LC1 Fitting functions

Harm Manders, 10677186

```r
rm(list=ls()) # clear the workspace.. always a good idea before you start a new project
set.seed(321) # so we will all be talking about the same random results
setwd("/home/harm/Uni/cogmod/LC1") # set to your working dir with the files you need.
```

Q1 To begin write your first function:

$$y = e^{\frac{(.3 \cdot x)}{2}}$$

and call this function *curve*. To be clear the input should be x and it should return y. (1 point)

```r
curve <- function(x) {

  y = exp((.3* x) / 2)
  return(y)
}
```

Q2 (save your code, you need the constructed code later) (3 points)

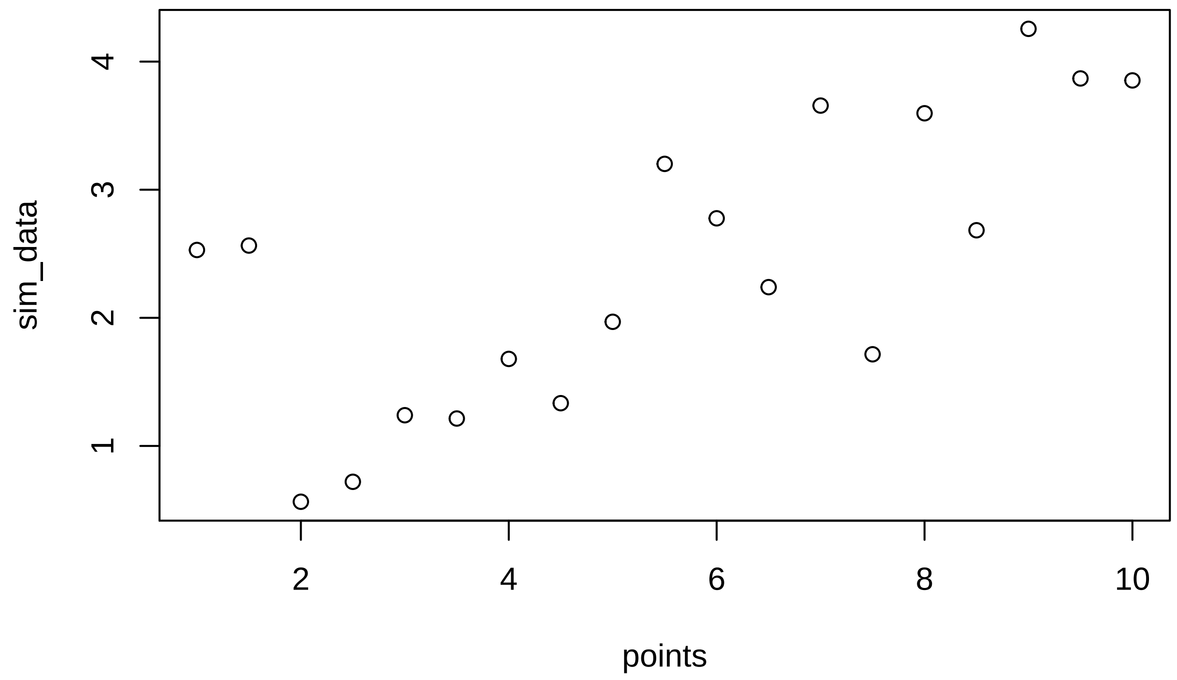
The next step is to generate some data using this function:

1. create an empty vector c() called **sim_data**
2. make and save a list called **points** with numbers between 1 and 10 in steps of .5 using **seq()**

3a. Use the numbers in **points** to generate y values using your **curve()** function and store them in **sim_data**

3b. also add noise to the **sim_data** from a uniform distribution between -1.5 to 1.5 (hunt use runif())

```r
sim_data = c()
points = seq(1, 10, 0.5)
sim_data = curve(points)
noise = runif(length(points), -1.5, 1.5)
sim_data = sim_data + noise
plot(points, sim_data)
```



```
# create a dataframe
dat_full<-as.data.frame(cbind(points,sim_data))
colnames(dat_full)<-c("x","y")
# create a subset for illstruation
dat_short<-subset(dat_full,dat_full$x<6)


## creating a simple linear function:
linear <- function(data,v){
  b1<-v
  y_hat=b1*data$x
  error = y_hat-data$y
  MSE = mean(error^2)
  return(MSE)
}
```

