Homework 1

Kendall Brown 8564403

Fall 2018

```
algae = read.table("algae.txt", header=T, na.strings="NA")
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

1abc. Counting observations of algae and comparing mean v median and variance v MAD.

```
#1a.
Datasum=summarise(algae,count=n())
Datasum##number of obs in algae data set
```

```
##   count
## 1   180
```

```
#1b.
?summarise
?summarise_all
?summarise_at
w=algae%>%summarise_at(c(6:11),mean,na.rm=T)
w
```

```
##        Cl      NO3      NH4     oPO4      PO4     Chla
## 1 40.2932 2.946356 414.1545 75.24621 133.2588 10.62003
```

```
x=algae%>%summarise_at(c(6:11),var,na.rm=T)
x
```

```
##         Cl     NO3     NH4     oPO4      PO4     Chla
## 1 1797.325 5.40628 1240777 9421.302 18136.13 238.7067
```

```
y=algae%>%summarise_at(c(6:11),median,na.rm=T)
y
```

```
##     Cl   NO3      NH4   oPO4   PO4 Chla
## 1 29.5 2.262 103.0415 35.928 85.95  4.5
```

```
z=algae%>%summarise_at(c(6:11),mad,na.rm=T)
z
```

```
##        Cl      NO3      NH4     oPO4      PO4     Chla
## 1 31.8759 2.095655 115.0868 44.37125 102.9206 5.355892
```

```
w>y ##Test if mean > median
```

```
##        Cl  NO3  NH4 oPO4  PO4 Chla
```

```
## [1,] TRUE TRUE TRUE TRUE TRUE TRUE
```

```
x>z ##Test if var > mad
```

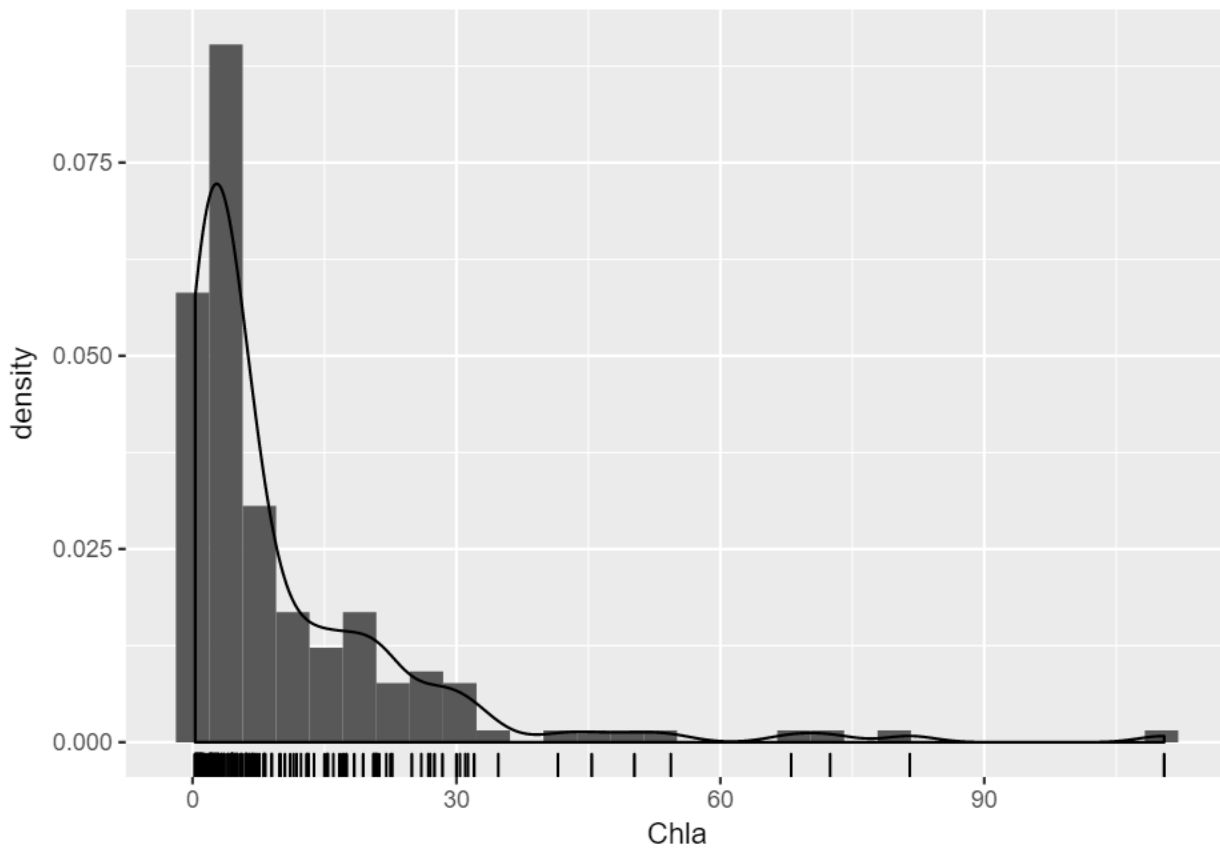
```
##        Cl  NO3  NH4 oPO4  PO4 Chla
## [1,] TRUE TRUE TRUE TRUE TRUE TRUE
```

What stands out most is that some chemicals have a rather large variance. Additionally each chemical seems to vary in mean quite significantly as well. 1c. The constant in the mad() function ensures consistency when calculating the median of a data set.

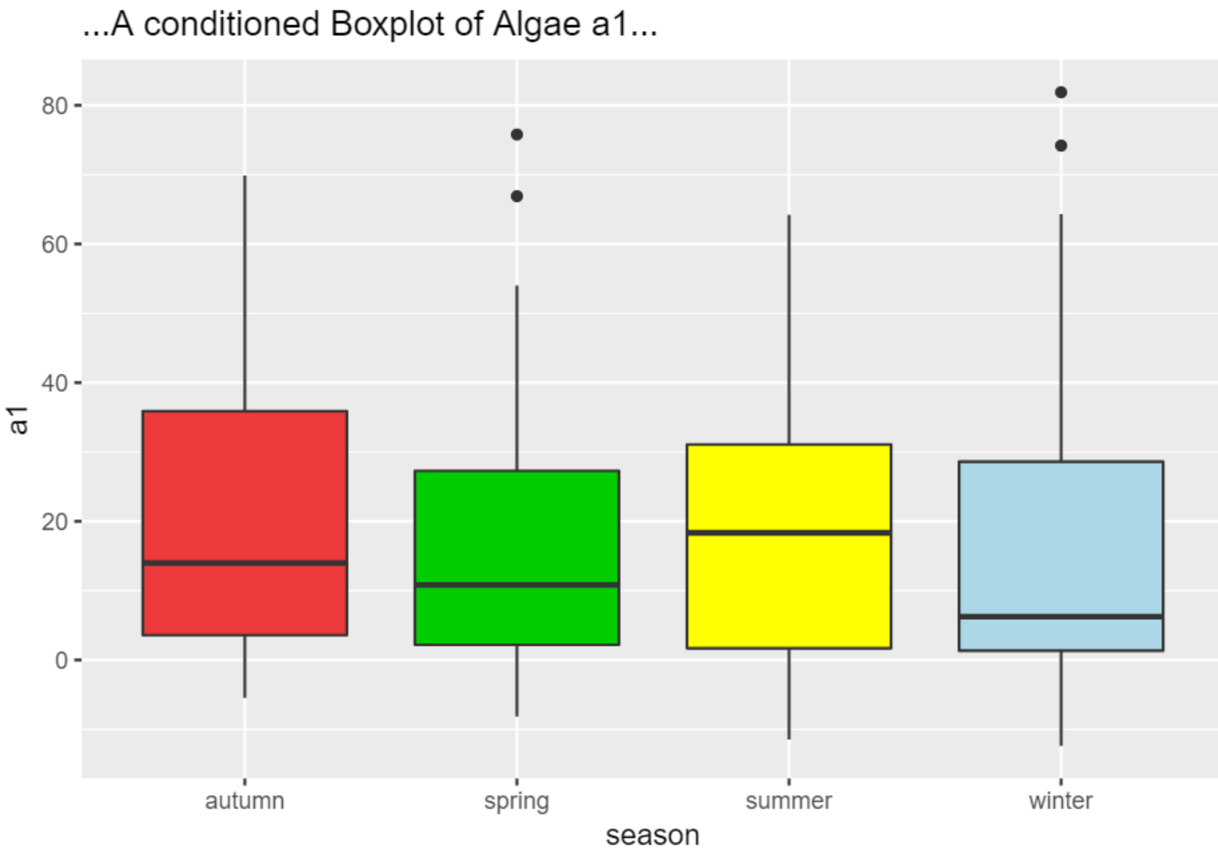
2abc. Altered histogram and boxplot of algae data set.

