Applied Quantitative Methods Assignment 1 and 2

Caroline Oliver

Basic R Questions

Questions 1 to 4 - Basic Data File Questions

```
# Set Up: download data file into RStudio
tallGrassPrairiePreserveData = read.csv(
  file = "http://dmcglinn.github.io/quant_methods/data/tgpp.csv", header = TRUE)
```

Question 1: What are the names of the columns in the dataset?

```
colnames(tallGrassPrairiePreserveData)
```

```
##  [1] "plot"      "year"      "record_id" "corner"    "scale"
##  [6] "richness"  "easting"   "northing"  "slope"     "ph"
## [11] "yrsslb"
```

Question 2: How many rows and columns does the data file have?

```
# Number of Rows:
nrow(tallGrassPrairiePreserveData)
```

```
## [1] 4080
```

```
# Number of Columns:
ncol(tallGrassPrairiePreserveData)
```

```
## [1] 11
```

Question 3: What kind of object is each data column?

```
sapply(tallGrassPrairiePreserveData, class)
```

```
##      plot      year record_id    corner     scale  richness   easting
## "integer" "integer" "integer" "integer" "numeric" "integer" "integer"
##  northing     slope        ph    yrsslb
## "integer" "integer" "numeric" "numeric"
```

Question 4: What are the values of the the datafile for rows 1, 5, and 8 at columns 3, 7, and 10?

```
# Row 1 Column 3:
tallGrassPrairiePreserveData[1,3]
```

```
## [1] 187
```

```r
# Row 1 Column 7:
tallGrassPrairiePreserveData[1,7]
```

```
## [1] 727000
```

```r
# Row 1 Column 10:
tallGrassPrairiePreserveData[1,10]
```

```
## [1] 6.9
```

```r
# Row 5 Column 3:
tallGrassPrairiePreserveData[5,3]
```

```
## [1] 191
```

```r
# Row 5 Column 7:
tallGrassPrairiePreserveData[5,7]
```

```
## [1] 727000
```

```r
# Row 5 Column 10:
tallGrassPrairiePreserveData[5,10]
```

```
## [1] 6.9
```

```r
# Row 8 Column 3:
tallGrassPrairiePreserveData[8,3]
```

```
## [1] 194
```

```r
# Row 8 Column 7:
tallGrassPrairiePreserveData[8,7]
```

```
## [1] 727000
```

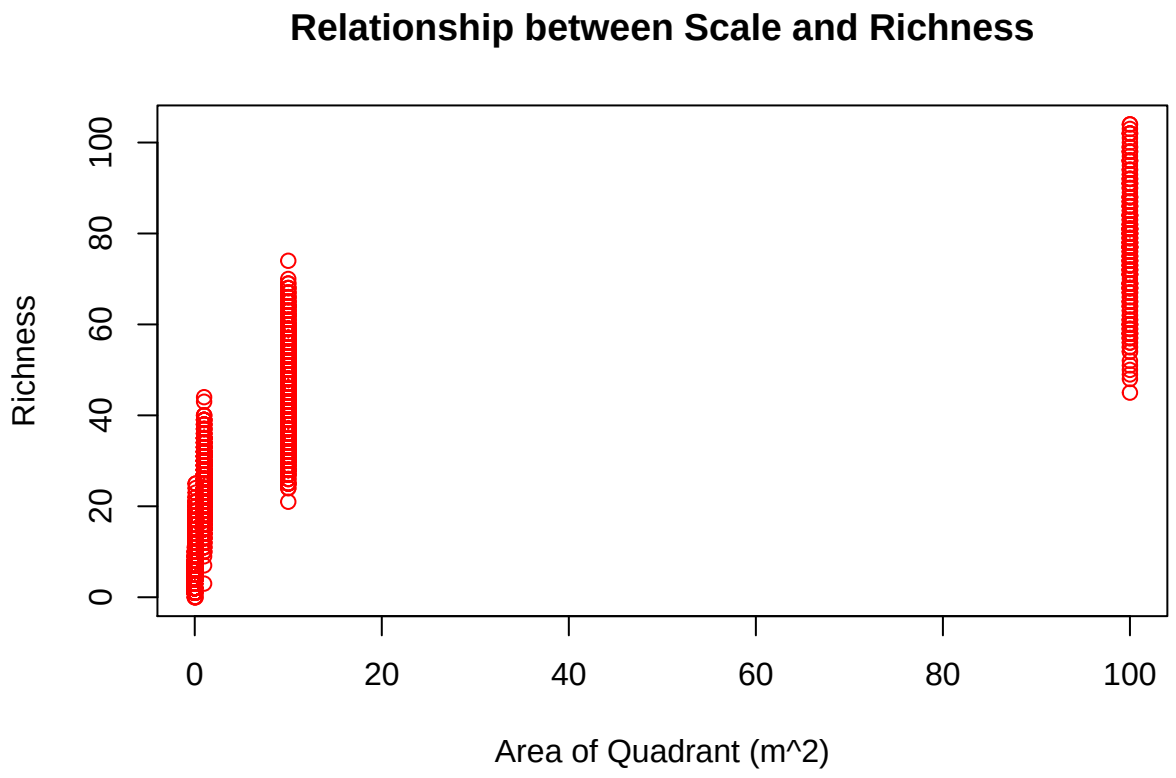
```r
# Row 8 Column 10:
tallGrassPrairiePreserveData[8,10]
```

```
## [1] 6.9
```

