**Chris Kyriazis R tutorial output, sections 9.1-9.3, 9.6, 11, and 12**

**9.1.0.0.1**

x%%y finds the remainder of x/y

x%/%y conducts integer division of x and y

**9.1.1.0.1**

R gives the following warning message:

Warning message:

In x - y : longer object length is not a multiple of shorter object length

**9.1.2.0.1**

> v <- seq(1,13,4)

> v

[1] 1 5 9 13

> z <- seq(1,5,0.2)

> z

[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 4.8 5.0

**9.1.2.0.2**

Seq counts backwards:

> 3:1

[1] 3 2 1

**9.1.3.0.1**

> z <- c(1,3,5,7,9,11)

> seq(1,5,2)

[1] 1 3 5

> v <- z[seq(1,5,2)]

> v

[1] 1 5 9

v <- z[seq(1,5,2)] outputs elements (1,3,5) from z, which are (1,5,9)

**9.1.3.0.2**

> z[1:3]

[1] 1 3

**9.1.3.0.3**

> z[9] <- 11

> z

[1] 1 3 5 7 9 11 NA NA 11

The command adds 11 as the 9th element of z.

**9.1.3.0.4**

> x=1:10

> x

[1] 1 2 3 4 5 6 7 8 9 10

> y <- (x-1)/(x+1)

> plot(x,y, type="b")



**9.1.3.0.6**

For n=10:

> g <- 0.5^(0:10)

> sum(g)

[1] 1.999023

> 1/(1-0.5)

[1] 2

For n=50:

> g <- 0.5^(0:50)

> sum(g)

[1] 2

R rounded the the output to 2.

**9.1.5.0.1**

> Light <- Light[Light<50]

> Light

[1] 20 20 20 20 21 24 44

The original vector Light is modified, which we probably do not want to do.

**9.1.5.0.2 Exercise**

> a <- runif(20)

> a

[1] 0.32786624 0.52197784 0.83464494 0.35860395 0.62909576 0.46549993 0.26402961 0.68617089

[9] 0.80683662 0.18370394 0.87429601 0.15283529 0.08903098 0.78928709 0.46085254 0.40118903

[17] 0.22657921 0.59089398 0.87653898 0.38446094

> a.low <- a[a<mean(a)]

> a.low

[1] 0.32786624 0.35860395 0.46549993 0.26402961 0.18370394 0.15283529 0.08903098 0.46085254

[9] 0.40118903 0.22657921 0.38446094

**9.1.5.0.3**

> (1:20)[a<mean(a)]

[1] 1 4 6 7 10 12 13 15 16 17 20

**9.1.5.0.4**

Either select the odd values or drop the even values as shown in the example below:

> b<-1:20

> b[seq(1,19,2)]

[1] 1 3 5 7 9 11 13 15 17 19

> b[-seq(2,20,2)]

[1] 1 3 5 7 9 11 13 15 17 19

**9.2.1.0.1**

> X <- matrix(1:2,nrow=2,ncol=4)

> X

[,1] [,2] [,3] [,4]

[1,] 1 1 1 1

[2,] 2 2 2 2

> y <-matrix(rnorm(35,1,2),nrow=5,ncol=7)

> y

[,1] [,2] [,3] [,4] [,5] [,6] [,7]

[1,] -2.369186 1.3991757 1.44035651 -0.7704891 -1.2993681 2.4753397 0.2148796

[2,] 1.269206 1.4714665 2.08952065 2.2044122 -3.3020126 0.9193257 0.1900809

[3,] -2.034773 2.4966281 0.07302853 -1.4596033 1.0704249 0.8024925 2.3595333

[4,] -3.405430 0.9455087 3.45595725 -1.6082654 0.5070914 2.5124908 -1.7097236

[5,] -1.529109 3.0010698 -0.10098538 0.6489161 6.1597398 5.2953451 -1.8554573

**9.2.2.0.1**

The number of rows don’t match for cbind(C,D)

> rbind(C,D)

[,1] [,2] [,3]

[1,] 1 4 5

[2,] 2 5 6

[3,] 3 6 7

[4,] 1 2 3

[5,] 4 5 6

> cbind(C,C)

[,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(C,D)

Error in cbind(C, D) : number of rows of matrices must match (see arg 2)

**9.3.0.0.1**

> x <- seq(1,27);x

[1] 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27

> dim(x) <- c(3,9)

> is.array(x)

[1] TRUE

> is.matrix(x)

[1] TRUE

> x

[,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9]

[1,] 1 4 7 10 13 16 19 22 25

[2,] 2 5 8 11 14 17 20 23 26

[3,] 3 6 9 12 15 18 21 24 27

[3,] 3 6 9 12 15 18 21 24 27

Setting the dimensions of x changed it from a vector to a 3X9 matrix/array.

