# Pharmacology practical R commands

*ABMS2*

*29th November 2018*

## Saving data from Excel

- In Excel, enter your data as follows: The first row should contain the name of your columns. In this example, we have one column called X (for ligand concentration), and one column called Y (for response), but the naming is up to you.
- Each observation is one row, with a number in the X column and a number in the Y column. It should look similar to this
- In the example drug concentration is the log drug concentration. You can either enter the log drug concentration or convert it later in R

| *X* | *Y* |
|---|---|
| | |
| −5 | 13.36 |
| −6 | 12.83 |
| … | … |

- Save your data as a .csv file. This stands for *c*omma *s*eparated *v*alues and can easily be read into R.
- Here we will use "data.csv" as the filename, but it up to you to choose. Make sure you store the file in the same directory where you open RStudio. If you don't know what directory that is, type

```
getwd()
```

## Importing data into R

To import data into R, we use the read.csv command. We create a new data frame called my_data. The arrow specifies what goes into this new data frame. In our case, it is what we read from the csv file. header=TRUE means that we use the first row of the data file as our column names

```
my_data <- read.csv("data.csv",header=TRUE)
```

You can inspect the data frame by typing

```
my_data
```

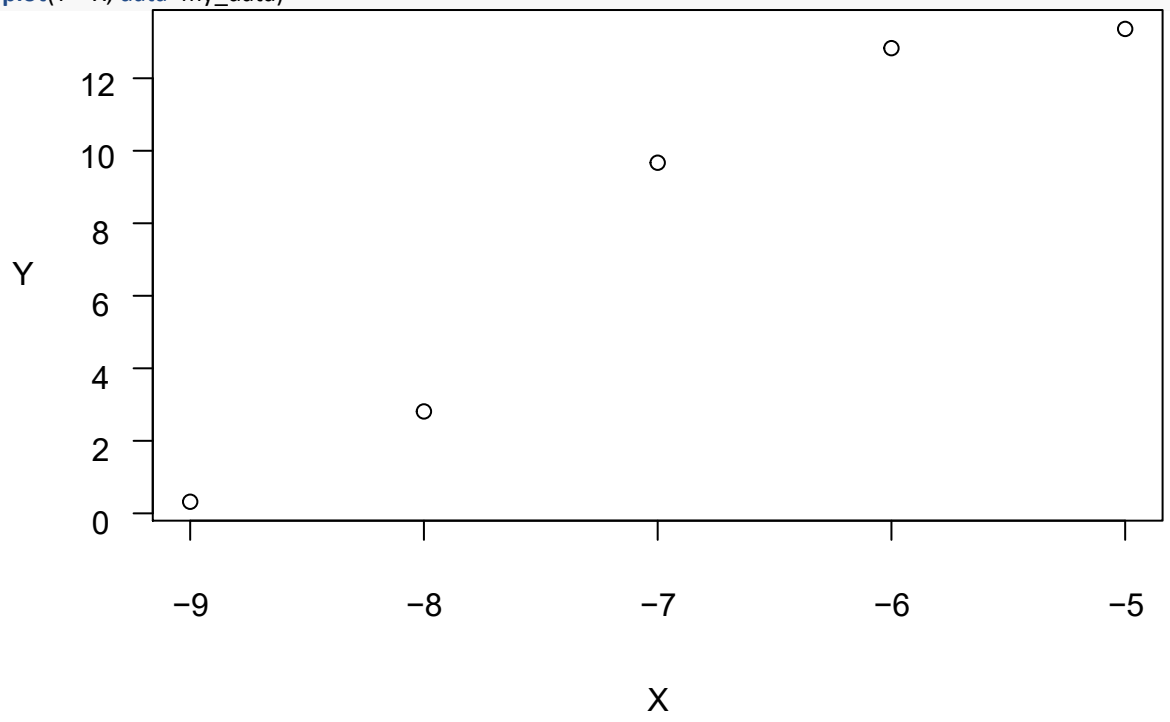
```
##        X         Y
## 1 -5 13.36
## 2 -6 12.83
## 3 -7 9.67
## 4 -8 2.81
## 5 -9 0.32
```

To convert drug concentration to log drug concentration use my_data[1]<-log10 (my_data[1])

## Plotting

In the following, we will often work with two columns in a data set. The syntax "Y ~ X, data = my_data" means we are looking at Y as a function of X for the dataset called my_data. For instance

```
plot(Y ~ X, data=my_data)
```




