Bio534 Laboratory 1: Solutions

Exercise 2.1

 # 1

```
> a <- 2^7/(2^7-1)
> b <- (1-1/2^7)^(-1)
> a == b
[1] TRUE
```

This logic answer implies that the right hand side (rhs) of the equation is equal to the left hand side (lhs): both equal 1.007874.

#2

```
> 1+0.2
[1] 1.2
> 1+0.2+ 0.2^2/2
[1] 1.22
> 1+0.2+ 0.2^2/2 + 0.2^3/6
[1] 1.221333
> exp(0.2)
[1] 1.221403
```

Find Value of e

```
> exp(1)
[1] 2.718282
```

The point of the exercise is to play with writing complicated functions, numerical approximation, and functions in R

 # 3

```
> x=1
> f <- 1/sqrt(2*pi) * exp(-x^2/2)
> f == dnorm(1)
[1] TRUE
> x=2
> f <- 1/sqrt(2*pi) * exp(-x^2/2)
> f == dnorm(2)
[1] TRUE
```

Exercise 3.1

Exercise 3.1 : Do an Apropos on sin via the Help menu, to see what it does.

```
apropos("sin")
```

```
 [1] ".__C__missing"       "as.single"           "as.single.default"

 [4] "asin"                "asinh"               "deviceIsInteractive"

 [7] "is.single"           "isIncomplete"        "missing"

[10] "missingArg"          "sin"                 "single"

[13] "sinh"                "sink"                "sink.number"
```

> help.search("sin")

Help files with alias or concept or title matching 'sin' using regular

expression matching:

AER::DutchAdvert TV and Radio Advertising Expenditures Data

AER::USInvest US Investment Data

DAAG::SP500W90 Closing Numbers for S and P 500 Index – First 100 Days of 1990

DAAG::SP500close Closing Numbers for S and P 500 Index

DAAG::bostonc Boston Housing Data - Corrected

DAAG::errorsINseveral Simulation of classical errors in x model, with multiple explanatory variables.

DAAG::errorsINx Simulate data for straight line regression, with "errors in x".

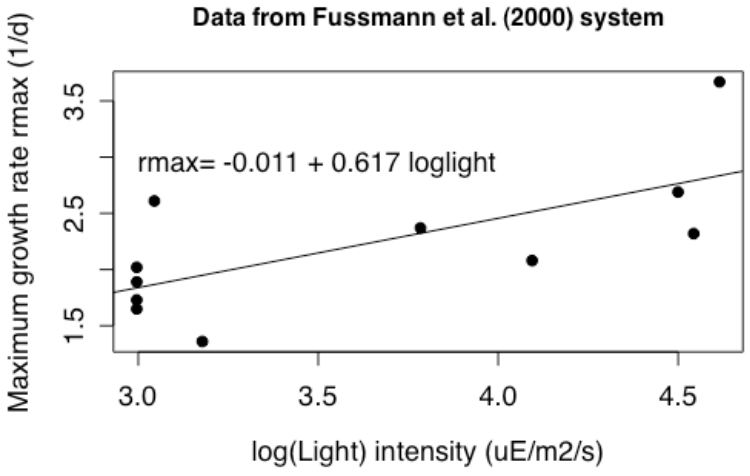
DBI::make.db.names Make R/Splus identifiers into legal SQL Identifiers

…

Apropos(“sin”) identifies all objects in the search list with the term “sin” in the name, whereas help.search(“sin”) identifies all help files that contain the term “sin”.

