Assignment #1

Janelle Morano

09 Feb 2022

BEE 4310/6310: Multivariate Statistics for Environmental Applications

Assignment #1 (10 points)

1.

In this assignment you will build intuition with matrix transformations using annual streamflow data from two USGS streamflow gages.

a. Read in the file containing the annual stream flow rate from 2 different gages. #01181000 is downstream of an unregulated tributary, and upstream of the other gage, #01183500, which is downstream of 3 major reservoirs.

```
data <- read.table("/Users/janellemorano/Git/Reference-R-scripts/Envtl-Multivariate-Stats/Annual_Flow_W
    header = TRUE)
head(data)
```

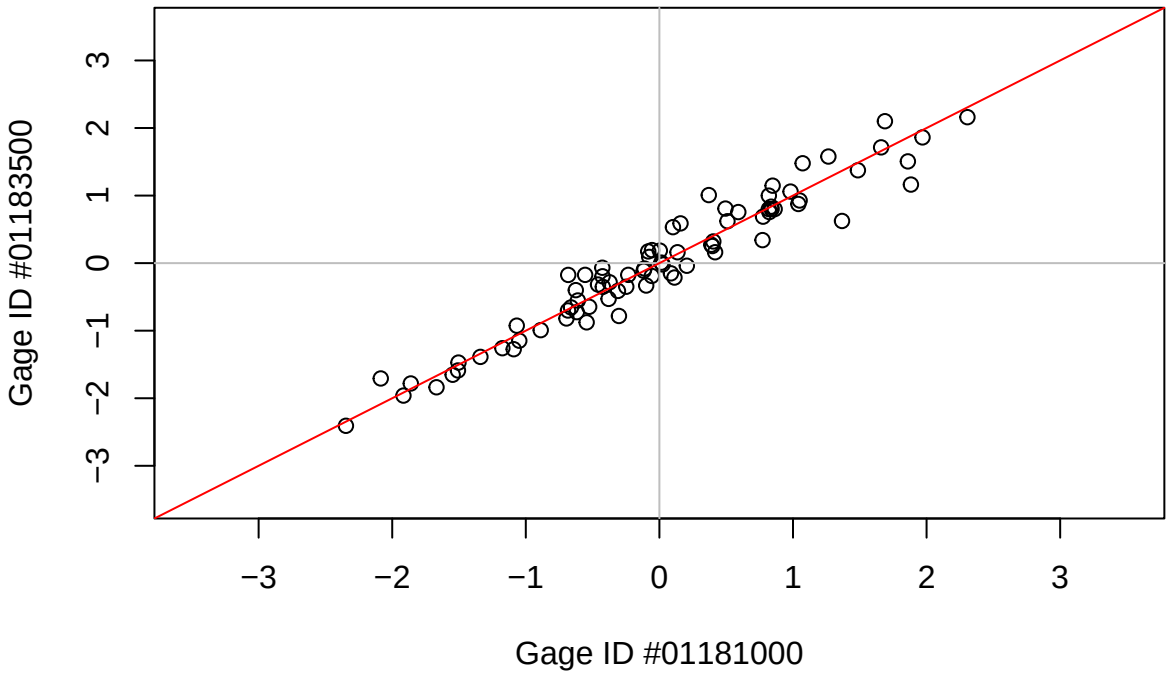
```
##   year gage.01181000 gage.01183500
## 1 1936         175.3         922.8
## 2 1937         218.4        1040.0
## 3 1938         264.8        1367.0
## 4 1939         175.2         954.5
## 5 1940         175.4         883.7
## 6 1941          99.3         525.2
```

b. Standardize the annual flow measurements of each gage by subtracting the mean from the annual measurement, then dividing by the standard deviation. Then combine these data into a nx2 matrix (both.gages.scale)

```
gage1.scale <- (data$gage.01181000 - mean(data$gage.01181000))/sd(data$gage.01181000)
gage2.scale <- (data$gage.01183500 - mean(data$gage.01183500))/sd(data$gage.01183500)
both.gages.scale <- cbind(gage1.scale, gage2.scale)
```

c. Plot the scaled data for the two gages against each other. Include a horizontal line, vertical line, and the 1:1 line, all of which pass through the origin.

```
plot(both.gages.scale[, 1], both.gages.scale[, 2], xlim = c(-3.5, 3.5), ylim = c(-3.5,
    3.5), xlab = "Gage ID #01181000", ylab = "Gage ID #01183500")
abline(h = 0, col = "gray")  #horizontal line through origin
abline(v = 0, col = "gray")  #vertical line through origin
abline(a = 0, b = 1, col = "red")  #1:1 line passing through intercept and slope of line
```



d. Create a 2x2 orthogonal projection matrix, P, that will collapse the scaled data from the 2 gages onto the 1:1 line. Project the both.gages.scaled onto the 1:1 line using matrix P and store the resulting projections in a matrix.

