reciprocal of the intercept. K_m is the gradiend divided by the intercept.

```
> #calculate Vmax
> 1/fit$coefficients[1]

(Intercept)
        100

> #calculate Km
> fit$coefficients[2]/fit$coefficients[1]

 x
50
```

Again, we will repeat this for the experimental data. First we will plot the transformed data, then fit a straight line to each data set.

```
> plot(c(-1/km, 1/substrate[2]), c(0,1/velocity[2]),
+      ylim=c(0, 1/min(experiment$velocity)), type='l', lty=3,
+      xlab='1/[S]', ylab='1/V', main='Lineweaver-Burke plot')
>   points(1/experiment$substrate[experiment$expr==1],
+          1/experiment$velocity[experiment$expr==1], col=1)
>   x<-1/experiment$substrate[experiment$expr==1]
>   y<-1/experiment$velocity[experiment$expr==1]
>   fit<-lm(y~x)
>   abline(fit$coefficients[1],fit$coefficients[2],col=1)
> abline(v=0, lty=3)
> lbres<-as.data.frame(cbind(1/fit$coefficients[1],
+                            fit$coefficients[2]/fit$coefficients[1]))
> colnames(lbres) <-c('Vmax','Km')
> for (e in 2:15){
+   points(1/experiment$substrate[experiment$expr==e],
+          1/experiment$velocity[experiment$expr==e], col=e)
+   x<-1/experiment$substrate[experiment$expr==e]
+   y<-1/experiment$velocity[experiment$expr==e]
+   fit<-lm(y~x)
+   abline(fit$coefficients[1],fit$coefficients[2],col=e)
+   lbres<-rbind(lbres,c(1/fit$coefficients[1],
+                        fit$coefficients[2]/fit$coefficients[1]))
+ }
```



The estimates from the experimental data are significantly worse than a direct fit.

```
> mean(lbres$Vmax)

[1] -7.721208

> sd(lbres$Vmax)

[1] 1751.666

> mean(lbres$Km)

[1] -15.89512

> sd(lbres$Km)

[1] 1009.351
```

1.2.3 Fitting to the Eadie-Hoftzee plot

To fit the Eadie-Hoftzee we first need to calculate the x and y values for each datapoint.

```
> x <- velocity[-1]/substrate[-1]
> y <- velocity[-1]
```

These can be fitted to a straight line with the lm function.

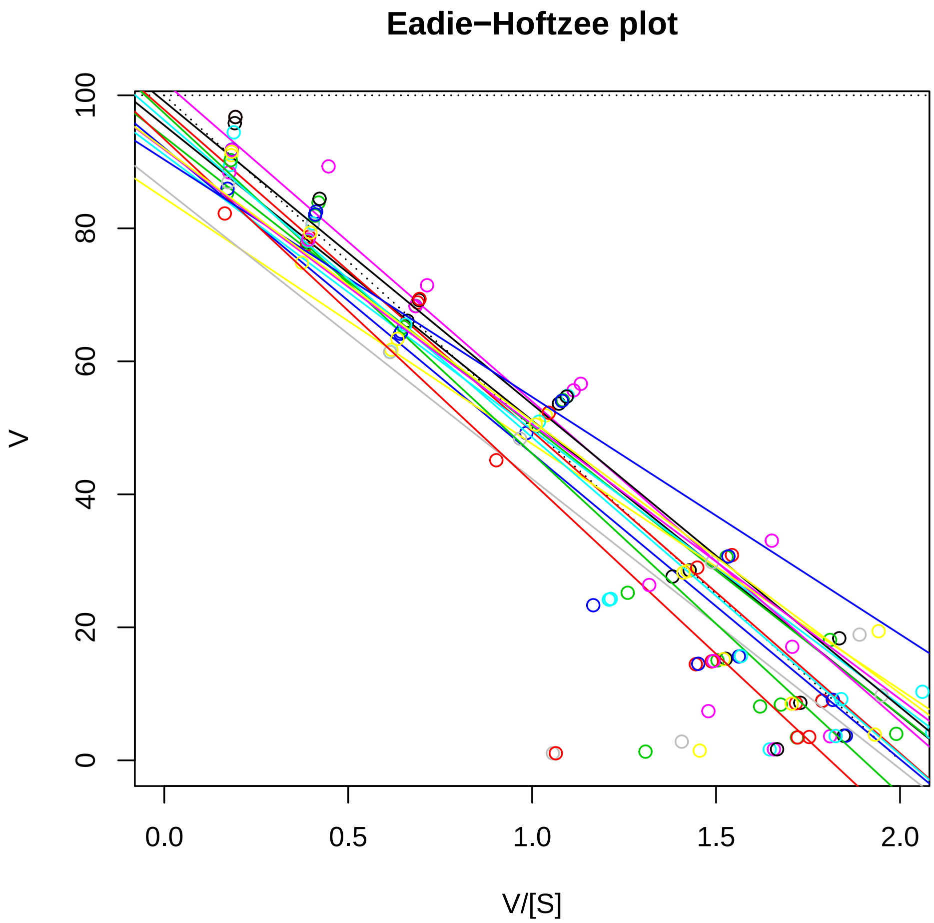
```
> fit <- lm(y~x)
> fit

Call:
lm(formula = y ~ x)

Coefficients:
(Intercept)            x
        100          -50
```