Exercise 5.1

```
# Intro2b.r
# Modified file from Intro2.r originally created by Ellener and Guckenhimer
# Stuart Borrett
# August 28, 2007
# ------------------------------

# --- Preliminary steps for script ---
rm(list=ls());                                          # clears the working memory
setwd("~/ZeusDocs/teaching/2007_ecomod/laboratory/Lab1-R");   # this changes the working directory

# Read in data and define variables
X=read.table('ChlorellaGrowth.txt');          # This reads in the file "ChlorellaGrowth.txt"
X=as.matrix(X);                               # this makes X a matrix type variable (but it is not necessary)
Light=X[,1]; rmax=X[,2];                      # this defines the variables Light and rmax
loglight=log(Light)

# --- Construct Plot ---
par(cex=1.5,cex.main=0.9);
plot(loglight, rmax,
        xlab="log(Light) intensity (uE/m2/s)",
        ylab="Maximum growth rate rmax (1/d)",
        pch=16,
        main="Data from Fussmann et al. (2000) system");

# --- perform statistical analysis ---
fit = lm(rmax~loglight);
summary(fit);
abline(fit);                        # draws regression line on existing plot!

# Next we get the regression equation to 'display itself' on the graph
c1=round(fit$coef[1],digits=3);       # intercept
c2=round(fit$coef[2],digits=3);       # slope

text(3,3,                           # these are the x and y coordinates on the plot where the text will be written
        paste("rmax=",c1,"+",c2,"loglight"), # this builds the text of the equation
        adj=0);                             # this tells text to left justify the text so that it starts at x=3
```



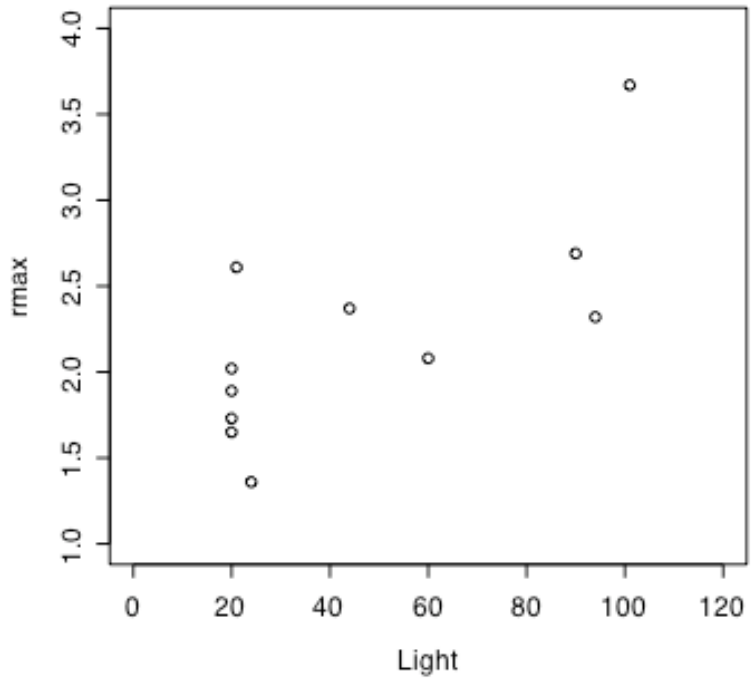
Exercise 5.2

Where "fit" is a linear model (lm) object. plot(fit) draws a series of the more common diagnostic plots to evaluate the statistical fit and appropriateness of using the linear regression data model.

Interestingly, this answer is given in the exercise.

Exercise 5.3

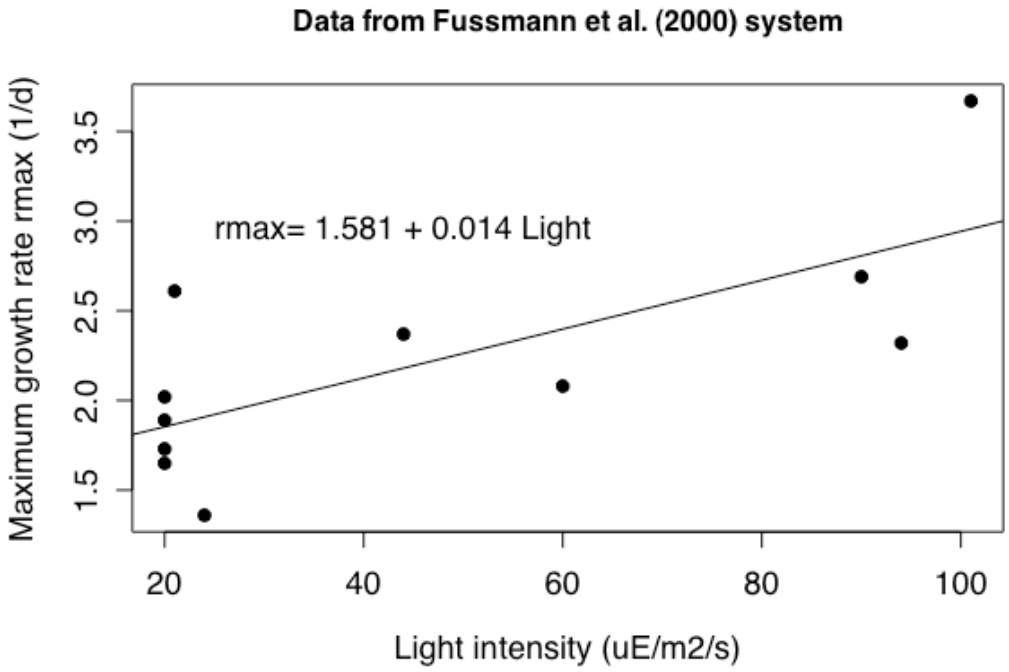
```
> plot(Light,rmax, xlim=c(0,120), ylim=c(1,4));
```