First, remember that an orthogonal projection is projecting a vector, x, onto vector, w, to get

$$x_w = P_w$$

and then thus the projection of x onto w is

$$P_x = (w * w^T)/(||w||^2)$$

For the 1:1 line, (1, 1), (-1, -1), (2, 2), etc. are all points that exist on the line. So, w needs to be set to any point on the vector line. Let's choose (1,1).

```
# Put (1,1) as a point on vector, w
w <- c(1, 1)
# Make it a matrix
wm <- as.matrix(w)
# Transpose it
wT <- t(w)
```

Now, we need to calculate the norm of the matrix w (wm). This can be done by hand or with function norm() in base R.

```
# Create your own function, which is the square root of the sum of x-squared or
# the square root of x*x^T.
norm_vec <- function(x) sqrt(sum(x^2))
P1 <- (w %*% wT)/(norm_vec(wm)^2)

# Or use built-in function, norm(), type = '2' (2=Euclidean norm)
P2 <- (w %*% wT)/(norm(c(1, 1), type = "2")^2)

# Compare
P1
```

```
##      [,1] [,2]
## [1,]  0.5  0.5
## [2,]  0.5  0.5
```

```
P2
```

```
##      [,1] [,2]
## [1,]  0.5  0.5
## [2,]  0.5  0.5
```

```
# Great! Let's make P1 = P
P <- P1
```

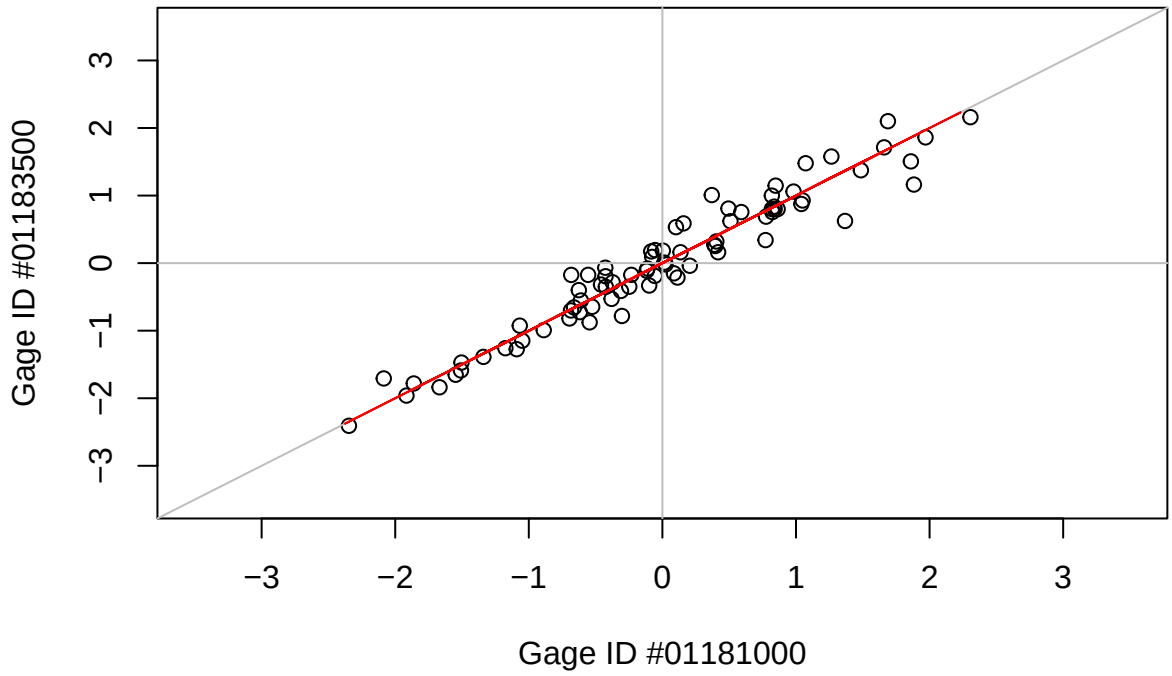
Ok, so now you've got the projection matrix, P, and you need to project both.gages.scaled onto the 1:1 line.

```
my.projection <- both.gages.scale %*% P
head(my.projection)
```

```
##            [,1]       [,2]
## [1,] -0.3101616 -0.3101616
## [2,]  0.3307566  0.3307566
## [3,]  1.4214680  1.4214680
## [4,] -0.2478532 -0.2478532
## [5,] -0.3872357 -0.3872357
## [6,] -1.8213055 -1.8213055
```