## Fitting a linear curve

We use the lm function. Look at the d

b / typing

```
?lm
```

In our case, we want to save the results of our fit into an object called fit

```
fit <- lm(Y~X, data=my_data)

fit
```

```
    Call
    :
## lm(formula = Y ~ X, data = my_data)
##
## Coefficients:
```

```
## (Intercept)                X
##       33.07             3.61
```

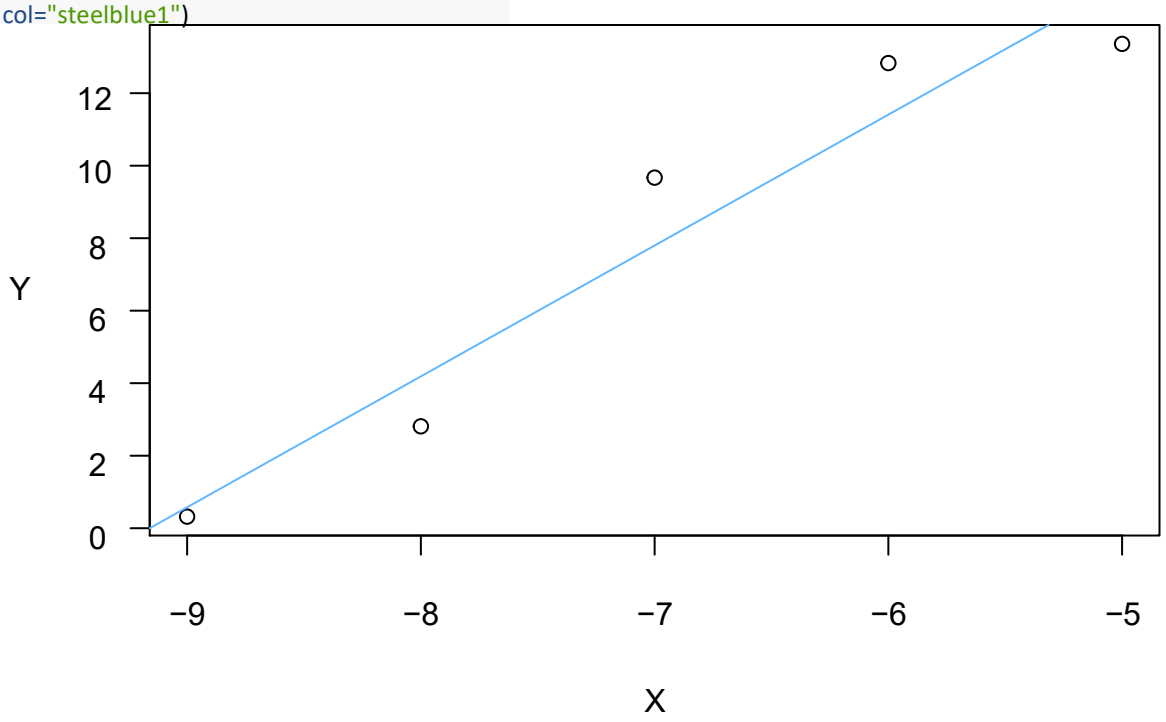
For linear models, the first parameter of the new "fit" object is the intercept, the second is the slope. We can save those to new variables called intercept and slope, and type in the name of those variables to look at them intercept<- **coef**(fit)[1] slope<- **coef**(fit)[2] intercept

```
## (Intercept)
##       33.068
slope
```

```
##        X
## 3.61
```

Finally, you can plot the line of the fit together with the original data as follows:

```
plot(Y ~ X, data=my_data) abline(fit,
col="steelblue1")
```