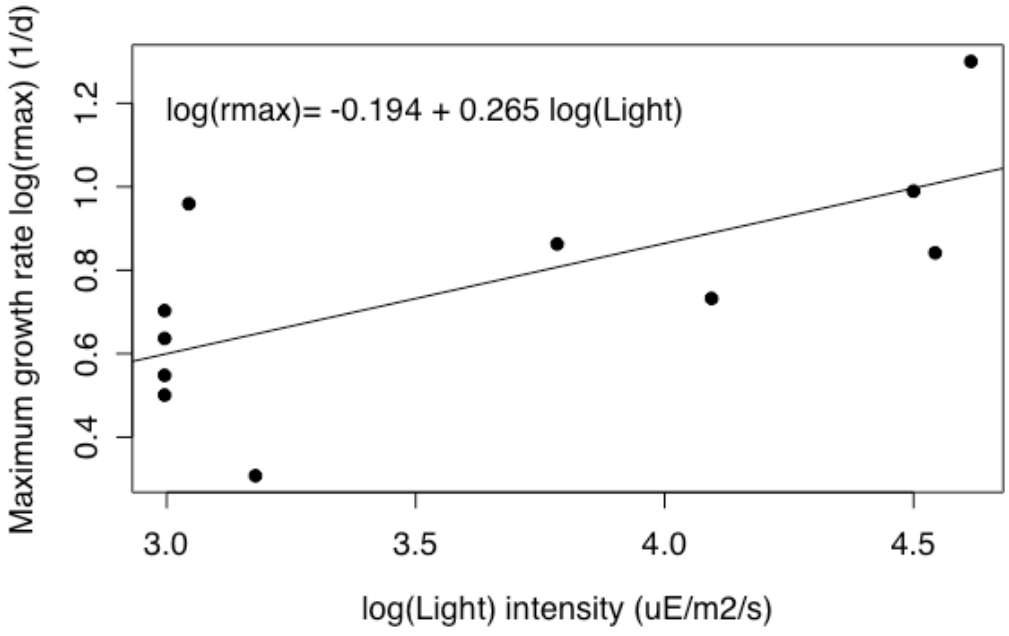


Exercise 5.4

```
# Intro2c.r
# Modified file from Intro2.r originally created by Ellener and Guckenhimer
# Stuart Borrett
# August 28, 2007
# ------------------------------

# --- Preliminary steps for script ---
rm(list=ls());                                  # clears the working memory
setwd("~/ZeusDocs/teaching/2007_ecomod/laboratory/Lab1-R");     # this changes the working directory

# Read in data and define variables
X=read.table('ChlorellaGrowth.txt');            # This reads in the file "ChlorellaGrowth.txt"
Light=X[,1]; rmax=X[,2];                        # this defines the variables Light and rmax
loglight=log(Light);    logrmax=log(rmax);

par(mfrow=c(2,1),cex=1.5,cex.main=0.9);    # divides figure region into 2 sections

# --- Construct 1st Plot ---
plot(Light, rmax,
        xlab="Light intensity (uE/m2/s)",
        ylab="Maximum growth rate rmax (1/d)",
        pch=16,
        main="Data from Fussmann et al. (2000) system");

# --- perform statistical analysis ---
fit = lm(rmax~Light);
summary(fit);
abline(fit);                    # draws regression line on existing plot!

# Next we get the regression equation to 'display itself' on the graph
c1=round(fit$coef[1],digits=3);   # intercept
c2=round(fit$coef[2],digits=3);   # slope

text(25,3,              # these are the x and y coordinates on the plot where the text will be written
        paste("rmax=",c1,"+",c2,"Light"),       # this builds the text of the equation
        adj=0);                                 # this tells text to left justify the text so that it starts at x=3


# --- Construct 2nd Plot -------
plot(loglight, logrmax,
        xlab="log(Light) intensity (uE/m2/s)",
        ylab="Maximum growth rate log(rmax) (1/d)",
        pch=16,
        );

# --- perform statistical analysis ---
fit = lm(logrmax~loglight);
summary(fit);
abline(fit);                    # draws regression line on existing plot!
```

```
# Next we get the regression equation to 'display itself' on the graph
c1=round(fit$coef[1],digits=3);   # intercept
c2=round(fit$coef[2],digits=3);   # slope

text(3,1.2,                       # these are the x and y coordinates on the plot where the text will be written
     paste("log(rmax)=",c1,"+",c2,"log(Light)"),     # this builds the text of the equation
     adj=0);                                         # this tells text to left justify the text so that it starts at x=3
```





