DSCI/MATH 530 RLab Two

Drew Remmenga

1 Bagging a sample.

The technique of bagging is used in machine learning to generate different representations of a data set and helps to avoid overfitting. It is also the basic technique used in bootstrapping a technique for statistical inference.

If a sample has n values $x_1, x_2, \ldots, x_n$ then a bagged sample is found by randomly selecting these values with replacement to create another sample of size n. If `X` is the vector of values in R of size 50 then

```
bagSample<- sample( X,50 , replace = TRUE)
```

will give a new sample of size 50, drawn randomly from `X` with replacement.

Here is the interesting feature of this technique: on average about 1/3 of the values in `X` will *not* be part of the bagged sample. Instead some values of `X` will be repeated. One can use a simple probability argument to show the expected fraction should converge to

$$1/e \approx 1/2.7182 = .3679$$

as the sample size gets large. However, one can also test this by Monte Carlo. Here is some R code to do it. I am also creating a random sample to use for testing. If we are just intereted in the *number* of values in the out-of-bag sample, the actual sample values do not matter (why?). You might want to run the code below and also print out `X` and the `bagSample` just make sure you understand how this works.

```r
set.seed( 123)
n<- 50
X<- runif( n)
bagSample<- sample( X,n , replace = TRUE)
m<- length( unique( bagSample))
fracMissing<-  1- m/n
print( fracMissing)
```

```
## [1] 0.32
```

Check the help file for the **unique** function if you are not familiar with this operation.

Finally, lets generate 2000 bagged samples so we can examine the distribution of the number left out. This uses a **for** loop and saves the values in an array. I like to initialize the array with missing values to start. Note that this example is also a general format for looping over cases and saving the computation.

```r
set.seed( 123)
n<- 50
X<- runif( n)
nBag<- 2000
outOfBagSize<- rep(NA, nBag)

for(  k in 1:nBag){
bagSample<- sample( X,n , replace = TRUE)
m<- length( unique( bagSample))

outOfBagSize[k]<- n-m
}
mean(  outOfBagSize/n)
```

```
## [1] 0.3635
```

```r
exp( -1)
```

```
## [1] 0.3678794
```

Hey! Pretty close to $1/e$! Some more statistics (be sure to load the fields package)

```r
suppressMessages(library( fields))
```

```
## Warning: package 'fields' was built under R version 4.3.3
```

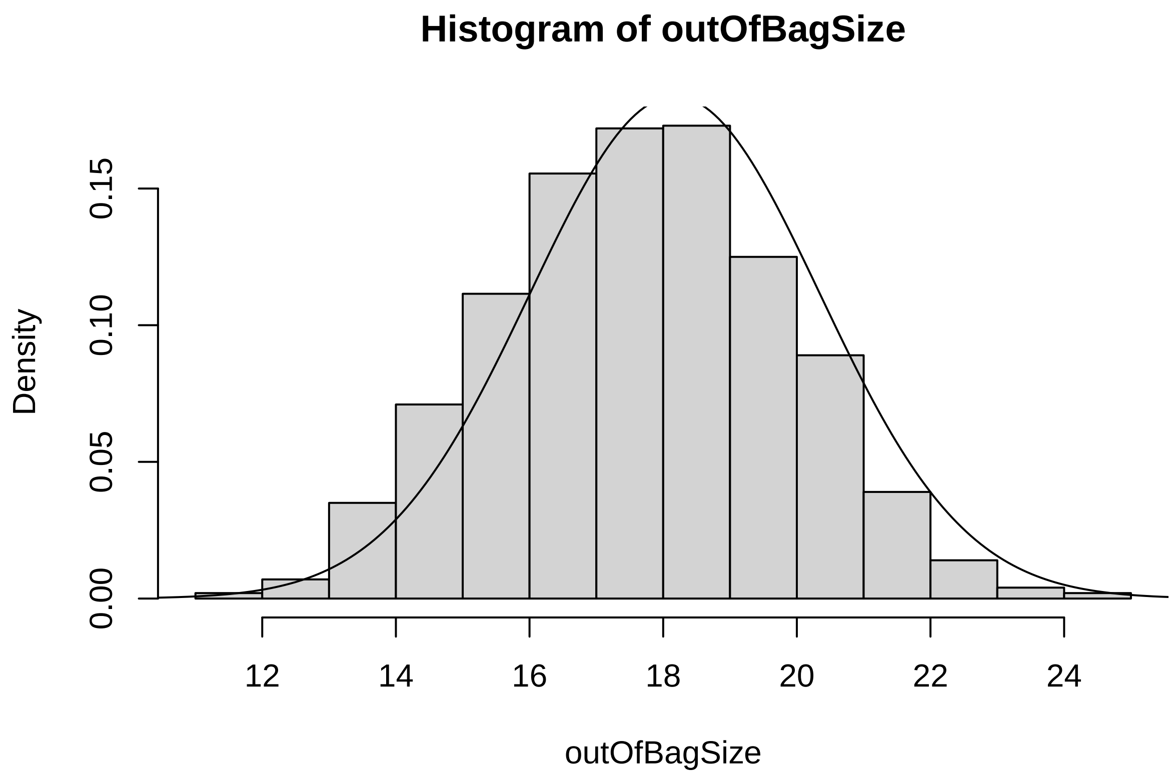
```
## Warning: package 'spam' was built under R version 4.3.3
```

```r
stats( outOfBagSize/n )
```

```
##                        [,1]
## N              2.000000e+03
## mean           3.635000e-01
## Std.Dev.       4.343148e-02
## min            2.200000e-01
## Q1             3.400000e-01
## median         3.600000e-01
## Q3             4.000000e-01
## max            5.000000e-01
## missing values 0.000000e+00
```