```
library(ggplot2)
ggplot(algae,aes(Chla,..density..))+
  geom_histogram(na.rm=T)+
  geom_density(na.rm = T)+
  geom_rug(aes(Chla),na.rm=T,inherit.aes = F)
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

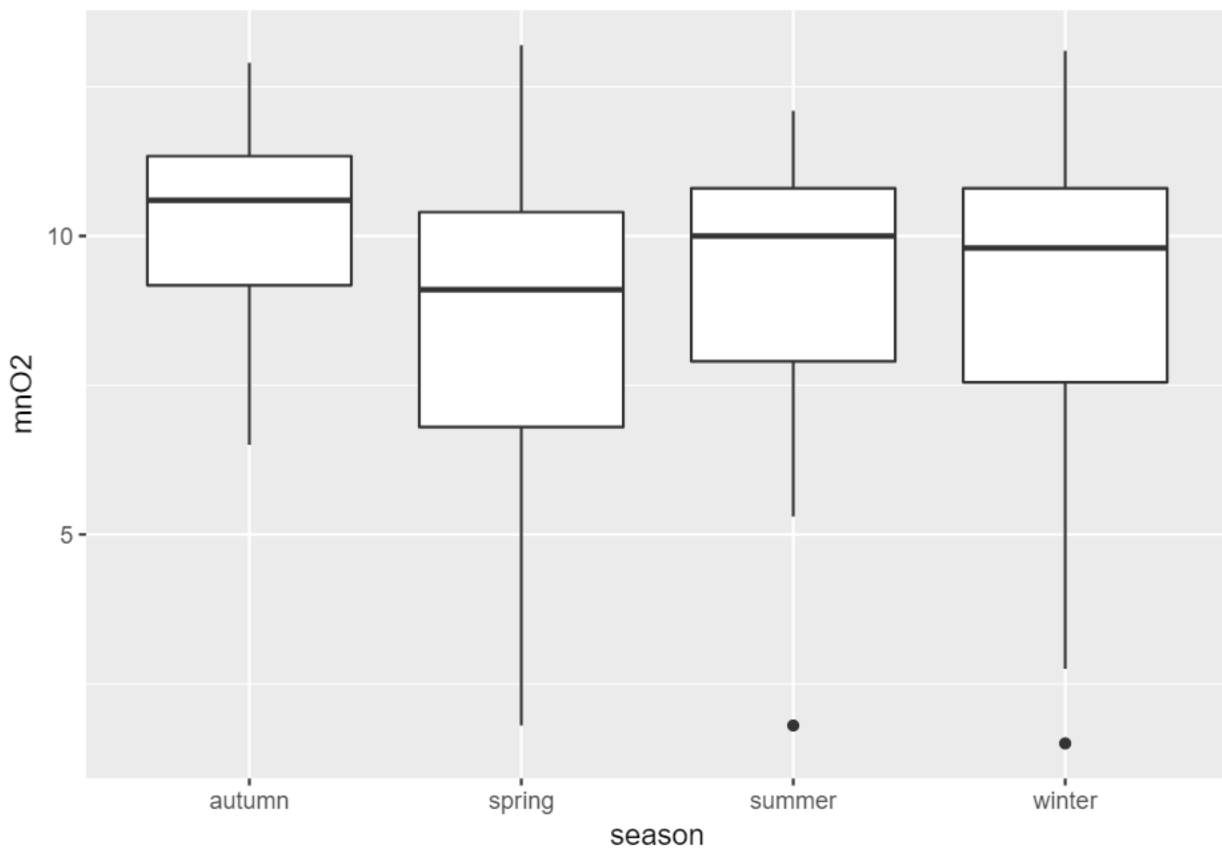



```
ggplot(algae,aes(season,a1))+
  geom_boxplot(na.rm=T,fill=c("brown2","green3","yellow","lightblue"))+
  labs(title="'A conditioned Boxplot of Algae a1'")
```

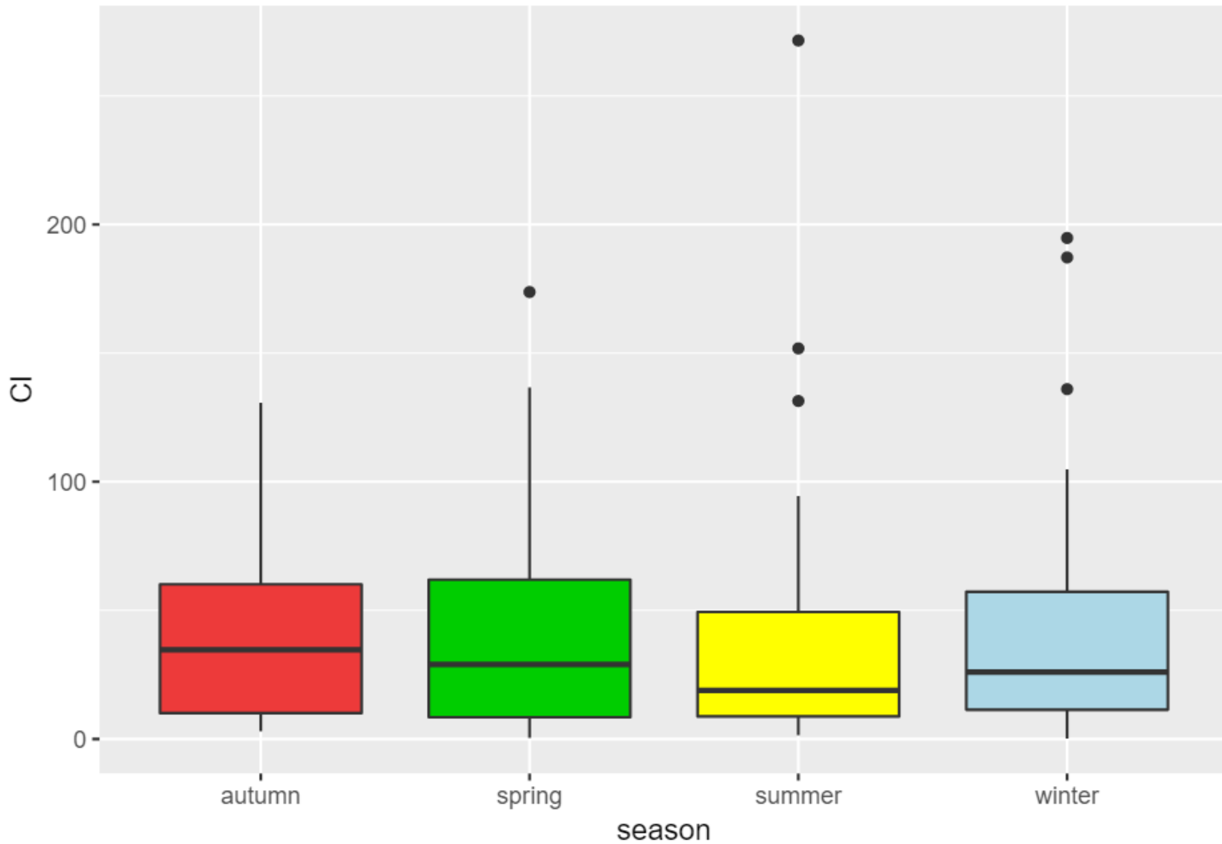


2d.

```
ggplot(algae,aes(season,mnO2))+
  geom_boxplot(na.rm=T)
```




```r
ggplot(algae,aes(season,Cl))+
  geom_boxplot(na.rm=T,fill=c("brown2","green3","yellow","lightblue"))
```



From the above boxplots we observe 2 seasonal outliers during the summer and winter when measuring mn02, and an additional 7 seasonal outliers when measuring Cl. Boxplots are a good way of finding outliers as they isolate the values that are significantly above or below the data set mean.