Q3 Generate a polynomial function to fit to the data: (1 point)

Generate a polynomial function:

$$y = b0 + b1 * x + b2 * x^2 + b3 * x^3 + b4 * x^4$$

```r
polynomial <- function(data, v) {
  x = data$x
  y_hat = v[1] + v[2]*x + v[3]*x^2 + v[4]*x^3 + v[5]*x^4
  error = y_hat - data$y
  MSE = mean(error^2)
  return(MSE)
}
```

Q4 Now fit the polynomial function to the data and compare the model fits

- Fit the model as above, and display the best fitted lines in a plots (2 points)
- Compare the MSE of the best fit for each model and indicate which is the best fitting one. Why do you think did the best fitting model won?

Hint 1: use the method="Nelder-Mead", it does not require definitions of upper and lower bounds
Hint 2: for the starting values use c c(-5,9,-4,1,.01)

Awnser

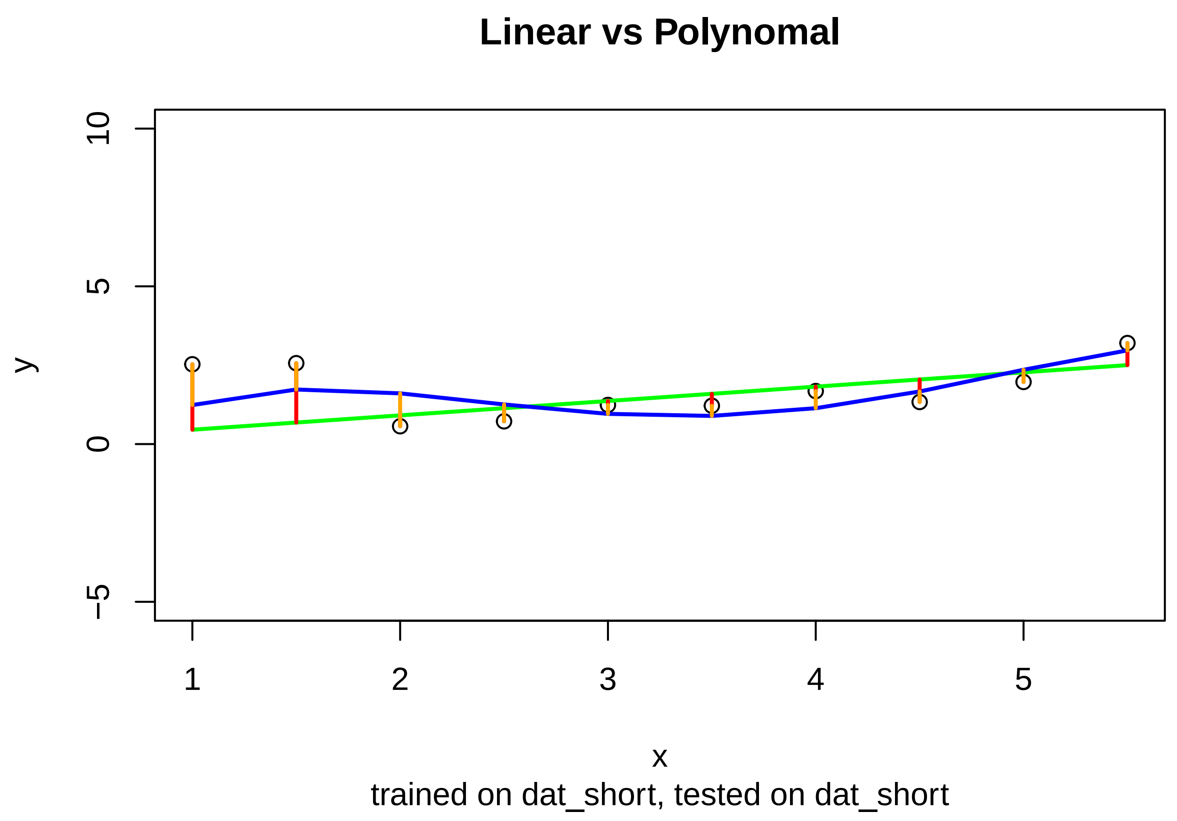
Polynomal model fits best, linear has a lower mse

```r
# Train on __dat_short__
results_lin<-optim(c(0),linear,data=dat_short,method="Brent", upper=10, lower=-10)
results_pol<-optim(c(-5,9,-4,1,.01),polynomial,data=dat_short,method="Nelder-Mead")