#1(a)

```
std=sd(outOfBagSize)
m=mean(outOfBagSize)
xValues<- seq( m - 4*std, m+ 4*std, length.out=250 )
pdf<- dnorm( xValues, mean=m, sd=std)
hist(outOfBagSize,freq=FALSE)
lines(xValues,pdf)
```



The normal distribution fits the data with a slight scew.

#1(b)

```
sd(outOfBagSize)
```

```
## [1] 2.171574
```

```
p=1/exp(1)
sqrt(50*p*(1-p))
```

```
## [1] 3.409869
```

It isn't particularly close.

2 Daily weather measurements for Boulder, CO

```r
load("BoulderDaily.rda")
# and examine first few months of data are missing ...
head(BoulderDaily,4 )
```

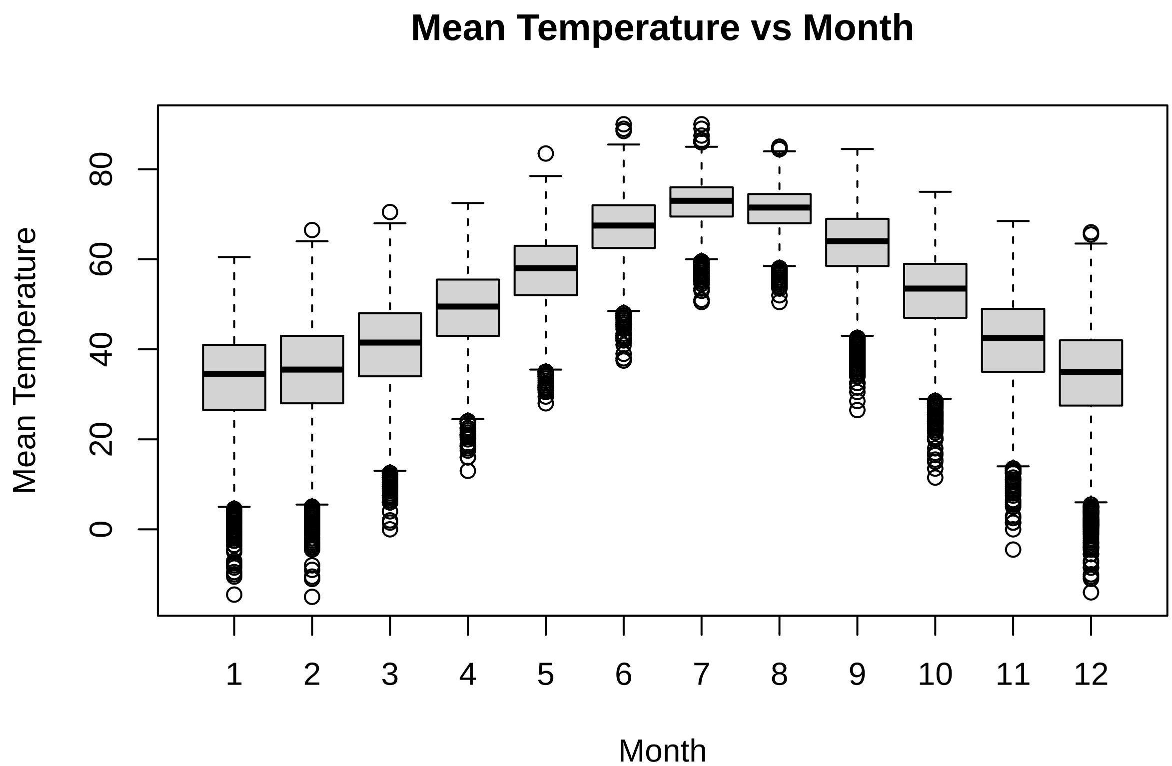
```
##   year month day tmax tmin precip snow snowcover     time tmean       date
## 1 1897     1   1   NA   NA     NA   NA        NA 1897.003    NA 1897-01-01
## 2 1897     1   2   NA   NA     NA   NA        NA 1897.005    NA 1897-01-02
## 3 1897     1   3   NA   NA     NA   NA        NA 1897.008    NA 1897-01-03
## 4 1897     1   4   NA   NA     NA   NA        NA 1897.011    NA 1897-01-04
```

```r
tail( BoulderDaily,4)
```

```
##       year month day tmax tmin precip snow snowcover     time tmean       date
## 45686 2021    10  28   59   34   0.00    0         0 2021.825  46.5 2021-10-28
## 45687 2021    10  29   76   36   0.00    0         0 2021.827  56.0 2021-10-29
## 45688 2021    10  30   76   43   0.00    0         0 2021.830  59.5 2021-10-30
## 45689 2021    10  31   55   36   0.01    0         0 2021.833  45.5 2021-10-31
```