Plots - Questions 5 & 6

Question 5: PDF of the relationship between the variables "scale" and "richness"

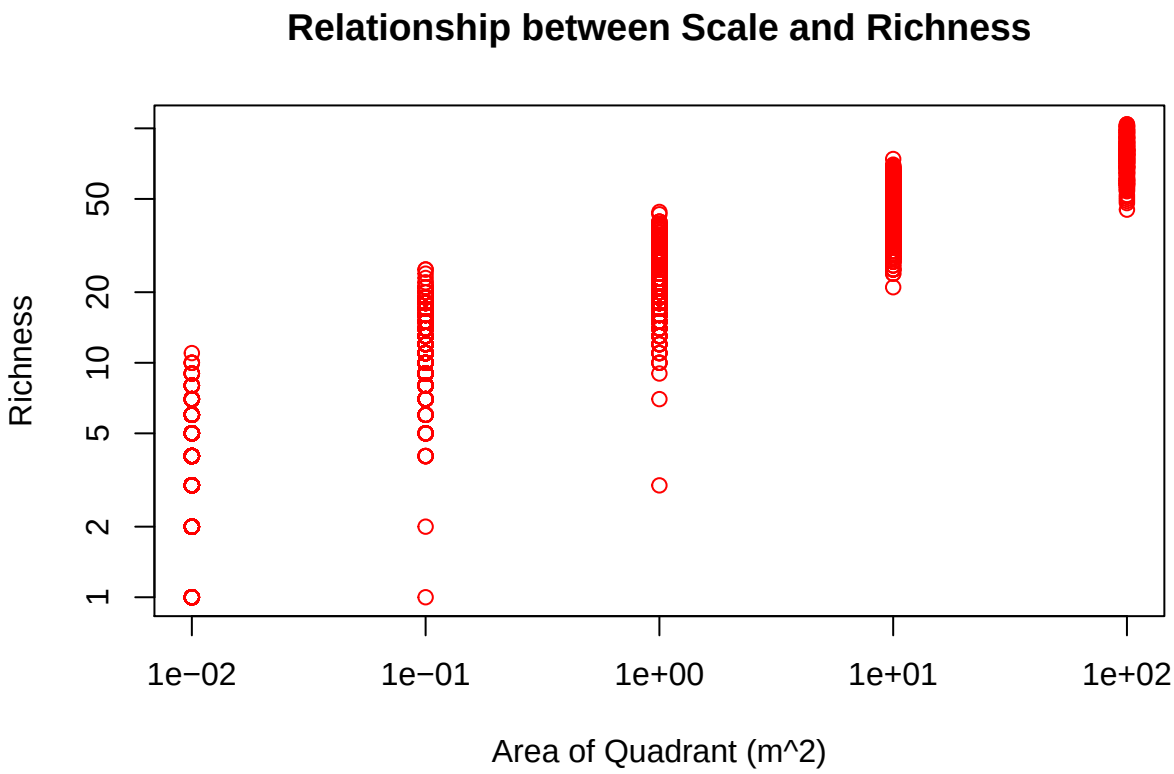
```r
plot(x = tallGrassPrairiePreserveData[,5], y = tallGrassPrairiePreserveData[,6],
     xlab = "Area of Quadrant (m^2)", ylab = "Richness",
     main = "Relationship between Scale and Richness", col = "red")
```



