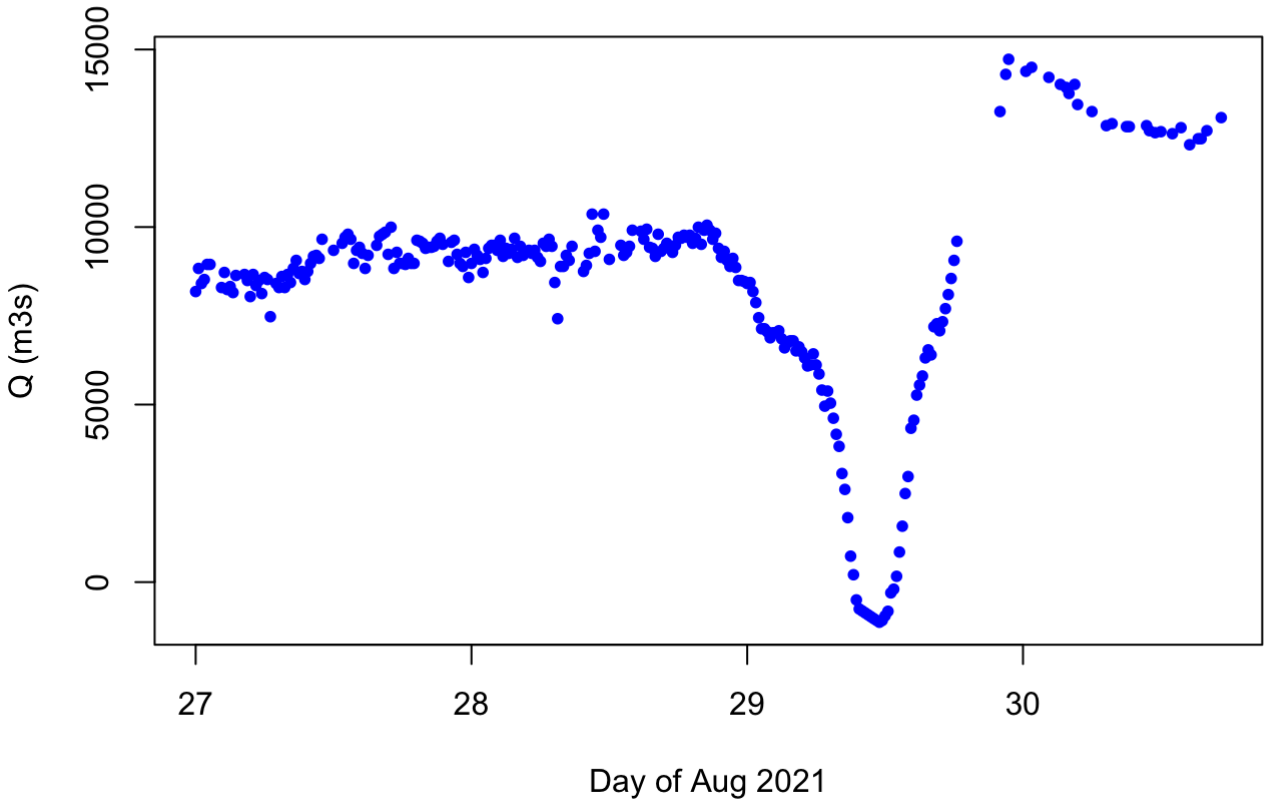
QM Programming Basics 4 - Matrices, more on plotting