The coefficients are the intercept and gradient. V_{max} is the intercept. K_m is the gradient.

```
> #calculate Vmax
> fit$coefficients[1]

(Intercept)
        100

> #calculate Km
> -fit$coefficients[2]

 x
50
```

Again, we will repeat this for the experimental data. First we will plot the transformed data, then fit a straight line to each data set.

```
> plot(c(0,vmax/km), c(vmax,0), ylim=c(0, max(experiment$velocity)), type='l', lty=3,
+      xlab='V/[S]', ylab='V',main='Eadie-Hoftzee plot')
>   points(
+     experiment$velocity[experiment$expr==1]/experiment$substrate[experiment$expr==1],
+          experiment$velocity[experiment$expr==1], col=1)
>   x<-experiment$velocity[experiment$expr==1]/experiment$substrate[experiment$expr==1]
>   y<-experiment$velocity[experiment$expr==1]
>   fit<-lm(y~x)
>   abline(fit$coefficients[1],fit$coefficients[2],col=1)
> abline(h=vmax, lty=3)
> ehres<-as.data.frame(cbind(fit$coefficients[1],-fit$coefficients[2]))
> colnames(ehres) <-c('Vmax','Km')
> for (e in 2:15){
+   points(
+     experiment$velocity[experiment$expr==e]/experiment$substrate[experiment$expr==e],
+          experiment$velocity[experiment$expr==e], col=e)
+   x<-experiment$velocity[experiment$expr==e]/experiment$substrate[experiment$expr==e]
+   y<-experiment$velocity[experiment$expr==e]
+   fit<-lm(y~x)
+   abline(fit$coefficients[1],fit$coefficients[2],col=e)
+
+   ehres<-rbind(ehres,c(fit$coefficients[1],-fit$coefficients[2]))
+ }
```



The estimates from the experimental data are significantly better than the Lineweaver-Burke plot.

```
> mean(ehres$Vmax)
```

```
[1] 93.53072
```

```
> sd(ehres$Vmax)
```

```
[1] 4.701549
```

```
> mean(ehres$Km)
```

```
[1] 44.41229
```

```
> sd(ehres$Km)
```

```
[1] 4.670773
```

1.2.4 Fitting to the Haynes-Woolf plot

To fit the Haynes-Woolf plot we first need to calculate the x and y values for each datapoint.

```
> x <- substrate[-1]
> y <- substrate[-1]/velocity[-1]
```

These can be fitted to a straight line with the lm function.

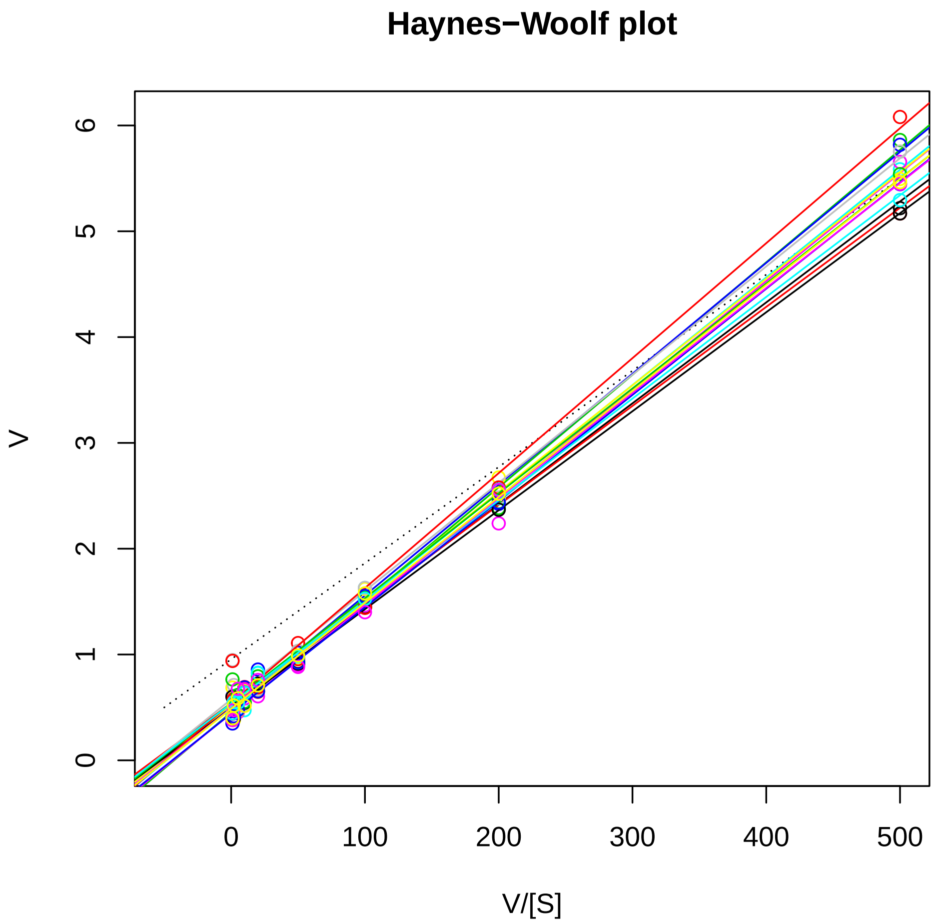
```
> fit <- lm(y~x)
> fit

Call:
lm(formula = y ~ x)

Coefficients:
(Intercept)            x
       0.50         0.01
```