## Fitting a sigmoid

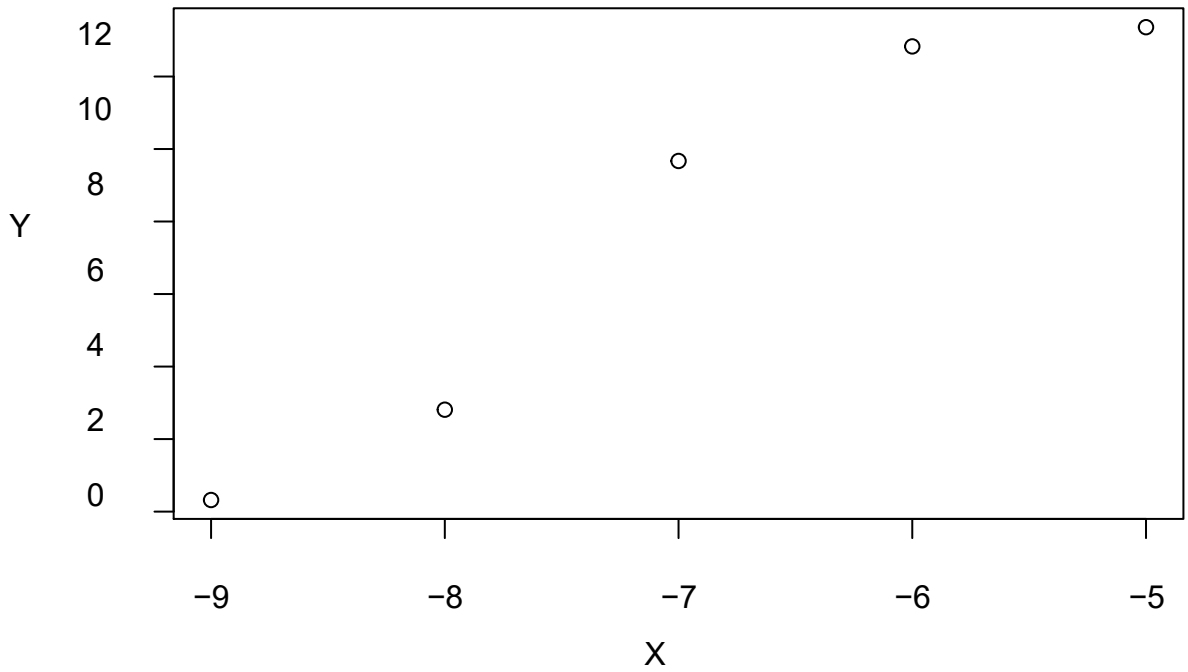
Let's say we want to fit a sigmoid instead. This is a bit more complicated, because we need to specify both the formula to fit, as well as initial values. The formula we are using is

$$Y = Max \times \frac{10^X}{EC_{50} + 10^X}$$

## Finding initial values

In order to have a good fit, you need to provide estimates for the unknown values, i.e. for Max and for EC50. Find those by plotting the dataset and inspecting the curve.

```
plot(Y~X, data=my_data)
```




In this case, I estimate a max of 14 and an EC50 of 3e-8 (*Why?, Please note that the fit uses log values but gives an absolute EC50 value*)

```
my_max <- 14  my_EC50<-

3e-8
```

Now comes the actual fitting. We use the nls function. Let's first look at what it does.

```
?nls
```

nls needs to be provided with the formula to be fitted, the dataset we are using, and initial values for the parameters to be fitted. We save the result of the fit to an object called fitmodel and look at the result:

```
fitmodel <- nls(Y~(Max * 10^X/(EC50 + 10^X)), data=my_data, start=list(Max = my_max, EC50=my_EC50))  fitmodel
```

```
## Nonlinear regression model
##             model: Y ~ (Max * 10^X/(EC50 + 10^X))
##          data: my_data
##              Max           EC50
## 1.336e+01 3.802e-08
## residual sum-of-squares: 0.005549
##
## Number of iterations to convergence: 4
## Achieved convergence tolerance: 1.853e-06
```

Again, we can extract the parameters by looking at the coefficients of fitmodel

```
params <- coef(fitmodel)
```

```
Max <- params[1]

 EC50 <- params[2]

Max
```

```
##           Max
## 13.36374
EC50
```

```
##                EC50
## 3.801898e-08
```

How can we plot that sigmoid? Let's first create a set of evenly spaced points (with spacing 0.01) between -9 and -5 (for the x axis)

```
X2 <- seq(-9,-5, 0.01)
```

Now we create Y values using our formula and the values we cound for Max and EC50

```
Y2 <- Max * 10^X2/(EC50 + 10^X2)
```

And finally, we can plot the sigmoid, in addition to the original dataset (use ?plot if you want to learn more about some of the options)

```
plot(Y2~X2, type="l", ylim=c(0,13), xlab="X", ylab="Y", col="violetred")
points(response~drug, data=my_data)
```