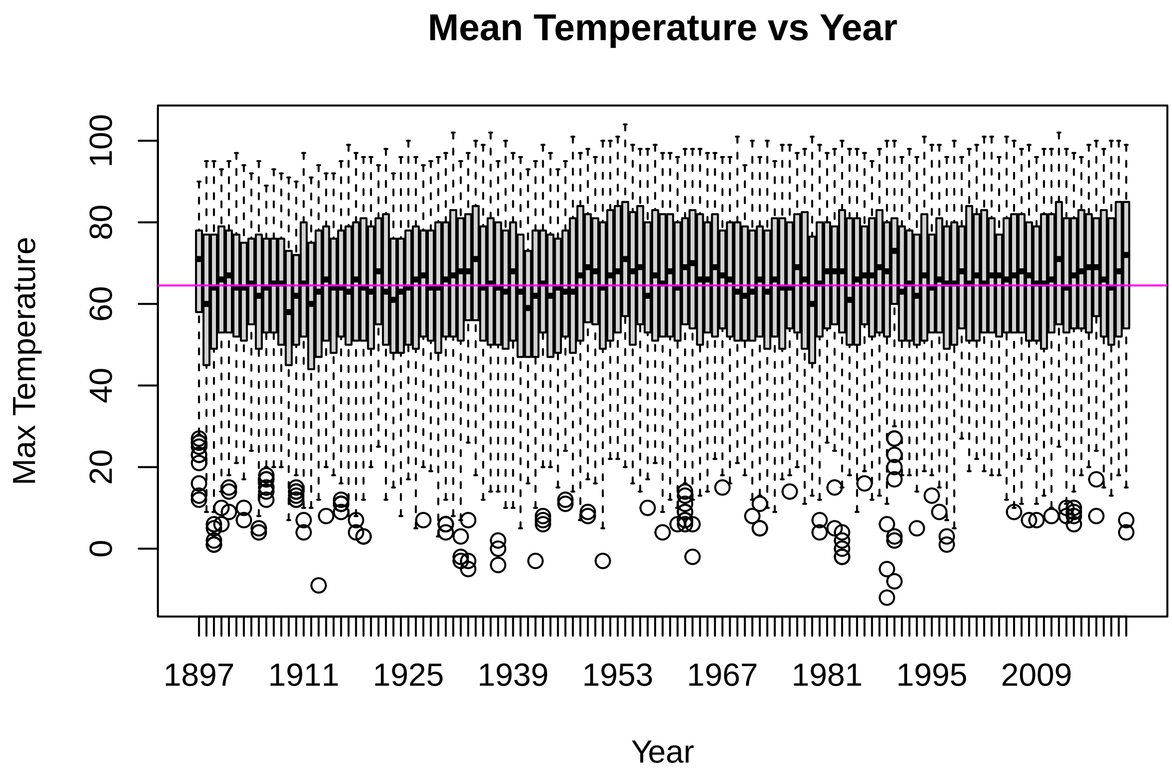
2(a)

```r
 boxplot(BoulderDaily$tmean ~ BoulderDaily$month, title="Mean Tmeperature per Month",xlab="Month",ylab=
title(main="Mean Temperature vs Month")
```