The coefficients are the intercept and gradient. $\frac{K_m}{V_{max}}$ is the y-intercept. $-K_m$ is the x-intercept.

```
> #calculate Vmax
> 1/fit$coefficients[2]

  x
100

> #calculate Km
> fit$coefficient[1]/fit$coefficients[2]

(Intercept)
         50
```

Again, we will repeat this for the experimental data. First we will plot the transformed data, then fit a straight line to each data set.

```
> plot(c(-km,max(experiment$substrate)), c(km/vmax, substrate[10]/velocity[10]),
+         ylim=c(0, max(experiment$substrate/experiment$velocity)), type='l', lty=3,
+      xlab='V/[S]', ylab='V',main='Haynes-Woolf plot')
>   x<-experiment$substrate[experiment$expr==1]
>   y<-experiment$substrate[experiment$expr==1]/experiment$velocity[experiment$expr==1]
>   points(x,y,col=1)
>   fit<-lm(y~x)
>   abline(fit$coefficients[1],fit$coefficients[2],col=1)
> abline(h=vmax, lty=3)
> hwres<-as.data.frame(cbind(1/fit$coefficients[2],
+                             fit$coefficients[1]/fit$coefficients[2]))
> colnames(hwres) <-c('Vmax','Km')
> for (e in 2:15){
+   x<-experiment$substrate[experiment$expr==e]
+   y<-experiment$substrate[experiment$expr==e]/experiment$velocity[experiment$expr==e]
+   points(x,y,col=e)
+   fit<-lm(y~x)
+   abline(fit$coefficients[1],fit$coefficients[2],col=e)
+
+   hwres<-rbind(hwres,c(1/fit$coefficients[2],
+                         fit$coefficients[1]/fit$coefficients[2]))
+ }
```



The estimates from the experimental data are significantly better than the Lineweaver-Burke plot.

```
> mean(hwres$Vmax)
[1] 99.89306
> sd(hwres$Vmax)
[1] 4.373188
> mean(hwres$Km)
[1] 50.53597
> sd(hwres$Km)
[1] 4.740968
```

1.3 Conclusions

We can compare the four methods described here - three linear and one direct fitting.

```
> mmres$method<-'Direct'
> lbres$method<-'Lineweaver-Burke'
> ehres$method<-'Eadie-Hoftzee'
> hwres$method<-'Haynes-Woolf'
> results <-rbind(mmres,lbres,ehres,hwres)
> tapply(results$Vmax, results$method, mean)
```

```
         Direct     Eadie-Hoftzee     Haynes-Woolf Lineweaver-Burke
      99.823256         93.530719        99.893060        -7.721208

> tapply(results$Vmax, results$method, sd)

         Direct     Eadie-Hoftzee     Haynes-Woolf Lineweaver-Burke
       3.876290          4.701549         4.373188      1751.666404

> tapply(results$Km, results$method, mean)

         Direct     Eadie-Hoftzee     Haynes-Woolf Lineweaver-Burke
       49.04345          44.41229         50.53597        -15.89512

> tapply(results$Km, results$method, sd)

         Direct     Eadie-Hoftzee     Haynes-Woolf Lineweaver-Burke
       5.412747          4.670773         4.740968      1009.350898
```

From these results we can see that the Haynes-Woolf plot gives the closest results to the real data, very similar to the direct fitting. Eadie-Hoftzee is close. Lineweaver-Burke should not be used.