```

```
## [1] "MSE linear:"

## [1] 0.9408333

## [1] "MSE polynomal:"

## [1] 0.4533178

## [1] "Polynomal looks better"
```

```r
fit_data_lin<-c()
fit_data_pol<-c()

v_lin = results_lin$par
v_pol = results_pol$par
for(i in 1:nrow(dat_short)){
  fit_data_lin[i]<-v_lin[1]*dat_short$x[i]
  fit_data_pol[i]<-v_pol[1] + v_pol[2]*dat_short$x[i] + v_pol[3]*dat_short$x[i]^2 +
    v_pol[4]*dat_short$x[i]^3 + v_pol[5]*dat_short$x[i]^4
}

dat_short$error_lin<-fit_data_lin
dat_short$error_pol<-fit_data_pol

plot(dat_short$x,dat_short$y, ylim=c(-5,10), main="Linear vs Polynomal",
     sub="trained on dat_short, tested on dat_short", xlab="x", ylab="y")
x = dat_short$x
y_hat_lin = v_lin[1]*x
```

```r
y_hat_pol = v_pol[1] + v_pol[2]*x + v_pol[3]*x^2 + v_pol[4]*x^3 + v_pol[5]*x^4


lines(dat_short$x, y_hat_lin, col="green", lwd=2)
lines(dat_short$x, y_hat_pol, col="blue", lwd=2)

segments(dat_short$x, dat_short$y, dat_short$x, dat_short$error_lin, col="red",lwd=2)
segments(dat_short$x, dat_short$y, dat_short$x, dat_short$error_pol, col="orange",lwd=2)
```



