data(iris)  
head(iris)

## Sepal.Length Sepal.Width Petal.Length Petal.Width Species  
## 1 5.1 3.5 1.4 0.2 setosa  
## 2 4.9 3.0 1.4 0.2 setosa  
## 3 4.7 3.2 1.3 0.2 setosa  
## 4 4.6 3.1 1.5 0.2 setosa  
## 5 5.0 3.6 1.4 0.2 setosa  
## 6 5.4 3.9 1.7 0.4 setosa

sp\_ids = unique(iris$Species)  
  
output = matrix(0, nrow=length(sp\_ids), ncol=ncol(iris)-1)  
#make an emply matrix that is 3x4  
rownames(output) = sp\_ids  
#assign species names as the row names of the output  
colnames(output) = names(iris[ , -ncol(iris)])  
#assign species names as the column names of the output  
  
for(i in seq\_along(sp\_ids)) {  
 iris\_sp = subset(iris, subset=Species == sp\_ids[i], select=-Species)  
 for(j in 1:(ncol(iris\_sp))) {  
 #Loops through how many columns there are  
 x = 0  
 y = 0  
 #you need to start at 0 each time  
 if (nrow(iris\_sp) > 0) {  
 #Needs to be a real number  
 for(k in 1:nrow(iris\_sp))  
 #Loops through how many rows there are  
 {  
 x = x + iris\_sp[k, j]  
 #x is the sum of all the rows in a given column  
 y = y + 1  
 #y is the number of rows in a given column  
 }  
 output[i, j] = x / y   
 }  
 }  
}  
output

## Sepal.Length Sepal.Width Petal.Length Petal.Width  
## setosa 5.006 3.428 1.462 0.246  
## versicolor 5.936 2.770 4.260 1.326  
## virginica 6.588 2.974 5.552 2.026

1. Describe the values stored in the object output. In other words what did the loops create?

The values are means

1. Describe using pseudo-code how output was calculated.

Loop from 1 to length of species identities Take a subset of iris data Loop from 1 to number of columns of the iris data Make sure that the number is postive Loop through how many rows there are Sum all the values in the column Divide by how many total rows there are Final answer is mean

1. The variables in the loop were named so as to be vague. How can the objects output, x, and y could be renamed such that it is clearer what is occurring in the loop.

X- Sum of all variables Y- How many variables there are Output- Mean

1. It is possible to accomplish the same task using fewer lines of code? Please suggest one other way to calculate output that decreases the number of loops by 1.

sp\_ids = unique(iris$Species)  
  
output = matrix(0, nrow=length(sp\_ids), ncol=ncol(iris)-1)  
rownames(output) = sp\_ids  
colnames(output) = names(iris[ , -ncol(iris)])  
for(i in seq\_along(sp\_ids)) {  
 iris\_sp = subset(iris, subset=Species == sp\_ids[i], select=-Species)  
 for(j in 1:(ncol(iris\_sp))) {  
 x = 0  
 y = 0  
 for(k in 1:nrow(iris\_sp))  
 {  
 x = x + iris\_sp[k, j]  
 y = y + 1  
 }  
 output[i, j] = x / y   
 }  
 }  
output

## Sepal.Length Sepal.Width Petal.Length Petal.Width  
## setosa 5.006 3.428 1.462 0.246  
## versicolor 5.936 2.770 4.260 1.326  
## virginica 6.588 2.974 5.552 2.026

1. You have a vector x with the numbers 1:10. Write a for loop that will produce a vector y that contains the sum of x up to that index of x. So for example the elements of x are 1, 2, 3, and so on and the elements of y would be 1, 3, 6, and so on.

y=NULL  
x<- c(1,2,3,4,5,6,7,8,9,10)  
for (i in x){  
 y[i] = sum(x[1:i])}  
   
  
y

## [1] 1 3 6 10 15 21 28 36 45 55

1. Modify your for loop so that if the sum is greater than 10 the value of y is set to NA

y <- NULL  
x <- 1:10  
for (i in x) {  
 y[i] = sum(x[1:i])   
 if (y[i]>10) {  
 y[i]<-'NA'  
 }  
}   
y

## [1] "1" "3" "6" "10" "NA" "NA" "NA" "NA" "NA" "NA"

1. Place your for loop into a function that accepts as its argument any vector of arbitrary length and it will return y.

looploop <- function(p) {  
d=NULL  
for (i in p) {  
 d[i] = sum(p[1:i])  
}  
print (d)  
}  
looploop(x)

## [1] 1 3 6 10 15 21 28 36 45 55

hihi<-c(1:20)  
looploop(hihi)

## [1] 1 3 6 10 15 21 28 36 45 55 66 78 91 105 120 136 153  
## [18] 171 190 210