2e.Comparing mean v median and var v MAD

```
w1=algae%>%summarise_at(c(9,10),mean,na.rm=T)
w1##Mean of oPO4 and PO4
```

```
##        oPO4      PO4
## 1 75.24621 133.2588
```

```
x1=algae%>%summarise_at(c(9,10),var,na.rm=T)
x1##Var of oPO4 and PO4
```

```
##        oPO4      PO4
## 1 9421.302 18136.13
```

```
y1=algae%>%summarise_at(c(9,10),median,na.rm=T)
y1##Median of oPO4 and PO4
```

```
##      oPO4   PO4
## 1 35.928 85.95
```

```
z1=algae%>%summarise_at(c(9,10),mad,na.rm=T)
z1##MAD of oPO4 and PO4
```

```
##        oPO4      PO4
## 1 44.37125 102.9206
```

We clearly see that the mean and median, along with the variance and mad, differ greatly. This implies that outliers are providing a rather significant skew when calculating the mean and variance of the two chemicals.

3a.Number of observations with missing values

```
na.test=is.na(algae)*1
colSums(na.test)
```

```
## season   size  speed   mxPH   mnO2     Cl    NO3    NH4   oPO4    PO4
##      0      0      0      1      1      7      0      0      0      0
##   Chla     a1
##      8      0
```

We observe that 4 variables have missing values. mxPh and mnO2 have 1 missing value each, Cl has 7 missing values, and Chla has 8 missing values.

3b.Removing observations with missing values with filter()

```
algae.del=filter(algae,is.na(mxPH)==0&is.na(mnO2)==0&is.na(Cl)==0&is.na(Chla)==0)
algae.cc=complete.cases(algae.del)
sum(algae.cc)
```

```
## [1] 169
```

Algae.del is a data set with 169 complete cases. 3c.Imputing unknowns with measures of central tendency

```
algae.mean=algae
for(j in c(1:180)){
for(i in c(4:12)){
  pre.am=as.data.frame(algae[j,i])
  pre.algae.mean=mutate_if(pre.am,is.double,funs(ifelse(is.na(pre.am),mean(algae[,i],na.rm=T),algae[j,i
  algae.mean[j,i]=pre.algae.mean
}
}
summarise(algae.mean, count=n())
```

```
##   count
## 1   180
```

```
algae.mean[c(70,117,180),4:12]
```

```
##     mxPH mnO2      Cl  NO3 NH4 oPO4 PO4     Chla   a1
## 70   6.6 11.3 40.2932 4.17  10    1   6 10.62003 47.1
## 117  6.6 10.8 40.2932 2.64  10    2  11 10.62003 46.9
## 180  5.7 10.8 40.2932 2.55  10    1   4 10.62003 16.8
```

3d. Imputing unknowns using correlation

```
cor(algae[,4:12],use="complete.obs")
```

```
##             mxPH        mnO2         Cl         NO3         NH4
## mxPH  1.00000000 -0.02913045  0.1557075  0.03339517 -0.12005457
## mnO2 -0.02913045  1.00000000 -0.3348301 -0.03292011 -0.28278234
## Cl    0.15570753 -0.33483005  1.0000000  0.45590366  0.16225274
## NO3   0.03339517 -0.03292011  0.4559037  1.00000000  0.14587533
## NH4  -0.12005457 -0.28278234  0.1622527  0.14587533  1.00000000
## oPO4  0.04653433 -0.39999298  0.3968281  0.30279529  0.56762718
## PO4   0.04409164 -0.48414176  0.4710502  0.32852261  0.62113218
## Chla  0.43879072 -0.16058108  0.1677026  0.06764031 -0.03901627
## a1   -0.23274670  0.27512691 -0.4145950 -0.37398729 -0.17565791
##             oPO4         PO4       Chla          a1
```

```
## mxPH  0.04653433  0.04409164  0.43879072 -0.2327467
## mnO2 -0.39999298 -0.48414176 -0.16058108  0.2751269
## Cl    0.39682805  0.47105019  0.16770264 -0.4145950
## NO3   0.30279529  0.32852261  0.06764031 -0.3739873
## NH4   0.56762718  0.62113218 -0.03901627 -0.1756579
## oPO4  1.00000000  0.93253518  0.04982285 -0.4200985
## PO4   0.93253518  1.00000000  0.15925064 -0.4508198
## Chla  0.04982285  0.15925064  1.00000000 -0.2925959
## a1   -0.42009846 -0.45081979 -0.29259589  1.0000000
```

```
fit1=lm(Chla~mxPH,data=algae)
coefficients(fit1)
```

```
## (Intercept)        mxPH
##  -118.88773    16.04646
```

```
predict(fit1,data.frame(mxPH=c(5.7,6.5,6.6,7.83,9.7)))
```

```
##          1          2          3          4          5
## -27.422907 -14.585738 -12.981092   6.756055  36.762936
```

```
#Obtained values when mxPH is measured to be 5.7,6.5,6.6,7.83,and 9.7
```

These are odd numbers to obtain since some of them are negative.