Exercise 5.5 (optional)

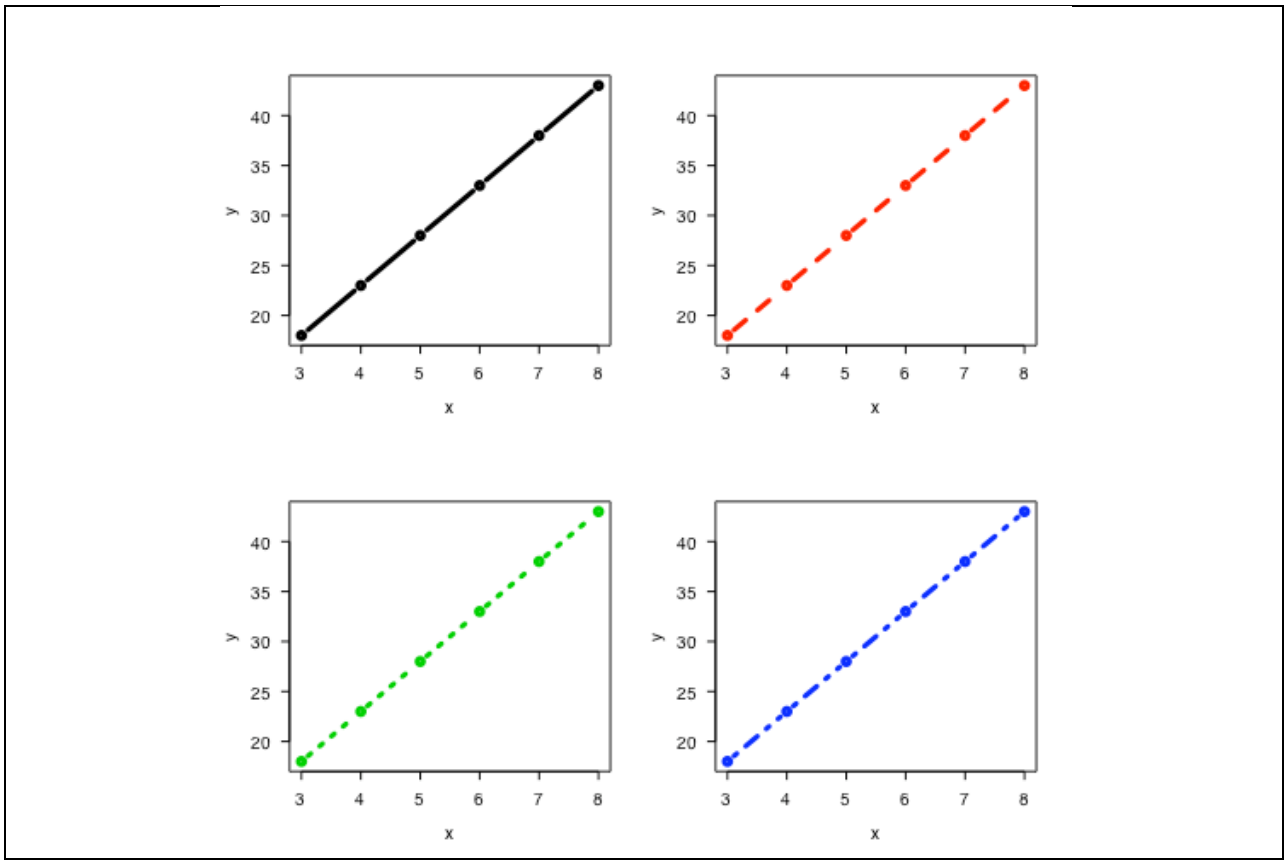
Solution #1

```
# exercise 3.5
# S.R. Borrett
# August 28, 2007
# ---------------

# Create Data
x<-3:8;
y<-5*x+3;

# Draw Plot
opar<-par(mfrow=c(2,2),las=1);
# 1
plot(x,y,type="b",lty=1,col=1,lwd=3)
     # lty = line type
     # col = line color
     # lwd = line width
#2
plot(x,y,type="b",lty=2,col=2,lwd=3)
#3
plot(x,y,type="b",lty=3,col=3,lwd=3)
#4
plot(x,y,type="b",lty=4,col=4,lwd=3)
rm(opar); # returns par to defaults
```