Q5 check the goodness of fit for the full data set using the parameter estimates derived from the small datasets. Describe which model fits the data the best and explain why, also plot the results to illustrate your point. (2 points)

hint 1: just use same script as before but change the data file to dat_full. **hint 2**: use ylim = c(-2,12)

Awnser

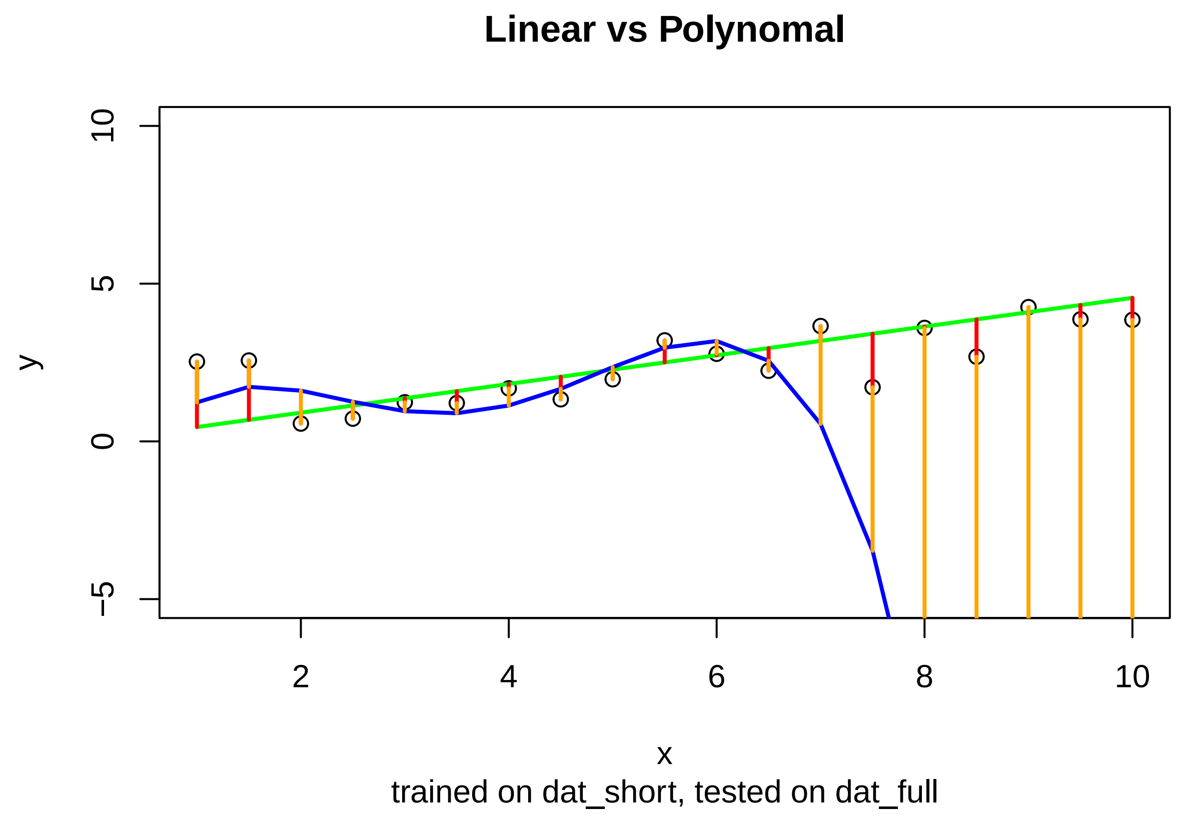
During training polynomal model looks a better "fit" But when full data is used you see that linear model has a better fit

```r
# Train on __dat_short__
results_lin<-optim(c(0),linear,data=dat_short,method="Brent", upper=10, lower=-10)
```

```
results_pol<-optim(c(-5,9,-4,1,.01),polynomial,data=dat_short,method="Nelder-Mead")

fit_data_lin<-c()
fit_data_pol<-c()

v_lin = results_lin$par
v_pol = results_pol$par
for(i in 1:nrow(dat_full)){
  fit_data_lin[i]<-v_lin[1]*dat_full$x[i]
  fit_data_pol[i]<-v_pol[1] + v_pol[2]*dat_full$x[i] + v_pol[3]*dat_full$x[i]^2 +
    v_pol[4]*dat_full$x[i]^3 + v_pol[5]*dat_full$x[i]^4
}

dat_full$error_lin<-fit_data_lin
dat_full$error_pol<-fit_data_pol


x = dat_full$x
y_hat_lin = v_lin[1]*x
y_hat_pol = v_pol[1] + v_pol[2]*x + v_pol[3]*x^2 + v_pol[4]*x^3 + v_pol[5]*x^4

full_error_lin = y_hat_lin - dat_full$y
full_error_pol = y_hat_pol - dat_full$y
full_mse_lin = mean(full_error_lin^2)
full_mse_pol = mean(full_error_pol^2)
```

```
## [1] "MSE linear:"

## [1] 0.7974191

## [1] "MSE polynomal:"

## [1] 713.7089

## [1] "Polynomal is better"
```

```
plot(dat_full$x,dat_full$y, ylim=c(-5,10), main="Linear vs Polynomal",
     sub="trained on dat_short, tested on dat_full", xlab="x", ylab="y")
lines(dat_full$x, y_hat_lin, col="green", lwd=2)
lines(dat_full$x, y_hat_pol, col="blue", lwd=2)

segments(dat_full$x, dat_full$y, dat_full$x, dat_full$error_lin, col="red",lwd=2)
segments(dat_full$x, dat_full$y, dat_full$x, dat_full$error_pol, col="orange",lwd=2)
```



Q6 Check if the exponential indeed fits better, report the fit and the parameters. Show in a plot(s) how the best fitting models fit the data points. (3 points, 2 for correct fits, 1 for plot)