4a. Partitioning of data set algae.mean

```
set.seed(10)
cut.id=cut(1:180,6,labels = F)
cut.id.rand=sample(cut.id,180)
algae.sample.id=cut(cut.id.rand,6,labels=F)
```

4b. 6 fold cross validation

```
library(plyr)
```

```
## ------------------------------------------------------------------------------

## You have loaded plyr after dplyr - this is likely to cause problems.
## If you need functions from both plyr and dplyr, please load plyr first, then dplyr:
## library(plyr); library(dplyr)

## ------------------------------------------------------------------------------

##
## Attaching package: 'plyr'

## The following objects are masked from 'package:dplyr':
##
##     arrange, count, desc, failwith, id, mutate, rename, summarise,
##     summarize
```

```
do.chunk <- function(chunkid, chunkdef, dat){ # function argument
train = (chunkdef != chunkid)
Ytr = dat[train,]$a1 # get true response values in training set
Yvl = dat[!train,]$a1 # get true response values in validation set
lm.a1 <- lm(a1~., data = dat[train,])
predYtr = predict(lm.a1) # predict training response values
predYvl = predict(lm.a1,dat[!train,]) # predict validation values
data.frame(fold = chunkid,
train.error = mean((predYtr - Ytr)^2), # compute and store training errors
```

```
val.error = mean((predYvl - Yvl)^2)) # compute and store validation errors
}
cv.6f=ldply(1:6,do.chunk,algae.sample.id,algae.mean)
cv.6f
```

```
##   fold train.error val.error
## 1    1    231.8446  466.6458
## 2    2    267.3170  264.8416
## 3    3    260.2019  295.8210
## 4    4    247.8380  383.4553
## 5    5    270.7286  247.7628
## 6    6    274.1990  233.6241
```

5.Testing algae.test file.

```
algaeTest = read.table("algae-test.txt", header=T, na.strings="NA")
```

```
val.error.avg=mean(cv.6f[,3])
val.error.avg##average of valuation error
```

```
## [1] 315.3584
```

```
lm.a1.2=lm(a1~.,data=algae.mean)
```

```
a1.test.pred=predict(lm.a1.2,algaeTest)
a1.test.pred
```

```
##           1           2           3           4           5           6
##   7.4336194   0.8124484  15.1236371  21.7018260  25.4605555  35.8558252
##           7           8           9          10          11          12
##  28.5927301  34.1002476  34.4933167  29.1322886          NA  26.0753508
##          13          14          15          16          17          18
##  41.5008572  23.8306777  37.9088291  14.4221764  10.1389309  26.7856776
##          19          20          21          22          23          24
##  28.0539124  31.3413817  28.3401972  30.3474327          NA          NA
##          25          26          27          28          29          30
##  36.9906760  40.0117574  38.0789920  15.2655612   8.5607639  24.9025220
##          31          32          33          34          35          36
##  21.9392482  25.2630485  24.1665620  19.3906134  15.8554695  18.5614626
##          37          38          39          40          41          42
##  -6.3539289  -4.3988611   8.6456563  -6.2158181   6.0432748  17.2266901
##          43          44          45          46          47          48
##  -1.5874397  18.5975122  -6.5439746  15.9606574  14.1833677  23.3995388
##          49          50          51          52          53          54
##  23.5475449  29.4925914   3.0469893  26.4815484  20.7319677  -7.7054715
##          55          56          57          58          59          60
## -12.6053118  23.6375182  22.5877617  19.2731724  20.2494723  10.1283481
##          61          62          63          64          65          66
##  16.8493796 -34.3547492  16.6198052  -4.8191644   2.1351614 -10.0340883
##          67          68          69          70          71          72
##  -8.4101419  22.1874309  15.3240174   4.5598152  11.4773656  -3.3972793
##          73          74          75          76          77          78
##  25.7980437  25.9591162  25.3244379          NA  25.1336833  -1.1539189
##          79          80          81          82          83          84
##  15.2409406  15.6371562  12.8505710  19.5443274  12.5991842  16.7327584
##          85          86          87          88          89          90
```

```
##   3.6777198          NA   7.7334436  17.6224599  23.1728851  36.7552461
##          91          92          93          94          95          96
## -54.6861243  38.4101709  37.9127800  29.8926220  35.3070955  10.7771792
##          97          98          99         100         101         102
##  14.2536802  -1.3950356  -6.0126564  24.9235553  43.4589283  32.7516646
##         103         104         105         106         107         108
##  35.3140973  40.3927125  39.2078702  36.7025597  38.7128532  35.9748090
##         109         110         111         112         113         114
##  39.0257476  19.0012175  19.3287288   8.4565746   7.7849182  13.4386025
##         115         116         117         118         119         120
##   0.9875589   4.7262051  16.1004242   1.6524832   6.2715781   6.1175048
##         121         122         123         124         125         126
##  17.9821767  17.7405451  25.9169313  24.0959127  23.7219232  28.7406761
##         127         128         129         130         131         132
##   5.0717693   2.3554747   1.9753651  21.5741983  22.2987850  20.7516918
##         133         134         135         136         137         138
##   4.3060994   6.9549463  -1.9994972  12.7915820  15.3610764  18.4532987
##         139         140         141         142         143         144
##  17.1504364  14.8373311  10.3433198  14.5213188  15.8886815  -2.0861936
##         145         146         147         148         149         150
##   0.6774467  -7.7484079   0.3642028  25.5103241  14.3845324  18.0971744
##         151         152         153         154         155         156
##  14.5800206  13.6600917  30.1768521  35.1738197  35.1270354  23.1504807
##         157         158         159         160
##  38.5180005  25.2949296  25.0570888  16.2703687
```

```
train.error.test=mean((a1.test.pred-algaeTest$a1)^2,na.rm=T)
train.error.test##test error from algaeTest
```

```
## [1] 290.6248
```

```
val.error.avg##Test error from 6 fold C.V
```

```
## [1] 315.3584
```