Solution #2

This generates the same graph as above, but is a shorter, more elegant script. It takes advantage of the loops discussed in Section 6.

```
# Create Data
x<-3:8;
y<-5*x+3;

# Draw Plot
opar<-par(mfrow=c(2,2),las=1);

for (i in 1:4){
     plot(x,y,type="b",lty=i,col=i,lwd=3)
}
rm(opar); # returns par to defaults
```



Exercise 5.5 (optional)

savePlot is an interesting command because it only works on the windows OS. This is a rare occurrence in R, but it is possible. On a windows machine, the following is a simple script that would work:

```
> plot(1:10, type="b",col=4,lwd=3);
> savePlot(filename="bob", type=c("png"));
```

This would save the plot as bob.png in your current working directory. You can type getwd() to learn what your current working directory is, and dir() to see its contents.

An alternative solution to the task proposed in Exercise 3.6 would be to use the png() or jpeg() command before drawing the plot. This redirects the command output from the screen into the file specified in the command. Use ?jpeg to learn more. Once you are finished drawing the figure, you must type the command dev.off() to turn off the graphic device you opened.

Section 8: Vectors

Exercise 8.1

```
> seq(1,13,by=4)
```

Here, you need to know how to use the optional parameter "by =". This can be found using ?seq.

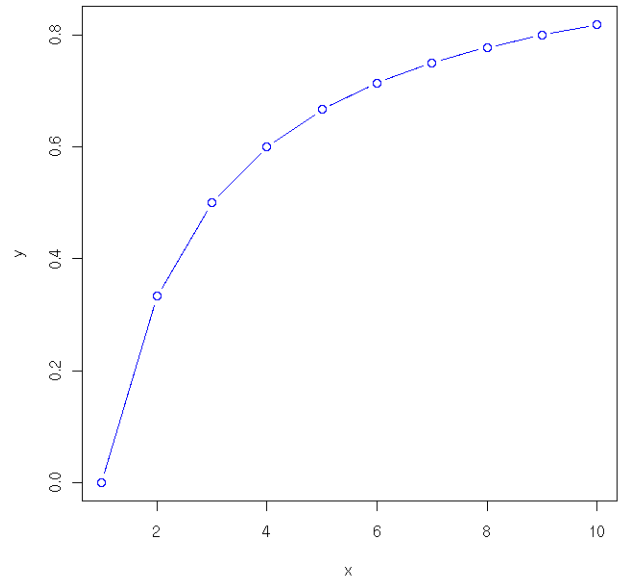
```
> seq(1,5,by=0.2)
 [1] 1.0 1.2 1.4 1.6 1.8 2.0 2.2 2.4 2.6 2.8 3.0 3.2 3.4 3.6 3.8 4.0 4.2 4.4 4.6
[20] 4.8 5.0
```

Exercise 8.2

```
> z=c(1,3,5,7,9,11)
> z[c(1,3,5)]=c(22,33,44)
> z
[1] 22  3 33  7 44 11
> z[c(2,1,3)]
[1]  3 22 33
```

Exercise 8.3

```
x=1:10
y=(x-1)/(x+1)
plot(x,y,col="blue",type="b")
```



Exercise 8.4

To solve this problem, it is helpful to recall a bit of information from sequences and series, which was probably one of the first topics you studied in calculus (if you don't believe me, check out a calculus book).