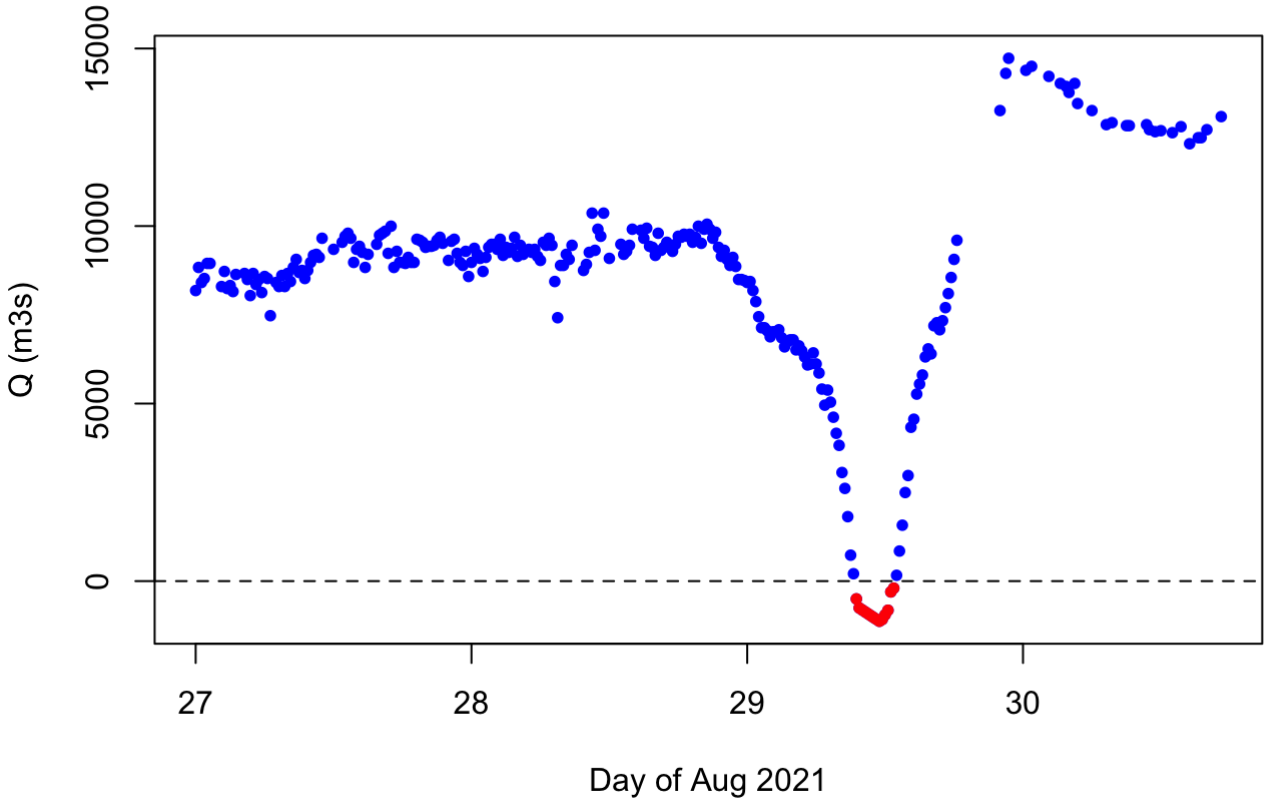
We've covered quite a bit of ground in our first week. Looking at the code we generated last class to open and plot the USGS discharge record for the lower Mississippi River during Hurricane Ida [also in Collab Resources/ClassCodes as ReadUSGS.R]

```
Qdf = read.csv("USGSMissRDischargeAug2021.csv")
#USGS record of discharge on lower Mississippi during Hurricane Ida

DecHr = Qdf$Hour + Qdf$Min/60  #decimal hour
DecDay = Qdf$Day + DecHr/24   #decimal day
Qm3s = Qdf$Qcfs * (0.3048^3)  #convert cfs to m3/s

plot(DecDay, Qm3s, pch=20, col="blue", xlab="Day of Aug 2021", ylab="Q (m3s)")
```



Things that might improve the visualization of this record? Perhaps add a line at Q=0; make the points red when Q<0.

```
plot(DecDay, Qm3s, pch=20, col="blue", xlab="Day of Aug 2021", ylab="Q (m3s)")
abline(h=0, lty = 2)
points(DecDay[Qm3s<0], Qm3s[Qm3s<0], pch=20, col="red")
```



We might also want make use of the DateTime column in Qdf. We will talk about this in a few weeks when we focus on methods for working with time series data.

Assignment 1 asks you to go through a similar process to what we did with the USGS data, plus calculate and plot some statistics, for a record of SST.

One difference between the SST dataset and the USGS data is that the SST data are in the form of a table. A table or spreadsheet of data can take the form of a matrix, a structure with rows and columns.

I'm going to spend a few minutes introducing matrices (or arrays) and then we'll come back to Assignment 1.

Matrices and arrays

We can create a matrix using the matrix() function [or we can read one in from a datafile].

```
A = matrix(c(1,2,3,4,5,6),nrow=2,ncol=3)
```

What do you think this will look like? In particular, what is the first row? 1 2 3 or 1 3 5?. Internal storage of matrix in R is in column-major order. That is, it fills the first column, then the 2nd…, so the first row would be 1 3 5.

```
print (A)
```

```
##      [,1] [,2] [,3]
## [1,]    1    3    5
## [2,]    2    4    6
```