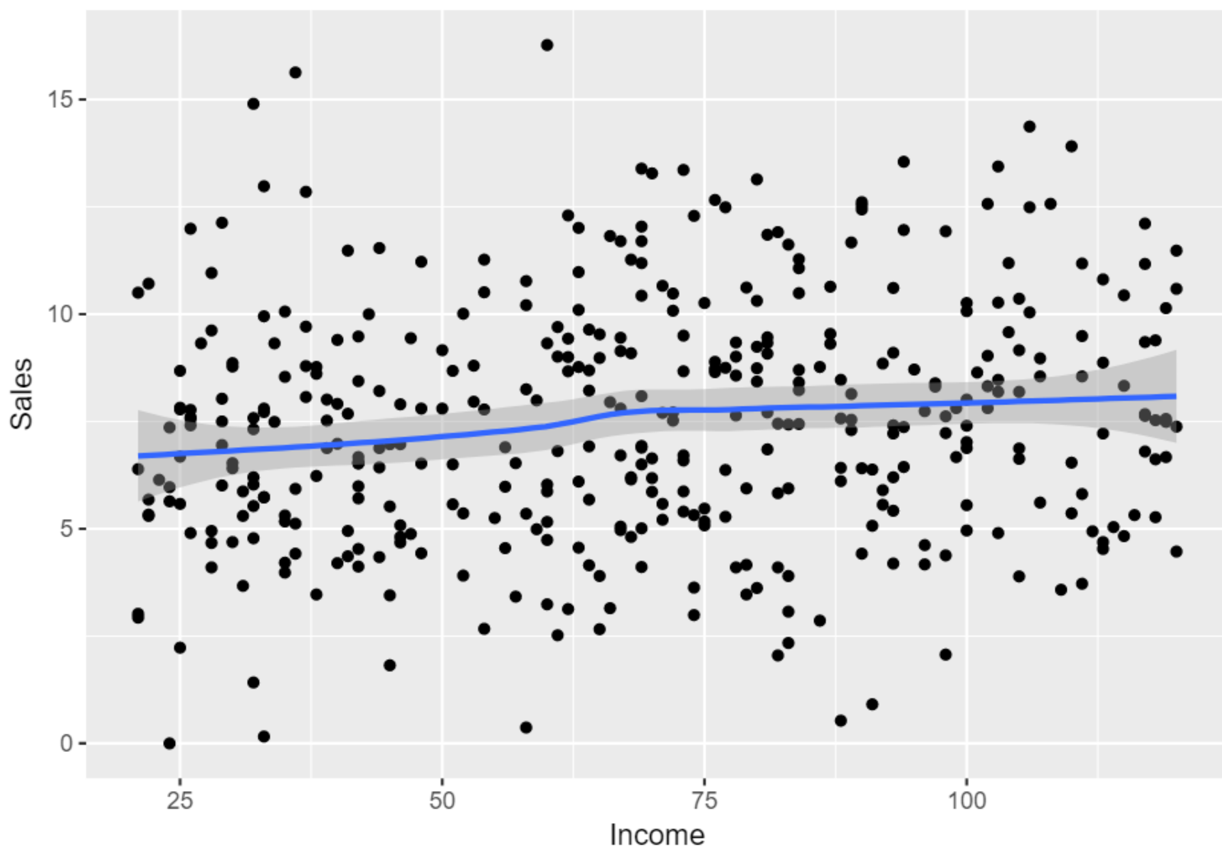
The test error of the two data sets are rather similar but still vary slightly. Considering the number of predictor variables, their possible interactions, and total observations, this result is expected.

6a.

```
library(ISLR)
data(Carseats)
attach(Carseats)
```

```
ggplot(Carseats,aes(Income,Sales))+
  geom_point()+
  geom_smooth()
```

```
## `geom_smooth()` using method = 'loess'
```



From this plot there does not appear to be a significantly large relation between sales and income. This matches my intuition as I believe child car seats are long lasting and are often reused between children. I believe carseats are seldomly repeatingly purchased by individual customers unless required. I assume carseat sales have more to do with measurements related to the number of children in a given area than it does with income.