The goal here is to write a script that generates the sum of the following sequence:

$$\lim_{n\to\infty} 1 + r + r^2 + r^3 + \ldots + r^n = 2$$

We can rewrite the right hand side of this equation as $\sum_{n=0}^{\infty} r^n$ because $r^0 = 1$ and $r^1 = r$. This form of the equation helps us see the programming solution required. We need *n* to be a sequence

from 0 to a number using a step size of 1. If we were to write out a long script, we might use the following.

```
1> r = 0.5;
2> n = seq(0, 10, by=1);        # this creates n = [0, 1, 2, 3, …, 10]
3> l = r^n;                     # this creates l = [r^0, r^1, r^2, r^3,…, r^10]
4> sum(l);                      # this finds the sum of the elements in l
```

Thus, we can reduce these commands into one line as

```
> ans = sum( 0.5^(seq(0,10)))
```

Notice that I did not define the step size in the sequence command. I did not need to because the default step size is 1. An equivalent way to write the command would be to use the ":" as

```
> ans = sum( 0.5^(0:10) )
```

Exercise 8.5

```
> Light<50
 [1]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE FALSE FALSE FALSE FALSE
> Light2 <- Light[Light<50]
> Light2
[1] 20 20 20 20 21 24 44
> rmax=rmax[Light<50]
> rmax
[1] 1.73 1.65 2.02 1.89 2.61 1.36 2.37
> rmax=rmax[Light2]
> rmax
[1] NA NA NA NA NA NA NA
```

The problem here is that when we used Light2<50 for indexing rmax, we had already changed the Light2 vector.

Exercise 8.6

```
> # exercsie 8.6
> set.seed(273)
> x <- runif(20)
> m <- mean(x)
> ans <- x[x<m]
> ans
[1] 0.31074019 0.19282041 0.13098994 0.37142339 0.27956770 0.11890541 0.23134014
[8] 0.15844929 0.02412082
>
```

Section 9: Matrices

Exercise 9.1 (optional)

```
> X=matrix(c(1,2),2,4)
> X
     [,1] [,2] [,3] [,4]
[1,]    1    1    1    1
[2,]    2    2    2    2
```

Exercise 9.2

```
> matrix(rnorm(35,mean=1,sd=2),5,7)
          [,1]       [,2]       [,3]       [,4]      [,5]       [,6]
[1,]  0.0124933  1.6935916  2.2846427 -0.1355578 -0.148306 -5.0065003
[2,] -0.7350996 -1.3609527  0.2766241  0.2968747  1.178356  2.4682289
[3,]  3.8704409 -0.4688811 -1.8624548 -0.4004704  1.395131 -2.0472483
[4,]  2.2342166  1.2756400  1.0299978 -0.4123120 -1.113879 -2.2713479
[5,]  0.3438284  1.2179006 -2.7554464  4.3327163  2.207769 -0.6704255
          [,7]
[1,]  1.4362361
[2,] -1.2505336
[3,]  0.3638702
[4,]  0.2761186
[5,]  3.6694570
>
```

Exercise 9.3

```
> A=cbind(1:3,4:6,5:7); A
     [,1] [,2] [,3]
[1,]    1    4    5
[2,]    2    5    6
[3,]    3    6    7
> B=rbind(1:3,4:6); B
     [,1] [,2] [,3]
[1,]    1    2    3
[2,]    4    5    6
> rbind(A,B)
     [,1] [,2] [,3]
[1,]    1    4    5
[2,]    2    5    6
[3,]    3    6    7
[4,]    1    2    3
[5,]    4    5    6
> cbind(A,A)
     [,1] [,2] [,3] [,4] [,5] [,6]
[1,]    1    4    5    1    4    5
```

```
[2,]    2    5    6    2    5    6
[3,]    3    6    7    3    6    7
> cbind(A,B)
Error in cbind(A, B) : number of rows of matrices must match (see arg 2)
```

The last command fails because the dimensions of A and B do not align.