What if we wanted it to fill rowwise rather than columnwise? There are several options.

```
A2 = matrix(c(1:6),nrow=2,byrow=T)  #specify to fill by row
print(A2)
```

```
##      [,1] [,2] [,3]
## [1,]    1    2    3
## [2,]    4    5    6
```

```
A2 = t(matrix(c(1:6),nrow=3)) #take transpose of matrix
print(A2)
```

```
##      [,1] [,2] [,3]
## [1,]    1    2    3
## [2,]    4    5    6
```

```
A2 = rbind(c(1,2,3),c(4,5,6))  #rbind() = row bind
                               #also cbind() = column bind
print(A2)
```

```
##      [,1] [,2] [,3]
## [1,]    1    2    3
## [2,]    4    5    6
```

Note 1: You can just specify # rows or cols since one determines other for a given vector length. R will complain if the number of elements in the matrix does not equal the product of nrow*ncol. Note 2: Elements of a matrix are specified by their row and column. E.g., A2[2,1] = ? (2 or 4)?

```
A2[2,1]
```

```
## [1] 4
```

The 1st value in [ ] is the row number (or index), the 2nd value is the column number.

Now try typing A2vec = c(A2).

```
A2vec = c(A2)
print(A2vec)
```

```
## [1] 1 4 2 5 3 6
```

This recreates a vector from the matrix - again columnwise. [Something I ask you to do in Assignment 1.]

Certain functions are designed to work on tables of data (as matrices of data frames). E.g., colMeans() calculates the mean of each column; rowMeans does same for each row

```
colMeans(A)
```

```
## [1] 1.5 3.5 5.5
```

What if the matrix includes an NA?

```
A3 = A2
A3[2,2] = NA
A3
```

```
##      [,1] [,2] [,3]
## [1,]    1    2    3
## [2,]    4   NA    6
```

```
rowMeans(A3)
```

```
## [1]  2 NA
```

We can ask many R functions to ignore NAs when performing their calculations by adding na.rm = TRUE to the command.

```
rowMeans(A3, na.rm=TRUE)
```

```
## [1] 2 5
```

Also colSums, rowSums. But no comparable command for standard deviation. For that we can use the “apply” function (see help apply):

```
Asd = apply (A, 1, sd)  #1 for row-wise, 2 for column-wise
print(Asd)
```

```
## [1] 2 2
```

Assignment 1

In Assignment 1 I have you read in a data file and create a matrix from it.

```
SSTdf = read.csv("SSTOI_CHLV2.csv")
SST = as.matrix(SSTdf[, 3:12])  #X[,1] give first column of X, X[1,] gives first row of
 X
M = SSTdf$M  #or M = SSTdf[,1]
```

If we look at SST in the Environment, it indicates the numbers of rows and columns. If we type class(SST):

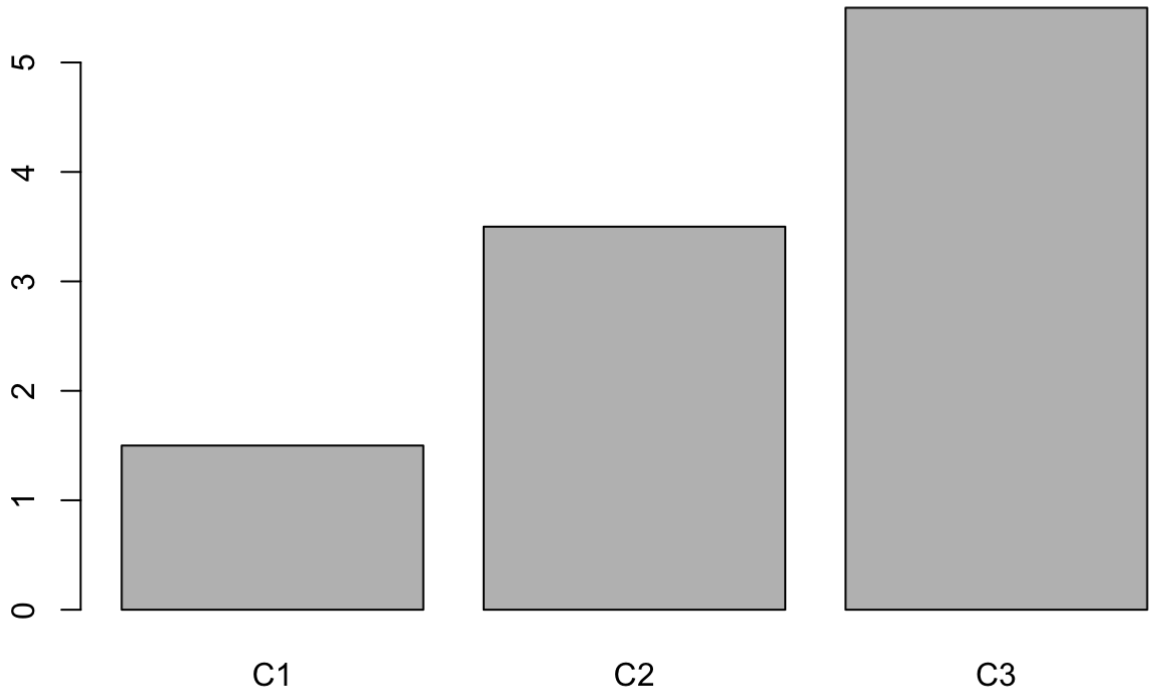
```
class(SST)
```

```
## [1] "matrix" "array"
```

It is a matrix or array object. Note: matrices are 2D whereas arrays can have more dimensions.