6b. Fitting linear models to the p-th degree of Income and running a 6 fold C.V

```
fit.sales.inc=lm(Sales~poly(Income,10,raw=F),data=Carseats)
summary(fit.sales.inc)
```

```
##
## Call:
## lm(formula = Sales ~ poly(Income, 10, raw = F), data = Carseats)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -7.0727 -1.9203 -0.1168  1.7017  8.9917
##
## Coefficients:
##                             Estimate Std. Error t value Pr(>|t|)
## (Intercept)                  7.49633    0.14026  53.445   <2e-16 ***
## poly(Income, 10, raw = F)1   8.57181    2.80527   3.056   0.0024 **
## poly(Income, 10, raw = F)2  -1.90088    2.80527  -0.678   0.4984
## poly(Income, 10, raw = F)3  -0.22592    2.80527  -0.081   0.9359
## poly(Income, 10, raw = F)4  -0.45907    2.80527  -0.164   0.8701
## poly(Income, 10, raw = F)5   1.39137    2.80527   0.496   0.6202
## poly(Income, 10, raw = F)6  -3.89104    2.80527  -1.387   0.1662
```

```
## poly(Income, 10, raw = F)7   2.04818    2.80527   0.730   0.4658
## poly(Income, 10, raw = F)8   4.39531    2.80527   1.567   0.1180
## poly(Income, 10, raw = F)9   1.75371    2.80527   0.625   0.5322
## poly(Income, 10, raw = F)10 -0.07317    2.80527  -0.026   0.9792
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.805 on 389 degrees of freedom
## Multiple R-squared:  0.03803,    Adjusted R-squared:  0.0133
## F-statistic: 1.538 on 10 and 389 DF,  p-value: 0.1237
```

```
set.seed(10)
cut.id.5=cut(1:400,6,labels = F)
cut.id.rand.5=sample(cut.id.5,400)
carseats.sample.id=cut(cut.id.rand.5,6,labels=F)
carseats.mut=mutate(Carseats,SampleID=carseats.sample.id)
```

6 Fold Cross Validation Function

```
do.chunk5 <- function(chunkid, chunkdef, dat, p){# function argument
train = (chunkdef != chunkid)
res = data.frame(degree=integer(), fold=integer(),
train.error=double(), val.error=double())
if (p==0) {
## Your code here
  Ytr.5=dat[train,]$Sales
  Yvl.5=dat[!train,]$Sales
## Fit an intercept only model to the data.
## Using poly(Income, degree=0) will cause an error
  sales.incp=lm(Sales~1,data=dat[train,])
  PYtr.5=predict(sales.incp,dat[train,])
  PYvl.5=predict(sales.incp,dat[!train,])
## Update residual
    res = data.frame(degree=p, fold=chunkid,
    train.error=mean((PYtr.5-Ytr.5)^2),
    val.error=mean((PYvl.5-Yvl.5)^2))
    res
} else {
## Your code here
  Ytr.5=dat[train,]$Sales
  Yvl.5=dat[!train,]$Sales
## Fit a polynomial regression or order p.
## Use poly(Income, degree=p, raw=FALSE)
  sales.incp=lm(Sales~poly(Income,p),data=dat[train,])
  PYtr.5=predict(sales.incp,dat[train,])
  PYvl.5=predict(sales.incp,dat[!train,])
## Update residual
  res = data.frame(degree=p, fold=chunkid,
  train.error=mean((PYtr.5-Ytr.5)^2),
  val.error=mean((PYvl.5-Yvl.5)^2))
  res
}
}
```

Returning each CV of carseats data set

```
cvlist=c()
mve=rep(0,11)
mte=rep(0,11)
for(i in c(0:10)){
cv.6f.carseats=ldply(1:6,do.chunk5,carseats.sample.id,Carseats,i)
mve[i+1]=mean(cv.6f.carseats$val.error)
mte[i+1]=mean(cv.6f.carseats$train.error)
print(cv.6f.carseats)
cvlist=append(cvlist,cv.6f.carseats)
}
```