e. Now plot those projected data on the graph from before.

```
plot(both.gages.scale[, 1], both.gages.scale[, 2], xlim = c(-3.5, 3.5), ylim = c(-3.5,
    3.5), xlab = "Gage ID #01181000", ylab = "Gage ID #01183500")
abline(h = 0, col = "gray")  #horizontal line through origin
abline(v = 0, col = "gray")  #vertical line through origin
abline(a = 0, b = 1, col = "gray")  #1:1 line passing through intercept and slope of line
lines(my.projection, col = "red")
```



f. Calculate the Euclidean norm of each of the new projected data points.

```
# This is wrong Transpose my.projection and plug it into the orthogonal formula
# my.projection.T <- t(my.projection) my.projection.norm <- (my.projection %*%
# my.projection.T)/ (norm_vec(wm)^2) head(my.projection.norm)

# This is also wrong, because the norm is being calculated for the entire
# matrix.  my.norm <- norm(my.projection, type = '2') test <- as.matrix(2, 2,
# 2)

# The norm should be calculated for every point (i.e. each line of the matrix).
# This is Taylor Brown's code. I tried to play with making it happen in a
# different way, but without success.
my.norm <- data.frame(Norm1 = NULL)
for (i in 1:length(my.projection[, 1])) {
    subs = my.projection[i, ]
    n1 = norm(subs, type = "2")
    my.norm = rbind(my.norm, n1)
}
head(my.norm)
```

```
##   X0.438634772078301
## 1          0.4386348
## 2          0.4677604
## 3          2.0102593
## 4          0.3505174
## 5          0.5476339
## 6          2.5757149
```

```
# Alex Koeberle used this, which is a different approach calculate euclidean
# norm
my.normAK <- (abs(my.projection[, 1])^2 + abs(my.projection[, 2])^2)^0.5
head(my.normAK)
```

```
## [1] 0.4386348 0.4677604 2.0102593 0.3505174 0.5476339 2.5757149
```

g. The projected data can either be positive or negative (see my.projection). For the data points that fall in the 1st or 3rd quadrants (negative values), multiply the Euclidean norm by -1 to indicate a negative anomaly along the projected axis. It would be most efficient to add in that negative value in the loop above, where we calculated the norm.

```
# Repeat code above as if we started from scratch. But save as my.norm2 just in
# case.
my.norm2 <- data.frame(Norm1 = NULL)
for (i in 1:length(my.projection[, 1])) {
    subs = my.projection[i, ]
    n1 = norm(subs, type = "2")
    if (subs[c(1)] < 0) {
        # Since the columns are identical, only need to check 1
        n1 = n1 * -1
    }
    my.norm2 = rbind(my.norm2, n1)
}
head(my.norm2)
```

```
##   X.0.438634772078301
## 1          -0.4386348
## 2           0.4677604
```

```
## 3           2.0102593
## 4          -0.3505174
## 5          -0.5476339
## 6          -2.5757149
```

```
# Alex Koeberle used this Multiply euclidean norm for 3rd quadrant

full.matrix <- cbind(my.projection, my.norm2)

adj.values <- ifelse(full.matrix[, 1] < 0, full.matrix[, 3] * -1, full.matrix[, 3])

full.matrix.adj <- cbind(full.matrix, adj.values)
head(full.matrix.adj)
```

