Intermediate Assignment_Radchenko

Anna Radchenko

January 14, 2020

Collaborator: Emily Parsons

```
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) #first function tells you the unique, not duplicated information i
n the data frame and named it sp_ids aka the single species names that there are

output = matrix(0, nrow=length(sp_ids), ncol=ncol(iris)-1)
#matrix creates a matrix from a given set of values
#get or set the length of vectors for the species ids
#0 for data to create an empty matrix that I will add data to later
#ncol returns the number of columns in the iris dataframe, we are then subtracting 1 and naming
 it
rownames(output) = sp_ids
colnames(output) = names(iris[ , -ncol(iris)])
#naming what the rownames and column names will be

for(i in seq_along(sp_ids))#counts through the sp_ids for the amount that there is (3) in this c
ase
  {
    iris_sp = subset(iris, subset=Species == sp_ids[i], select=-Species)
    #for each iris_sp, look at the subset of species for
    for(j in 1:(ncol(iris_sp))) {
      #another iterator saying 1 through the number of columns in iris_sp
        x = 0
        y = 0 #entering in a blank to be specified below
        if (nrow(iris_sp) > 0) {#if there are any rows in iris_sp greater than 0, checking that
 there is data in the row for each species
            for(k in 1:nrow(iris_sp)) {
                x = x + iris_sp[k, j] #adding row and column together for a sum
                y = y + 1 #gives the total number of observations
            }
            output[i, j] = x / y
        } #output = dataset value/1 for each species of interest and column
    }
}
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 loop created a table for the species setosa, versicolor and virginica with their

coresponding sepal and petal lengths and widths.

2. Describe using pseudo-code how `output` was calculated, for example,

{r, eval=FALSE} 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 If … occurs then do …

```
Identify the number of species in the iris dataset
Identify the rownames and columnames for the output datasets

Iterator in the count of species ids
      For each iris species look at the subset of species for
      all the columns in iris species
      if there is data in the row for iris_sp
      then take the sum (x) and number of observations (y),
      print a table with the mean of each variable for each species as an output
```

3. 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.

Output could be renamed as species means to understand what the ‘output’ at the end should look like.

x and y could be renamed as placeholders that the iterators were filling. X could be named sum while y could be called counts, as it is the amount of counts data type..

4. 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.

```
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)
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
```

removed the part of the loop that checks if there is data for that species. If you know that there is data for each species in the rows you

are working with, then you do not need to complete this logical check if (nrow(iris_sp) > 0)

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.

```
x <- c(1, 2, 3, 4, 5, 6, 7, 8, 9, 10)
y<-0
for (i in 1:10) {
    y[i] <- sum(x[1:i])
}
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

```
x <- c(1, 2, 3, 4, 5, 6, 7, 8, 9, 10)
y<-0
for (i in 1:10) {
    y[i] <- sum(x[1:i])

    if (y[i]>10) {
     y[i]<-NA
      } else {
      print(y[i])
      }
}
```

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

```
y
```

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

7. Place your for loop into a function that accepts as its argument any vector

of arbitrary length and it will return `y`.

```
series <- function (x) {

for (i in 1:length(x)) {
    y[i] <- sum(x[1:i])

 if (y[i]>10)
     y[i]= NA
      else {
      (y[i] <-sum(x[1:i]))
      }
 print(y)
}
}

series(8)
```

```
##  [1]  8  3  6 10 NA NA NA NA NA NA
```