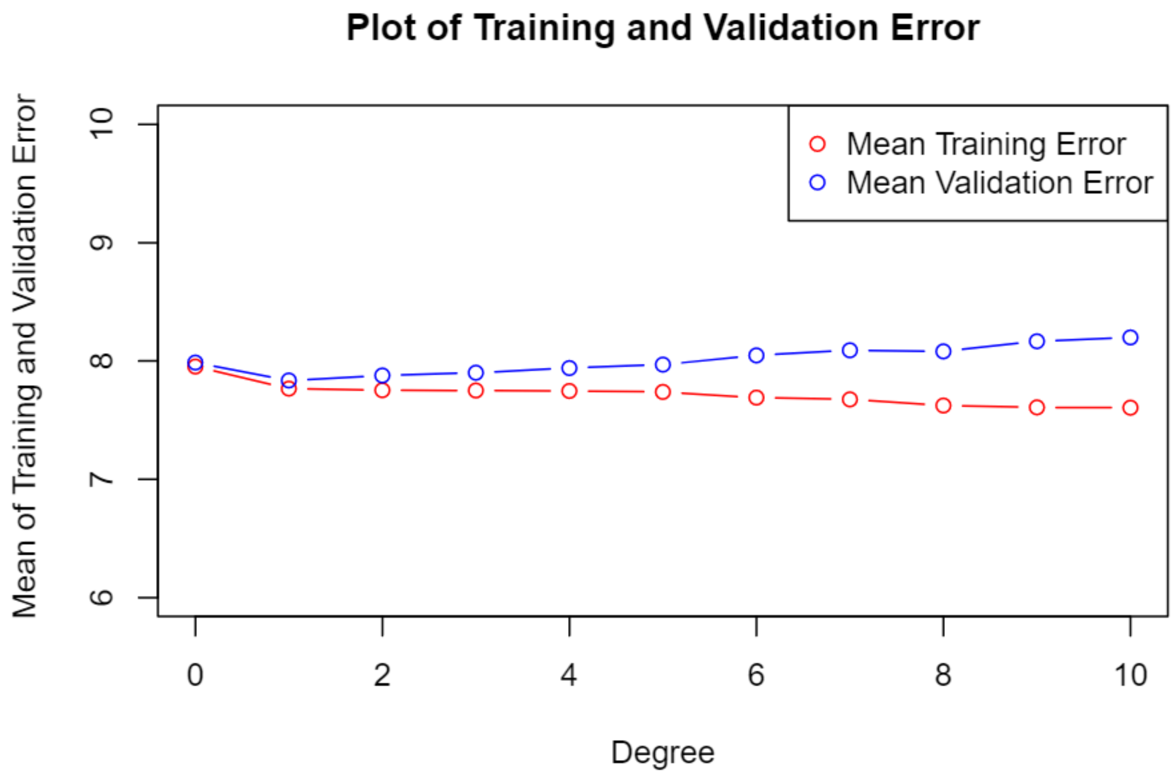
```
##   degree fold train.error val.error
## 1      0    1    8.518380  5.163070
## 2      0    2    7.966481  7.902722
## 3      0    3    7.756241  8.989137
## 4      0    4    7.526500 10.095775
## 5      0    5    7.809419  8.699561
## 6      0    6    8.143375  7.071808
##   degree fold train.error val.error
## 1      1    1    8.373222  4.798782
## 2      1    2    7.782577  7.719448
## 3      1    3    7.524387  9.073789
## 4      1    4    7.317815 10.035983
## 5      1    5    7.700618  8.195184
## 6      1    6    7.901821  7.189609
##   degree fold train.error val.error
## 1      2    1    8.372869  4.781148
## 2      2    2    7.762206  7.781148
## 3      2    3    7.495361  9.196735
## 4      2    4    7.290461 10.148359
## 5      2    5    7.695717  8.167896
## 6      2    6    7.901820  7.190583
##   degree fold train.error val.error
## 1      3    1    8.371449  4.800467
## 2      3    2    7.759393  7.803331
## 3      3    3    7.495351  9.196521
## 4      3    4    7.290161 10.153321
## 5      3    5    7.687124  8.255757
## 6      3    6    7.901481  7.197431
##   degree fold train.error val.error
## 1      4    1    8.368867  4.816662
## 2      4    2    7.757257  7.813426
## 3      4    3    7.495269  9.195032
## 4      4    4    7.278740 10.307075
## 5      4    5    7.683365  8.278994
## 6      4    6    7.896173  7.230387
##   degree fold train.error val.error
## 1      5    1    8.366804  4.803476
## 2      5    2    7.744634  7.861714
## 3      5    3    7.493632  9.177620
## 4      5    4    7.266668 10.360326
## 5      5    5    7.665263  8.358911
## 6      5    6    7.895677  7.255280
##   degree fold train.error val.error
```

```
## 1      6    1    8.314426  4.851425
## 2      6    2    7.668704  8.055806
## 3      6    3    7.462140  9.111067
## 4      6    4    7.266611 10.379073
## 5      6    5    7.590881  8.568181
## 6      6    6    7.842271  7.318575
##   degree fold train.error val.error
## 1      7    1    8.297903  4.881005
## 2      7    2    7.622620  8.302501
## 3      7    3    7.455335  9.086806
## 4      7    4    7.250783 10.402811
## 5      7    5    7.590860  8.574298
## 6      7    6    7.835940  7.296061
##   degree fold train.error val.error
## 1      8    1    8.249949  4.822586
## 2      8    2    7.502641  8.716494
## 3      8    3    7.426507  8.950169
## 4      8    4    7.227724 10.261017
## 5      8    5    7.547281  8.486072
## 6      8    6    7.788168  7.251606
##   degree fold train.error val.error
## 1      9    1    8.240748  4.823584
## 2      9    2    7.492778  8.715575
## 3      9    3    7.425849  8.930280
## 4      9    4    7.191047 10.453235
## 5      9    5    7.539225  8.705025
## 6      9    6    7.759523  7.374172
##   degree fold train.error val.error
## 1     10    1    8.240486  4.825421
## 2     10    2    7.492546  8.719564
## 3     10    3    7.423976  8.953799
## 4     10    4    7.190618 10.457062
## 5     10    5    7.528974  8.822139
## 6     10    6    7.755880  7.418140
```

Couldnt figure out how to make the ldply function calculate different values of p in addition to chunkid. Eventually I nested it in a for loop to get the required results.

Plotting average Traing and Validation error

```
plot(c(0:10),mte,col="red",type="b",ylim = c(6,10),
     main="Plot of Training and Validation Error",xlab="Degree",
     ylab=" Mean of Training and Validation Error")
lines(c(0:10),mve,col="blue",type="b")
     legend("topright",legend = c("Mean Training Error","Mean Validation Error"),pch=c(1,1),col=c("red"
```



From this plot I see that the training error related to this data set decreases, and validation error increases, slightly with higher degree polynomials.

Based on the graph, I would choose the first degree of the polynomial. The reason being that it has the smallest average validation error.