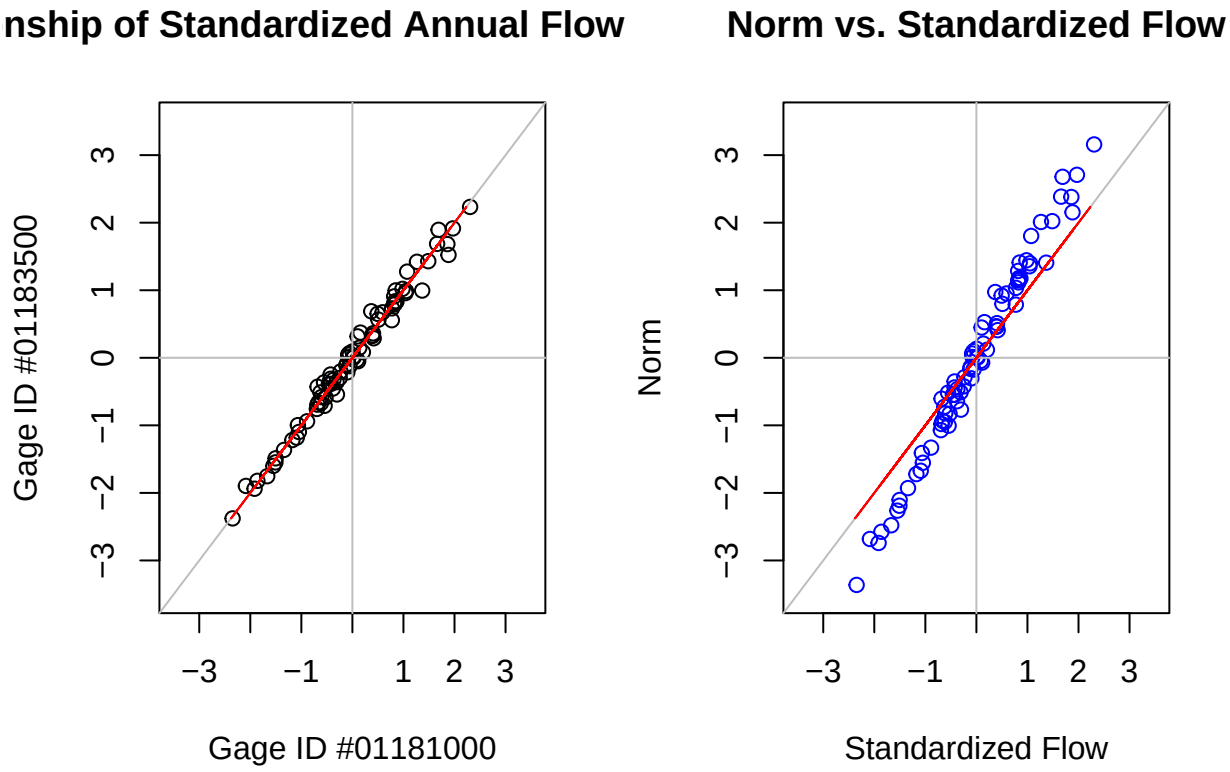
```
##            1          2 X.0.438634772078301 adj.values
## 1 -0.3101616 -0.3101616          -0.4386348  0.4386348
## 2  0.3307566  0.3307566           0.4677604  0.4677604
## 3  1.4214680  1.4214680           2.0102593  2.0102593
## 4 -0.2478532 -0.2478532          -0.3505174  0.3505174
## 5 -0.3872357 -0.3872357          -0.5476339  0.5476339
## 6 -1.8213055 -1.8213055          -2.5757149  2.5757149
```

h. Plot my.norm against the standardized data for both gages (both.gages.scale). I've added titles to explain what is being shown, but I am not clearly understanding how to interpret what is plotted.

```
par(mfrow = c(1, 2))

plot(both.gages.scale[, 1], my.projection[, 2], xlim = c(-3.5, 3.5), ylim = c(-3.5,
    3.5), xlab = "Gage ID #01181000", ylab = "Gage ID #01183500", main = "Relationship of Standardized
abline(h = 0, col = "gray")  #horizontal line through origin
abline(v = 0, col = "gray")  #vertical line through origin
abline(a = 0, b = 1, col = "gray")  #1:1 line passing through intercept and slope of line
lines(my.projection, col = "red")

plot(both.gages.scale[, 1], my.norm2[, 1], xlim = c(-3.5, 3.5), ylim = c(-3.5, 3.5),
    col = "blue", xlab = "Standardized Flow", ylab = "Norm", main = "Norm vs. Standardized Flow")
abline(h = 0, col = "gray")  #horizontal line through origin
abline(v = 0, col = "gray")  #vertical line through origin
abline(a = 0, b = 1, col = "gray")  #1:1 line passing through intercept and slope of line
lines(my.projection, col = "red")
```



i. Some interpretation. . .

Q: Is it possible to use my.norm to summarize the variability of the annual streamflow at both gages? A: Yes, because the values of both gages were projected down into a subspace, my.norm, which represents a summary variable of the 2 gages. The 1:1 line represents the scaled average flow, where positive values represent higher than average, and negative values represent lower than average.

Q: How good is this summary, based on the strength of the relationships between my.norm and each gage? A: The trend is that the summary value explains many of the values pretty well because they closely follow the summary value. However, the values at the extreme ends deviate more and are therefore not as good as being explained by the summary value.