hint : use method="L-BFGS-B", upper=10, lower=-10 and c(2,1) as starting values

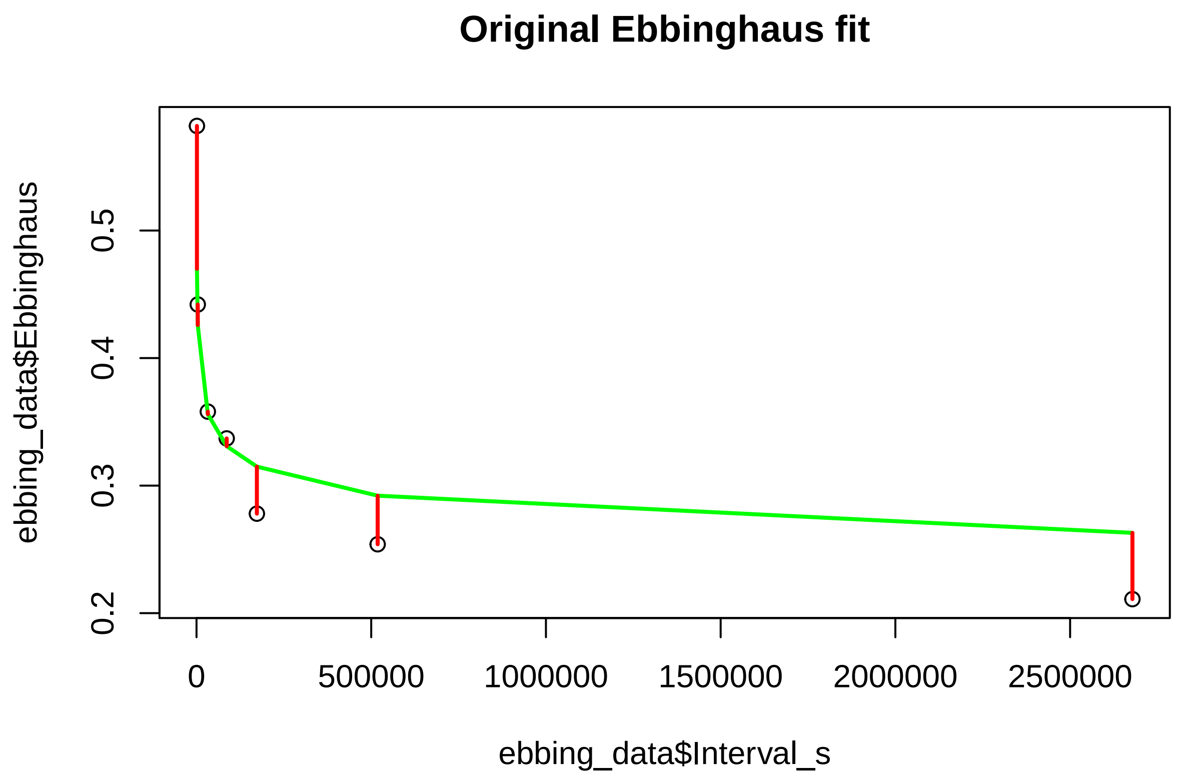
---

Awnser

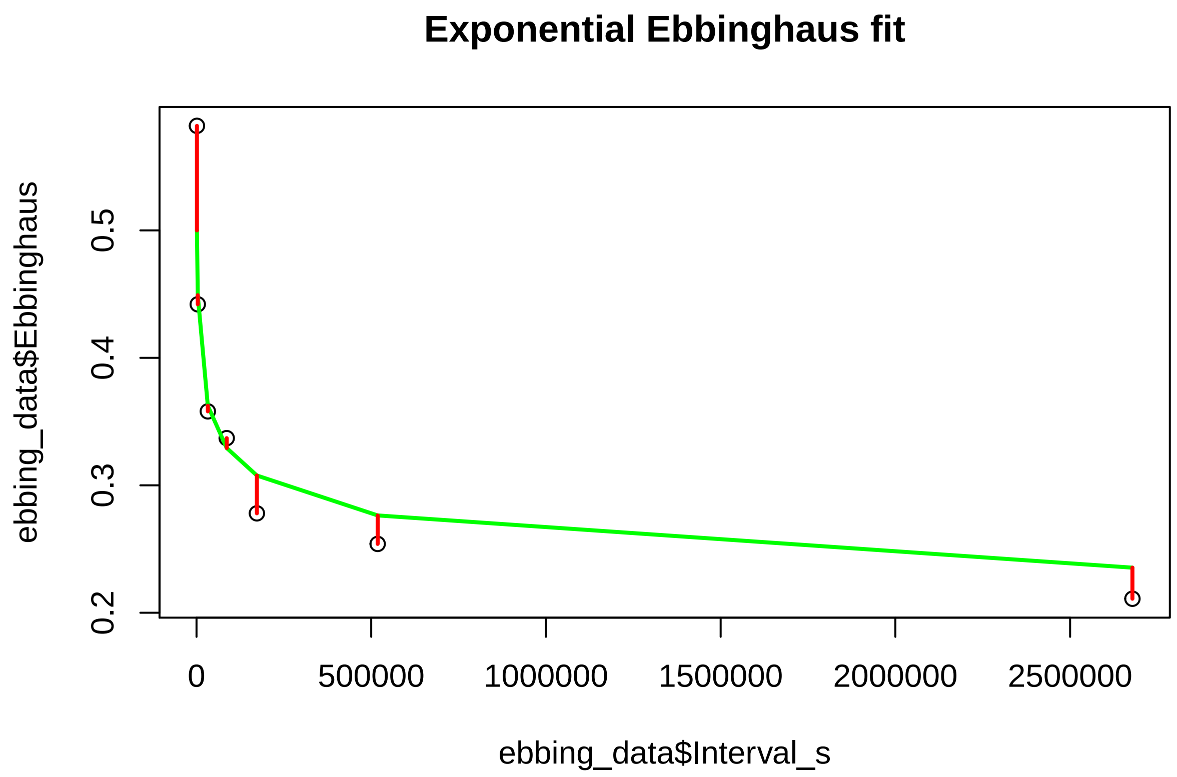
Exponential model fits the data better see plots and output values Parameters are different although for the original model fit a1 is similar to hand found result and for exponential model u1 is similar to hand found result

```r
#importing data
ebbing_data <- read.delim("Ebbinghaus.txt",header= TRUE, sep = "\t")


ebbinghaus = function(data, v) {
  x = data$Interval_s
  y_hat = v[1] / ((log(x))^v[2] + v[1])
  error = y_hat - data$Ebbinghaus
  MSE = mean(error^2)
  return(MSE)
}
```

```r
ebbinghaus_simple = function(data, v) {
  x = data$Interval_s
  y_hat = (1 + v[1] + x)^(-v[2])
  error = y_hat - data$Ebbinghaus
  MSE = mean(error^2)
  return(MSE)
}


ebbing_results<-optim(c(1.84, 1.25),ebbinghaus,data=ebbing_data,method="L-BFGS-B",
                      upper = 10, lower = -10)
ebbing_results_simple<-optim(c(1.84, 1.25),ebbinghaus_simple,data=ebbing_data,
                             method="L-BFGS-B", upper = 10, lower = -10)


fit_data<-c()
fit_data_simple<-c()

v = ebbing_results$par
vs = ebbing_results_simple$par

for(i in 1:nrow(ebbing_data)){
  fit_data[i]<- v[1] / ((log(ebbing_data$Interval_s[i]))^v[2] + v[1])