Add a title and axis labels to this plot. Comment on how the distribution changes over the yearly cycle. Are these distributions symmetric or skewed? Which months are the most variable? Which two months have about the same distribution? skewed. The winter motnhs are most variable. 8 and 7 are most similar. 12 and 1 are most similar. # 2(b)

```
boxplot(BoulderDaily$tmax ~ BoulderDaily$year, title="Max Tmeperature per Year",xlab="Year",ylab="Max '
title(main="Mean Temperature vs Year")
abline( h= mean( BoulderDaily$tmax, na.rm=TRUE), col="magenta")
```

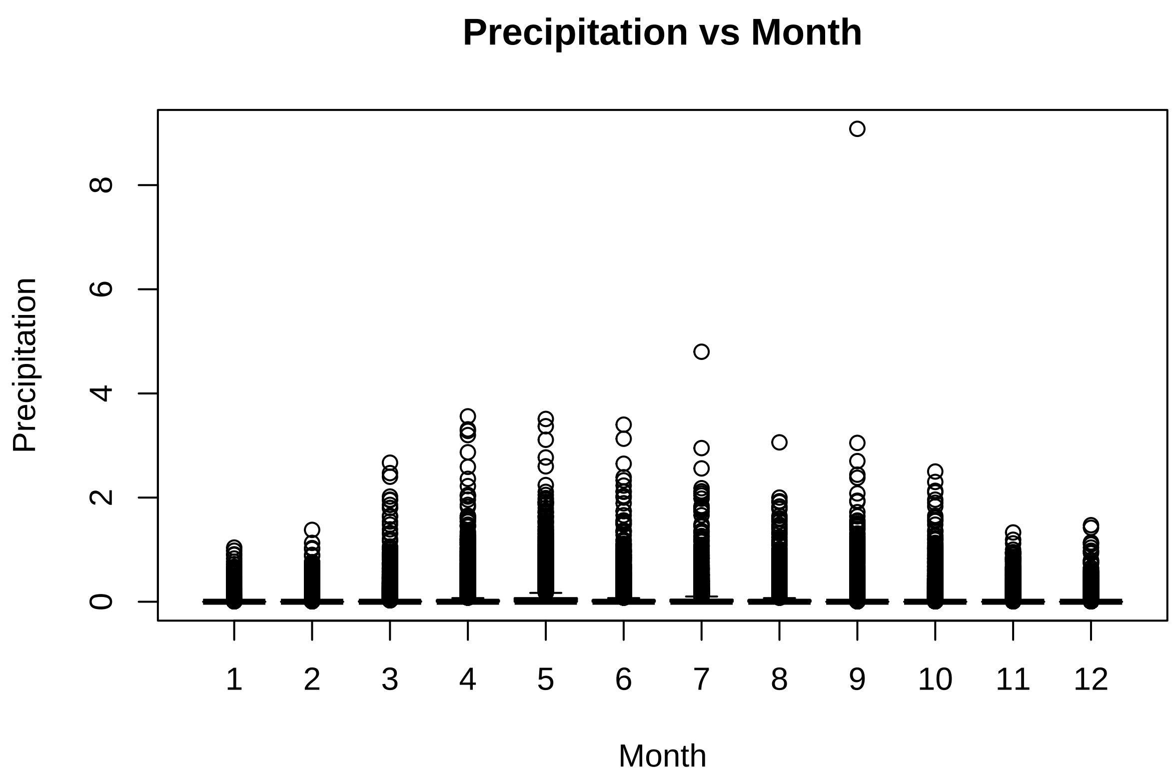


There is no discernable temperature increase trend.