Assignment 1 introduces you to several new types of plots. E.g., barplot: barplot(height, …) see help barplot

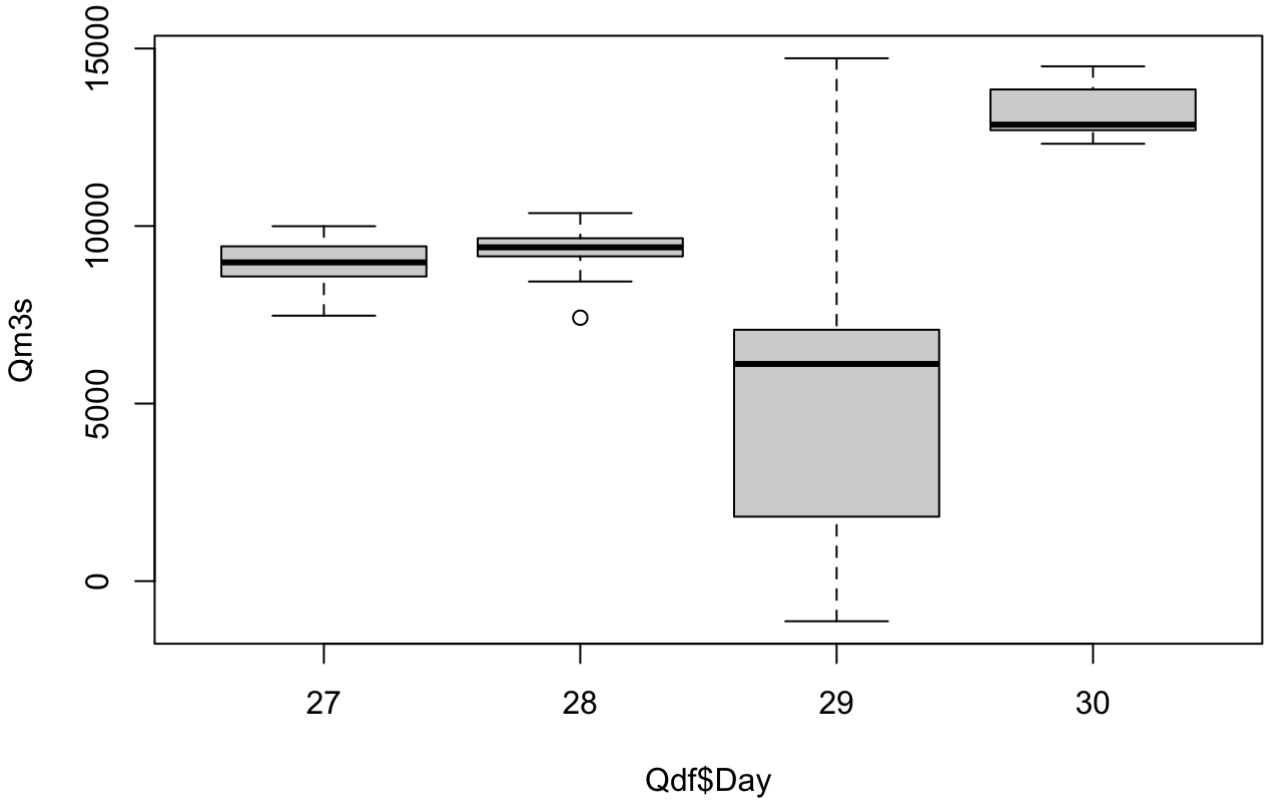
```
barplot(colMeans(A), names.arg = c("C1","C2","C3"))
```



boxplot()

boxplots summarize the statistics of values in a column or group. We can see how this works if we apply it to the USGS data, making a boxplot of discharge for each day.

```
boxplot(Qm3s ~ Qdf$Day)
```



The boxes outline the 25th-75th percentiles of the distribution of values in each group (each day in this case). The upper and lower bars indicate the 5th and 95th percentiles. The bar in the middle is the median. Dots beyond the bars are “outliers”.

In Assignment 1 I ask you to create a boxplot of monthly SST.