Balcewicz HW 1

Katie Balcewicz

1/17/2018

R basics

Download and read in the datafile "./quant_methods/data/tgpp.csv" from the class website. This dataset represents the vascular plant species richness that was collected from the Tallgrass Prairie Preserve from 10 x 10 m quadrats. Species richness is simply the number of species that occur within a quadrat.

Read the data into R, note this datafile has a header (i.e., it has column names) unlike the example we examined in class.

```
df <- read.csv('https://raw.githubusercontent.com/dmcglinn/quant_methods/gh-pages/data/tgpp.csv',
               header = TRUE)
```

1. What are the names of the columns in this dataset?

```
colnames(df)
```

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

2. How many rows and columns does this data file have?

```
nrow(df)
```

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

```
ncol(df)
```

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

3. What kind of object is each data column? Hint: checkout the function sapply().

```
str(df)
```

```
## 'data.frame':    4080 obs. of  11 variables:
##  $ plot     : int  205 205 205 205 205 205 205 205 205 205 ...
##  $ year     : int  1998 1998 1998 1998 1998 1998 1998 1998 1998 1998 ...
##  $ record_id: int  187 188 189 190 191 192 193 194 195 196 ...
##  $ corner   : int  NA 1 2 3 4 1 2 3 4 1 ...
##  $ scale    : num  100 10 10 10 10 1 1 1 1 0.1 ...
##  $ richness : int  60 36 34 37 33 21 23 19 25 10 ...
##  $ easting  : int  727000 727000 727000 727000 727000 727000 727000 727000 727000 727000 ...
##  $ northing : int  4080000 4080000 4080000 4080000 4080000 4080000 4080000 4080000 4080000 4080000 .
##  $ slope    : int  3 3 3 3 3 3 3 3 3 3 ...
##  $ ph       : num  6.9 6.9 6.9 6.9 6.9 6.9 6.9 6.9 6.9 6.9 ...
##  $ yrsslb   : num  0.39 0.39 0.39 0.39 0.39 0.39 0.39 0.39 0.39 0.39 ...
```

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

```
df[c(1,5,8),c(3,7,8)]
```

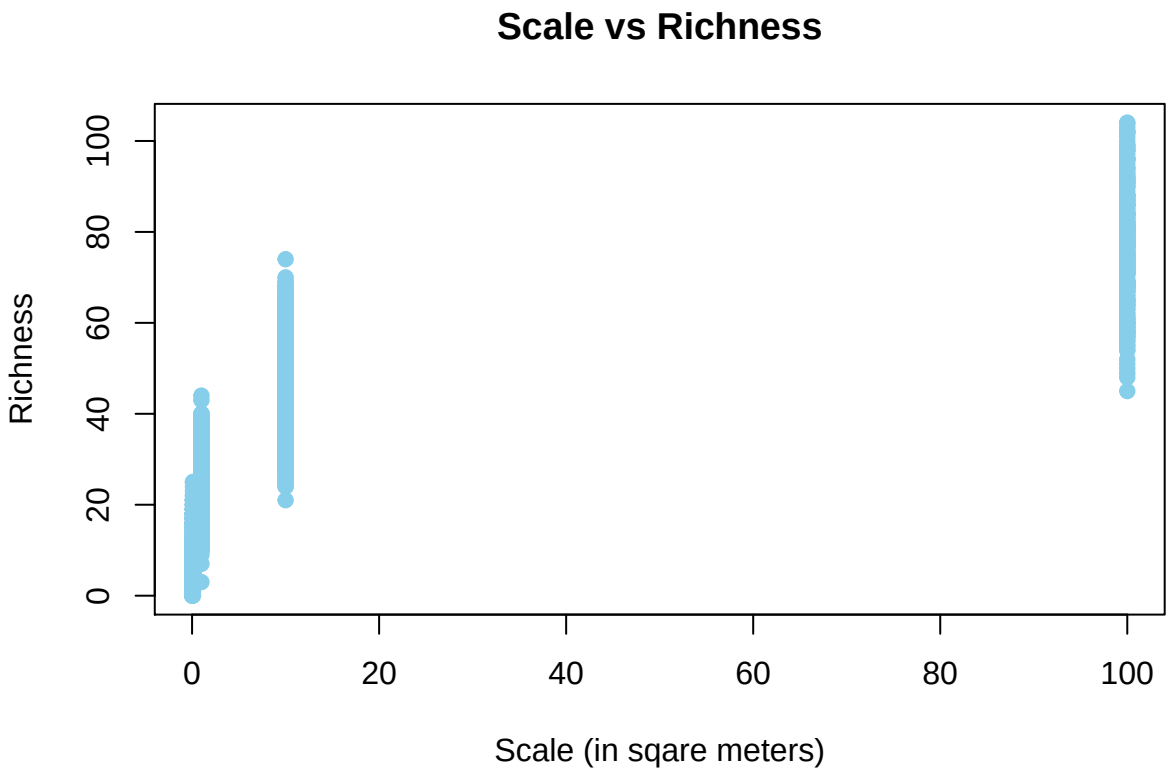
```
##   record_id easting northing
## 1       187  727000  4080000
```

```
## 5        191  727000  4080000
## 8        194  727000  4080000
```

5. Create a pdf of the relationship between the variables "scale" and "richness". Scale is the area in square meters of the quadrat in which richness was recorded. Be sure to label your axes clearly, and choose a color you find pleasing for the points. To get a list of available stock colors use the function colors(). Also see this link:

http://research.stowers-institute.org/efg/R/Color/Chart/index.htm.