2(c)

This data is highly scewed because Colorado is a desert. We might be able to normalize the data so that we can study the scew. Perhaps taking the log of the data would help.

```
boxplot(BoulderDaily$precip ~ BoulderDaily$month, title="Precipitation Per Month",xlab="Month",ylab="Pr
title(main="Precipitation vs Month")
```


Precipitation vs Month
Precipitation
0
2
4
6
8
1
2
3
4
5
6
7
8
9
10
11
12
Month

3 Correlation in daily maximum temperatures

```r
N<- nrow(BoulderDaily )
# the first value is missing because it the day before # the data starts
tmaxLag1<- c(NA, BoulderDaily$tmax[1: (N-1)])
BoulderDaily$tmaxLag1<- tmaxLag1
# to check compare the values from days 201 to 205
index<- 201:205
BoulderDaily[index,]
```

```
##     year month day tmax tmin precip snow snowcover     time tmean       date
## 202 1897     7  20   74   48   0.00   NA        NA 1897.551  61.0 1897-07-20
## 203 1897     7  21   81   48   0.00   NA        NA 1897.553  64.5 1897-07-21
## 204 1897     7  22   86   62   0.00   NA        NA 1897.556  74.0 1897-07-22
## 205 1897     7  23   85   63   0.03   NA        NA 1897.559  74.0 1897-07-23
## 206 1897     7  24   74   61   0.24   NA        NA 1897.562  67.5 1897-07-24
##     tmaxLag1
## 202       67
## 203       74
## 204       81
## 205       86
## 206       85
```

3(a)

```r
cor( BoulderDaily$tmaxLag1 , BoulderDaily$tmax, use="complete" )
```

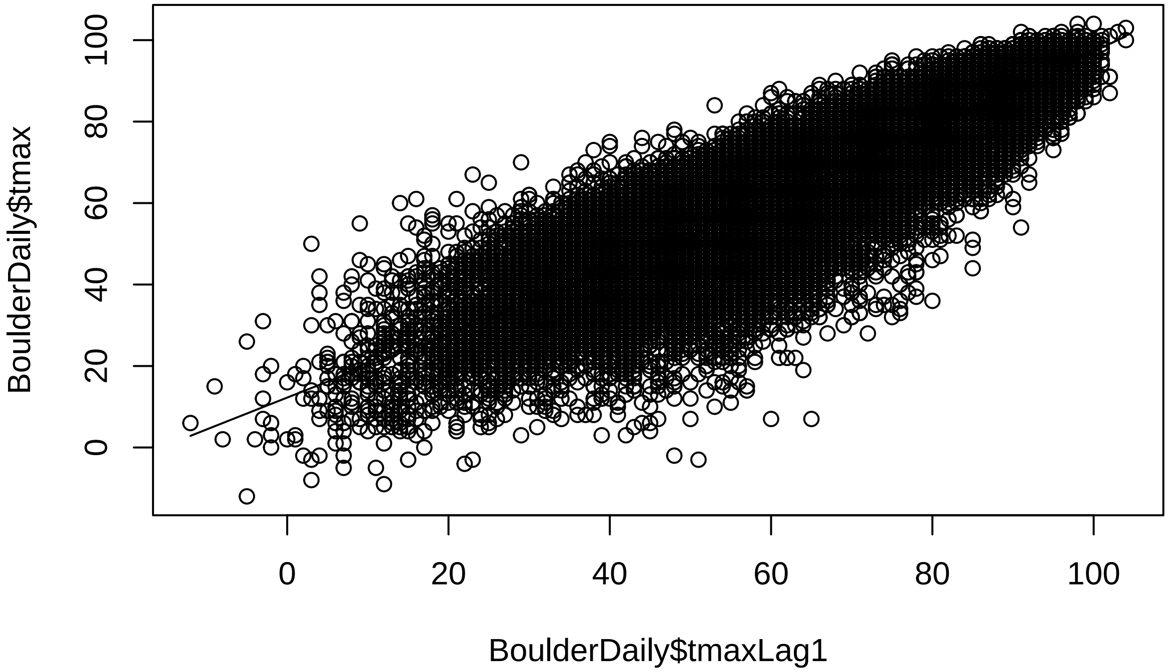
```
## [1] 0.8868995
```

The data is strongly correlated linearly.