**9.6.0.0.1**

> data.url <- "http://kingaa.github.io/R\_Tutorial/hurricanes.csv"

> dat <- read.csv(data.url,comment.char='#')

> str(dat)

'data.frame': 92 obs. of 14 variables:

$ Year : int 1950 1950 1952 1953 1953 1954 1954 1954 1955 1955 ...

$ Name : Factor w/ 83 levels "Able","Agnes",..: 38 77 1 9 47 20 40 60 27 33 ...

$ MasFem : num 6.78 1.39 3.83 9.83 8.33 ...

$ MinPressure\_before : int 958 955 985 987 985 960 954 938 962 987 ...

$ Minpressure\_Updated.2014: int 960 955 985 987 985 960 954 938 962 987 ...

$ Gender\_MF : int 1 0 0 1 1 1 1 1 1 1 ...

$ Category : int 3 3 1 1 1 3 3 4 3 1 ...

$ alldeaths : int 2 4 3 1 0 60 20 20 0 200 ...

$ NDAM : int 1590 5350 150 58 15 19321 3230 24260 2030 14730 ...

$ Elapsed.Yrs : int 63 63 61 60 60 59 59 59 58 58 ...

$ Source : Factor w/ 3 levels "http://www.nhc.noaa.gov/pdf/NWS-TPC-5.pdf",..: 2 2 2 2 2 2 2 2 2 2 ...

$ ZMasFem : num -0.00094 -1.67076 -0.91331 0.94587 0.48108 ...

$ ZMinPressure\_A : num -0.356 -0.511 1.038 1.141 1.038 ...

$ ZNDAM : num -0.439 -0.148 -0.55 -0.558 -0.561 ...

**11.0.0.0.1**

X <- read.csv('ChlorellaGrowth.csv',comment.char='#')

# using '#' as a comment character allows us the CSV file to be self-documenting

par(cex=1.5,cex.main=0.9)

plot(rmax~log(light),data=X,xlab="Log light intensity (uE/m2/s)",ylab="Maximum growth rate (1/d)",pch=16)

# xlab and ylab are x and y axis labels, pch is "plotting character"

# cex is 'character expansion' - cex=1.5 increases symbol & label sizes by 50%

# cex.main sets the character expansion for the main title of the plot

title(main="Data from Fussmann et al. (2000) system")

fit <- lm(rmax~log(Light))

summary(fit); abline(fit)

# 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(3.8,3,paste("rmax=",c1,"+",c2,"light"))

# You can use ?round, ?text and ?paste to read about these commands

# for working with plots



**11.0.0.0.2**

plot(fit) resulted in several plots comparing the fitted values to the residuals, comparing theoretical quantiles vs standardized residuals, comparing fitted values vs sqrt(standardized residuals), and comparing the leverage of each data point vs its standardized residual. These plots are helpful for examining potential problems in the fit of our regression line (e.g., one or a few outliers driving our result).

**11.0.0.0.3**



**11.0.0.0.4**







**11.0.0.0.6**

> dev.copy(png,'myplot1.png')

quartz\_off\_screen

5

> dev.off()

RStudioGD

2

##### 12.1.0.0.2

##### If T has length 1, the loop runs once, and if T has length 0, the loop counts back from 1 to 0 and runs twice. By using one of the seq functions instead of 1:length(T), you can make sure that the loop only runs in the forward direction.

##### 12.2.0.0.1

> phi <- 20

> conv <- FALSE

> count=1

> while (!conv) {

+ phi.new <- 1+1/phi

+ conv <- phi==phi.new

+ phi <- phi.new

+ count=count+1

+ }

> print(count)

[1] 41

The code works and requires 41 iterations to complete.

**12.2.0.0.2**

a <- 1.1

b <- 0.001

N <- numeric(0)

n <- 2

count=1

while(TRUE) {

N[count] <- n

count <- count+1

n.new <- a\*n/(1+b\*n)

if (n==n.new) break

n <- n.new

}

plot(N)



The trajectory of the model is the same as with the for loop above, however it runs for about 400 iterations rather than the previously specified 200.