  fit_data_simple[i]<- (1 + vs[1] + ebbing_data$Interval_s[i])^(-vs[2])
}
ebbing_data$error<-fit_data
ebbing_data$error_simple<-fit_data_simple

x = ebbing_data$Interval_s
y = ebbing_data$Ebbinghaus

plot(ebbing_data$Interval_s, ebbing_data$Ebbinghaus, main="Original Ebbinghaus fit")
y_hat = v[1] / (log(ebbing_data$Interval_s)^v[2] + v[1])
lines(ebbing_data$Interval_s, y_hat, col="green", lwd=2)
segments(ebbing_data$Interval_s, ebbing_data$Ebbinghaus, ebbing_data$Interval_s,
         ebbing_data$error, col="red",lwd=2)
```



```
plot(ebbing_data$Interval_s, ebbing_data$Ebbinghaus, main="Exponential Ebbinghaus fit")
y_hat_simple = (1 + vs[1] + ebbing_data$Interval_s)^(-vs[2])
lines(ebbing_data$Interval_s, y_hat_simple, col="green", lwd=2)
segments(ebbing_data$Interval_s, ebbing_data$Ebbinghaus, ebbing_data$Interval_s,
         ebbing_data$error_simple, col="red",lwd=2)
```



```
# Exponential model MSE < original model MSE
ebbing_results_simple$value < ebbing_results$value
```

```
## [1] TRUE
```

```
# Exponential model MSE and params
ebbing_results_simple$value
```

```
## [1] 0.001261812
```

```
ebbing_results_simple$par
```

```
## [1] 1.83985777 0.09773283
```

```
# Original model MSE and params
ebbing_results$value
```

```
## [1] 0.002623261
```

```
ebbing_results$par
```

```
## [1] 10.000000  1.236981
```