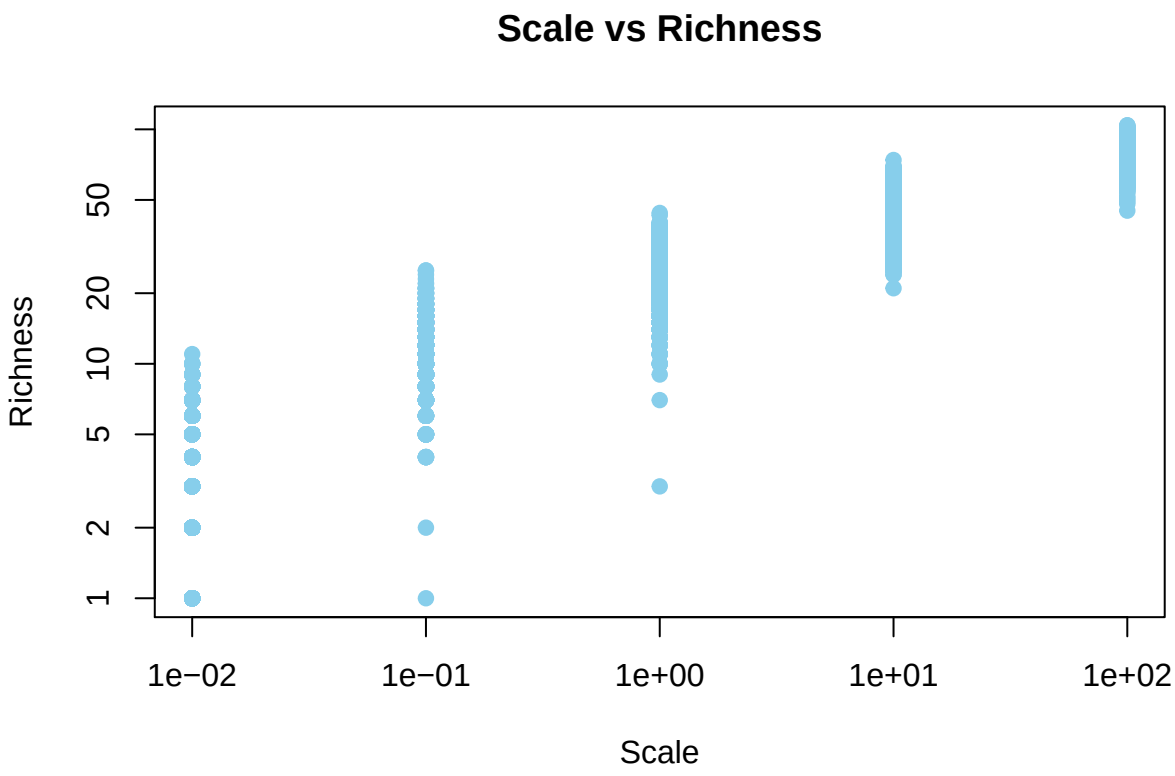
```
plot(x = df$scale, y = df$richness, pch = 19, col = "skyblue", main = "Scale vs Richness",
     xlab = "Scale (in sqare meters)", ylab = "Richness")
```



6. What happens to your plot when you set the plot argument log equal to 'xy'. plot(. . . , log='xy')?

```
plot(x = df$scale, y = df$richness, pch = 19, log = 'xy', col = "skyblue", main = "Scale vs Richness",
     xlab = "Scale", ylab = "Richness")
```

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



R intermediate

```
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
        if (nrow(iris_sp) > 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. Describe the values stored in the object output. In other words what did the loops create?

Output stores the mean values of each column for each type of iris.

2. Describe using pseudo-code how output was calculated

```
loop through iris species
  loop through first four columns of iris table
    loop through rows of iris table
      increase column sum by cell amount
      increase row count by one
    output cell = column sum / row count
```

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.

Rename output to species.averages, x to column.sum, y to row.count

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.

```
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 = sum(iris_sp[,j])
        y = nrow(iris_sp)
        if (nrow(iris_sp) > 0) {
            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
```

You can also use ddply in one line of code:

```
library(plyr); library(dplyr)
```

```
output = ddply(iris, .(Species), summarize, Sepal.Length = mean(Sepal.Length),
                                            Sepal.Width = mean(Sepal.Width),
                                            Petal.Length = mean(Petal.Length),
                                            Petal.Width = mean(Petal.Width))
output
```

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

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(1:10)
y = numeric(10)
sum = 0
for(i in x){
  sum = sum + i
  y[i] = sum
}
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(1:10)
y = numeric(10)
sum = 0
for(i in x){
  sum = sum + i
  if(sum <= 10)
    y[i] = sum
  else
    y[i] = NA
}
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.

```
sum.to.index = function(x){
  y = numeric(length(x))
  sum = 0
  for(i in x){
    sum = sum + i
    if(sum <= 10)
      y[i] = sum
    else
      y[i] = NA
  }
```

```
  return(y)
}
sum.to.index(c(1:15))
```

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