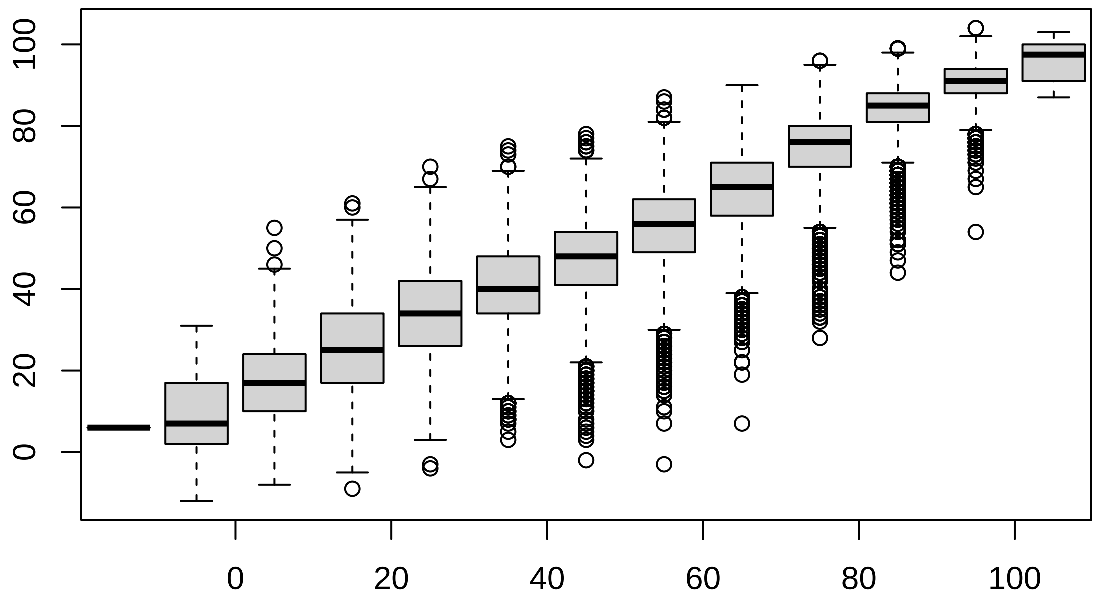
3(b)

This data appears linear

```r
library( fields)
scatter.smooth(BoulderDaily$tmaxLag1, BoulderDaily$tmax)
```




```
bplot.xy( BoulderDaily$tmaxLag1,BoulderDaily$tmax,
N=10)
```



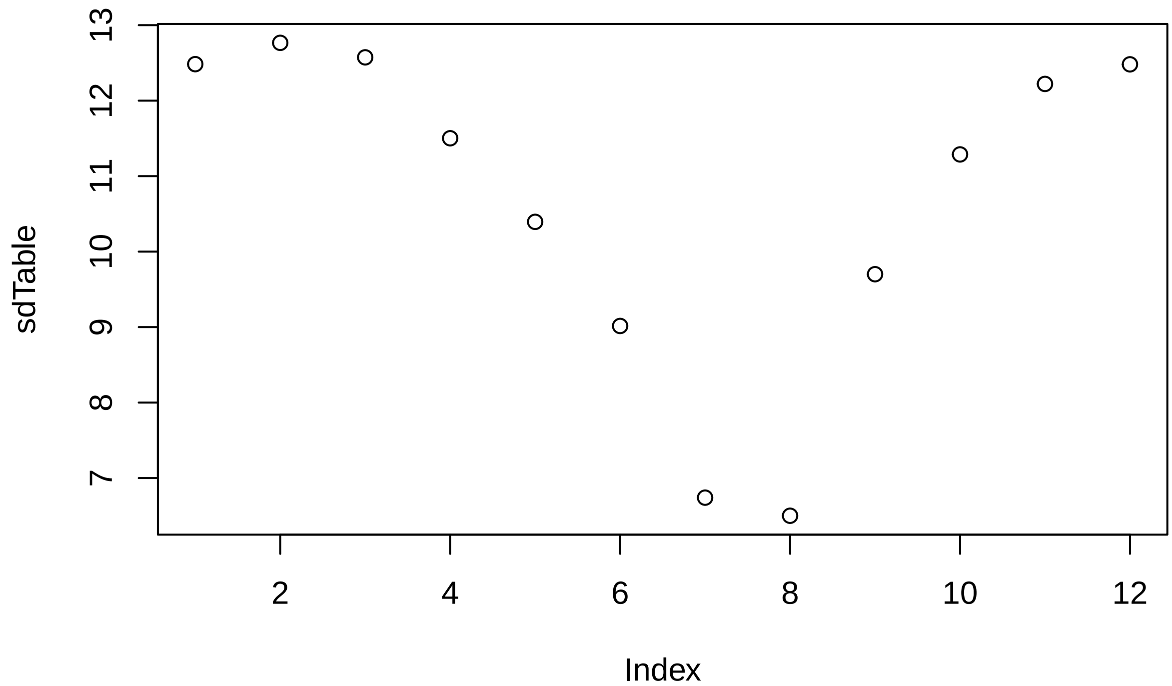
Ther dtaa towards the beginning and end can be more accurately predicted. Its hard to tell this from the scatter plot.