Question 6: What happens to your plot when you set the plot argument log equal to 'xy'?

```
plot(x = tallGrassPrairiePreserveData[,5], y = tallGrassPrairiePreserveData[,6],
     xlab = "Area of Quadrant (m^2)", ylab = "Richness",
     main = "Relationship between Scale and Richness", col = "red", log='xy')
```

```
## Warning in xy.coords(x, y, xlabel, ylabel, log): 4 y values <= 0 omitted
## from logarithmic plot
```



Adding "log = 'xy' " to the plot function changes the x axis (scale) back to log values which is how it is represented in the original data file. Also 4 values were ommitted from the log plot that were included in the non-logarithmic model in Question 5.

Intermediate R Questions

Question 1: what did the loops create?

The values stored in the output object are the averages of the three flower types sepal length and width, as well as, petal length and width arranged in matrix format.

Question 2: Describe using pseudo-code how output was calculated:

Loop through 3 types of flowers represented as integers 1, 2, and 3 and set the variable iris_sp to have only the samples from one type of flower at a time. Then loop through the columns (features, i.e. sepal length and width & petal length and width) of the instantiated flower type from the first loop and assign x and y variables to a 0 value for summation purposes. If the number of rows in a flower types feature is greater than 0, loop through the rows of each feature for each flower type. The x variable is a sum of all values in the rows for a particular feature and y is the number of rows or values for that feature. Output is then calculated for each flower types certain feature by taking the quotient of x over y.

Question 3: How can the objects output, x, and y could be renamed such that it is clearer what is occurring in the loop.

Other possible names:

output -> featureAverages

x -> featureSum

y -> numberOfFeatures

Question 4: It is possible to accomplish the same task using fewer lines of code?

```
data(iris)
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)
  if (nrow(iris_sp) > 0) {
      for(j in 1:(ncol(iris_sp))) {
        x = sum(iris_sp[j])
        y = nrow(iris_sp)

        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
```

For Question 4, I was able to decrease the number of loops by one by using the sum function to get the total of the values for a given flower's feature (x) and by using the nrow function to get the total number of data points for the given flower's feature (y). I then used those x and y values to calculate the average.

Question 5: for loop that will produce a vector y that contains the sum of vector x up to that index of x

```
x = 1:10
y = vector()
```

```
for(i in 1:length(x)){
  if(i==1)
  {
    y[i] = x[i]
  }
  else
    y[i] = y[i-1] + x[i]
}

# the updated y vector

y
```

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

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

```
x = 1:10
y = vector()
for(i in 1:length(x)){
    if(i==1)
    {
      y[i] = x[i]

      if(y[i] > 10){
        y[i] = "NA"
      }
    }
    else if(y[i-1] <= 10){
      y[i] = y[i-1] + x[i]

      if(y[i] > 10){
        y[i] = "NA"
      }
    }
    else
      y[i] = "NA"
}

# updated y vector with intergers larger than 10 represented with NA
suppressWarnings(as.integer(y))
```

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

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

```
incrementalVectorSummation = function(x){
    y = vector()
    for(i in 1:length(x)){
```

```
        if(i==1)
        {
          y[i] = x[i]

          if(y[i] > 10){
            y[i] = "NA"
          }
        }
        else if(y[i-1] <= 10){
          y[i] = y[i-1] + x[i]

          if(y[i] > 10){
            y[i] = "NA"
          }
        }
        else
          y[i] = "NA"
    }

    return(suppressWarnings(as.integer(y)))
}
```

Function call and output examples:

```
incrementalVectorSummation(x = 1:10)
```

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

```
incrementalVectorSummation(x = 1:20)
```

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

```
incrementalVectorSummation(x = -5:10)
```

```
##  [1]  -5  -9 -12 -14 -15 -15 -14 -12  -9  -5   0   6  NA  NA  NA  NA
```

```
incrementalVectorSummation(x = 4:1)
```

```
## [1]  4  7  9 10
```

NOTE:

The supressWarning function was used in Questions 6 and 7 because an error was being thrown about the NA values having to be coerced. This coersion did not effect the validity of the algorithms/functions output so I silenced the error messages for a cleaner output.