# HW2_MargaretWalker

*Margaret Walker*

*January 18, 2016*

## R Intermediate

### Iris loops

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

**The output of the loops is a matrix with three rows and four columns. The three rows are the three different iris species and the four columns are the different characteristics of the flowers (sepal length, sepal width, petal length, and petal width). Each value in the matrix is the average value of the flower characteristic for the different species. For example, the first value in the matrix is 5.006, which is the average sepal length for the species setosa.**

**2. Describe using pseudo-code how output was calculated.**

**First, loop through the three unique species in the iris dataset and create a dataframe for each species excluding the species column of the original iris data set. Then, loop through the different columns in the data set created above (that is the four different flower characteristics), then if the number of rows in iris_sp is greater than 0 loop through each row in the dataset created (sp_ids). For x you add all of the values in a particular column together, and then for y you are just adding 1 for each iteration to get the total number of rows. The final output is the mean for each characteristic and species because you divide the sum of the characterisic for each species by the total number of observations for each species.**

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

**Output could be renamed as species_averages since it is a matrix of the average values of the characterisitics for each species. X could be renamed columnsum because it is the sum of all the values for a particular column for a particular species. Then Y could be renamed numobservations because y ends up being the number of observations per species.**

**4. Is it possible to accomplish to accomplish the same task using fewer lines of code? Please suggest one other way to calculate output that decreases number of loops by one.**

I was able to rewrite to the loop to only contain one for loop. I used tapply to loop through the different columns of iris dataset except species and calculate the mean for each species. See loop below:

```
sp_ids = unique(iris$Species)
output = matrix(0,nrow=length(sp_ids), ncol=ncol(iris)-1)
rownames(output) = sp_ids
colnames(output) = names(iris[,-ncol(iris)])
for(i in 1:4){
  output[,i]=tapply(iris[,i], iris$Species,mean)
}
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
```

## Sum of a sequence

**5. 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.**

I created a simple for loop where I used the sum function to caculate the sum of the elements for each iteration and then I used the append function to append each sum into vector y. See code below:

```
y = c()
for(i in 1:10){
  s = sum(1:i)
  y = append(y, s)
}
y
```

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

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

I modified my loop by adding an if else statment to make sure NAs were added to the vector if the sum was larger than 10. See work below:

```
y = c()
for(i in 1:10){
  s = sum(1:i)
  if(s <= 10){
    y = append(y, s)
  }
  else{
    y = append(y, NA)
  }
}
y
```

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

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

I was able to create a sum_vector function by changing a few things in my original code. See code below as well as example I ran through the function to make sure it worked:

```
sum_vector = function(y){
  x = c()
  for(i in y){
    s = sum(1:i)
    x = append(x, s)
  }
  return(x)
}
vec=c(1,2,3,4,5,6)
sum_vector(vec)
```

```
## [1]  1  3  6 10 15 21
```

I also created a second function to work for the for loop where NA's are produced for values larger than 10. This function is called sum_vector2. See code below:

```
sum_vector2 = function(z){
  x = c()
  for(i in z){
    s = sum(1:i)
    if(s<=10){
      x = append(x, s)
    }
    else{
      x = append(x, NA)
    }
  }
  return(x)
}
vec2=c(1,2,3,4,5,6,7)
sum_vector2(vec2)
```

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