3(c)

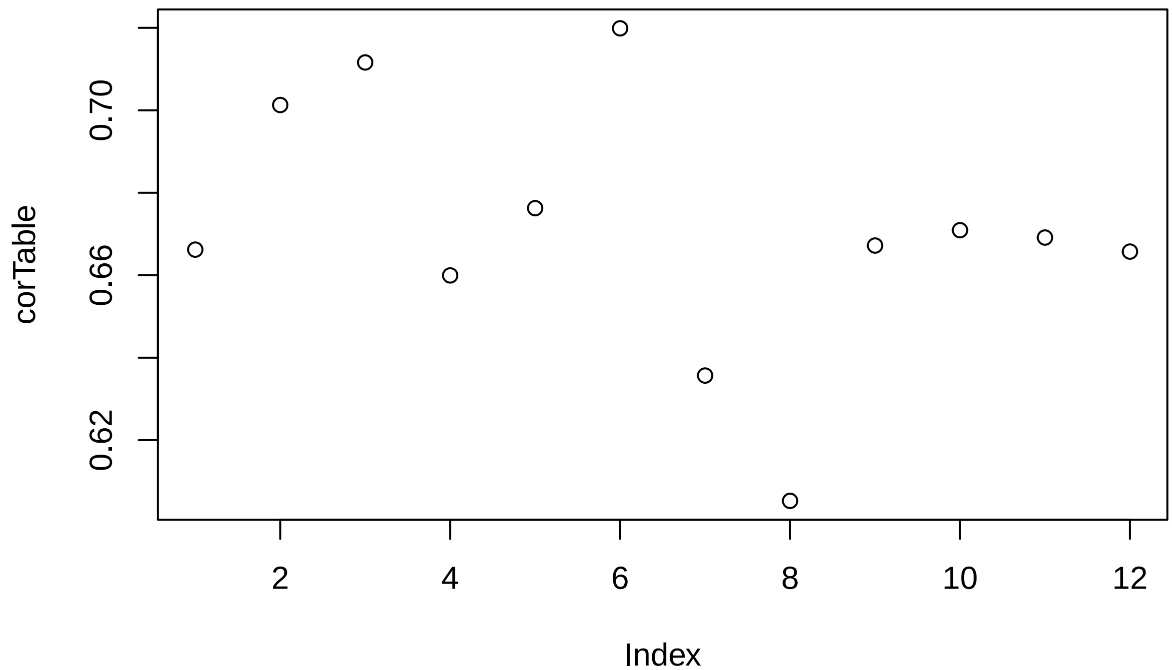
```
corTable<- rep( NA, 12)
sdTable<- rep( NA, 12)
  for( k in 1:12){
    ind<- which( BoulderDaily$month==k )
      corTable[k]<- cor( BoulderDaily$tmaxLag1[ind] ,
                           BoulderDaily$tmax[ind],
                           use="complete" )
      sdTable[k]<- sd( BoulderDaily$tmax[ind], na.rm=TRUE )
  }
sdTable
```

```
##  [1] 12.483580 12.765799 12.573728 11.501789 10.394411  9.015061  6.740491
##  [8]  6.500901  9.700843 11.287805 12.221630 12.481262
```

```
plot(sdTable)
```




```
plot(corTable)
```



The seventh, and eighth months are most easy to predict with a linear relationship.

Extra Credit: We are able to have seperate correlation coefficients for each mont with different slopes so that the data is easier to explain month by month.