**9.1.0.0.1 What do the %% and %/% operators do?**

%% finds the remainder after the division of two numbers. While %/% produces the whole number portion of the quotient of two numbers.

##### **9.1.1.0.1 What happens when the length of the longer vector is not a multiple of that of the shorter?**

When the length of the longer vector is not a multiple of a shorter vector, R will recycle the elements in the shorter vector until the operation has been completed on each element of the longer vector. Furthermore, a warning message will be presented indicating that the longer vector is not a multiple.

**9.1.2.0.1 Use seq to create the vector v=(1 5 9 13), and to create a vector going from 1 to 5 in increments of 0.2 .**

vector(v)=(1 5 9 13)=seq(1,13,4)

Vector from 1 to 5 in increments of 0.2 =seq(1,5,0.2)

**9.1.2.0.2 What happens when to is less than from in seq? What result does 3:1 give?**

When the starting value is less than the the final value, R will count down. The result of 3:1 will be the vector v=(3 2 1).

##### **9.1.3.0.1 If you enter v <- z[seq(1,5,2)], what will happen?**

Entering v<-z[seq(1,5,2)] will produce a sequence of the first, fifth and second values of the vector z in that order.

##### **9.1.3.0.2 Write a *one-line* command to extract a vector consisting of the second, first, and third elements of z *in that order*.**

v<-z[seq(2,1,3)]

##### **9.1.3.0.3 What happens when I set the value of an element that doesn’t exist?**

If this occurs, R will create a value in the specified space and note “NA” in the spaces between what had been defined and the new value.

**9.1.3.0.4 Write code that computes values of y=(x−1)(x+1)y=(x−1)(x+1) for x=1,2,⋯,10x=1,2,⋯,10, and plots y versus x with the points plotted and connected by a line.**

x<-1:10;y<-(x-1)/(x+1);plot(x,y);lines(x,y)

##### **9.1.3.0.5 What happened to the third case?**

Because there were more than seven significant figures, R rounded to 2 but treated x as 2 instead of what occurred in the second case.

**9.1.3.0.6 Take r=0.5 and n=10, and write a one-line command that creates the vector G=(r0,r1,r2,...,rn)G=(r0,r1,r2,...,rn). Compare the sum (using sum()) of this vector to the limiting value 1/(1−r). Repeat for n=50.**

r<-0.5;n<-0:10;w<-r^n;sum(w)

r<-0.5;n<-0:50;w<-r^n;sum(w)

Both of these values are equal to the limiting value which is 2.

**9.1.4**

“=” sets the variable to whatever value you apply after. “==” is testing whether the two vectors are equal for each of the corresponding elements.

**9.1.5.0.1 What would happen if instead of setting lowLight you replaced Light by saying Light <- Light[Light<50]? Why would that be the wrong thing to do in the above example?**

Doing so would change the outcomes for the actions afterwards because you would replace the data frame is being used to create the new vectors.

**9.1.5.0.2 runif(n) is a function that generates a vector of n random, uniformly distributed numbers between 0 and 1. Create a vector of 20 numbers, then find the subset of those numbers that is less than the mean. More on runif and related functions soon.**

runif<-runif(20);subset1<-runif[runif<mean(runif)]

**9.1.5.0.3 Find the *positions* of the elements that are less than the mean of the vector you just created (e.g. if your vector were (0.1 0.9 0.7 0.3) the answer would be (1 4)).**

(3,4,8,9,10)

**9.1.5.0.4 Specify two ways to take only the elements in the odd positions (first, third, …) of a vector of arbitrary length.**

test<-c(1:6);result <- test[c(TRUE, FALSE)]

test<-c(1:6);c(seq(min(test), max(test), by=2))

**9.2.1.0.1 Use a command of the form X <- matrix(v,nrow=2,ncol=4) where v is a data vector, to create the following matrix X:**

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

**9.2.1.0.2 Use rnorm and matrix to create a 5×7 matrix of Gaussian random numbers with mean 1 and standard deviation 2.**

matrix(rnorm(35,mean=1,sd=2),nrow=5,ncol=7)

**9.2.2.0.1 Verify that rbind(C,D) works, cbind(C,C) works, but cbind(C,D) doesn’t. Why not?**

*C<-cbind(1:3,4:6,5:7);C*

*D<-rbind(1:3,4:6);D*

*rbind(C,D)*

*cbind(C,C)*

*cbind(C,D)*

cbind(C,D) does not work because the number of rows of the matrices must match.

**9.3.0.0.1 What happens when we set the dimension attribute on a vector?**

When you apply a dimension attribute to a vector, R creates a matrix with the values in the vector using the given the dimensions.

**9.6.0.0.1 Download the hurricanes.csv file from the tutorial website:** [**http://kingaa.github.io/R\_Tutorial/hurricanes.csv**](http://kingaa.github.io/R_Tutorial/hurricanes.csv)**. Examine the resulting data frame by printing it and using the str command. Note the class type of each variable.**

*library(readr)*

*hurricanes <- read\_csv("~/hurricanes.csv")*

*View(hurricanes)*

Classes of variables:

Year-integer

Name- character

MasFem-numerical

MinPressure\_before- integer

MinPressure\_updated.2014-integer

Gender\_MF- integer

Category- integer

Alldeaths-integer

NDAM- integer

Elapsed.Yrs-integer

Source-character

ZMasFem-numerical

ZMinPressure\_A-numerical

ZNDAM-numerical

**11.0.0.0.1 See R script “Updintro.R”**

**11.0.0.0.3 See R script “11\_0\_0\_0\_3”**

**11.0.0.0.4 See R script 11.0.0.0.4**

**11.0.0.0.5 and 11.0.0.0.6 See R script 11.0.0.0.5**

##### 