PSTAT 131 Homework One

Lilian Lu and Ted Tinker

10/8/2017

1

(a)

Having installed and loaded the relevant packages,

```
algae %>% group_by(season) %>% summarise(count = n()) # Group algae by season, then summarise
```

```
## # A tibble: 4 x 2
##   season count
##    <chr> <int>
## 1 autumn    40
## 2 spring    53
## 3 summer    45
## 4 winter    62
```

All four seasons are well-represented in the data.

(b)

The number of missing values for each column:

```
colSums(is.na(algae)) # Report missing values in each variable
```

```
## season   size  speed   mxPH   mnO2     Cl    NO3    NH4   oPO4    PO4
##      0      0      0      1      2     10      2      2      2      2
##   Chla     a1     a2     a3     a4     a5     a6     a7
##     12      0      0      0      0      0      0      0
```

Most of the variables have almost all of their values. Chloride and Clorophyll have the most missing entries.

The mean and variance for each chemical:

```
colMeans(algae[6:11],na.rm = TRUE)  # Find mean in each column 6 through 11
```

```
##      Cl     NO3     NH4    oPO4     PO4    Chla
##  43.636   3.282 501.296  73.591 137.882  13.971
```

```
sapply(algae[6:11], var,na.rm=TRUE) # Find variance of each column 6 through 11
```

```
##        Cl       NO3       NH4      oPO4       PO4      Chla
## 2.193e+03 1.426e+01 3.852e+06 8.306e+03 1.664e+04 4.201e+02
```

The variances seems high, often an order of magnitude above the mean. Ammonium, in particular, has a variance 7600 times its mean.

(c)

$MAD = median(|X_i - median(X)|)$

```
(med<-sapply(algae[6:11], median,na.rm=TRUE)) # Median of columns 6 through 11
```

```
##      Cl     NO3     NH4    oPO4     PO4    Chla
##  32.730   2.675 103.166  40.150 103.285   5.475
```

```
temp<- rep(0,6) # Empty list
for (i in 1:6){
  temp[i]<-abs(algae[5+i][1]-med[i])} # Enter absolute value of residual from median
(MAD<-sapply(temp,median,na.rm=TRUE)) # Find medians of absolute residuals
```

```
## [1] 22.427  1.465 75.285 29.709 82.505  4.500
```

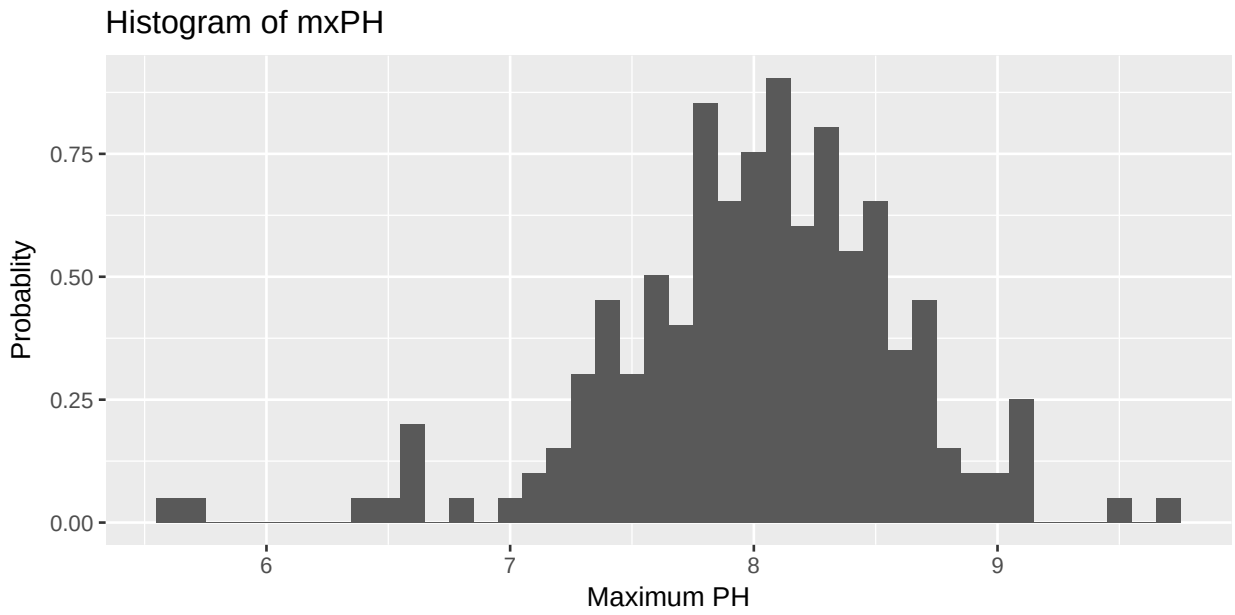
The MAD is the median of the absolute values of the residuals from the median. For example, the Median Absolute Residual for Chloride samples was 22.427, so (if we're understanding this correctly) 50% of chloride samples differ from the median Chloride sample by less than 22.427 units, and 50% of Chloride samples differ from the median Chloride sample by more than 22.427 units.

The medians are similar in scale to the means, but the MADs are within an order of magnitude of the medians whereas the variances were huge. The median and MAD are therefore better measurements of central tendency and data spread for this dataset.

2

(a)

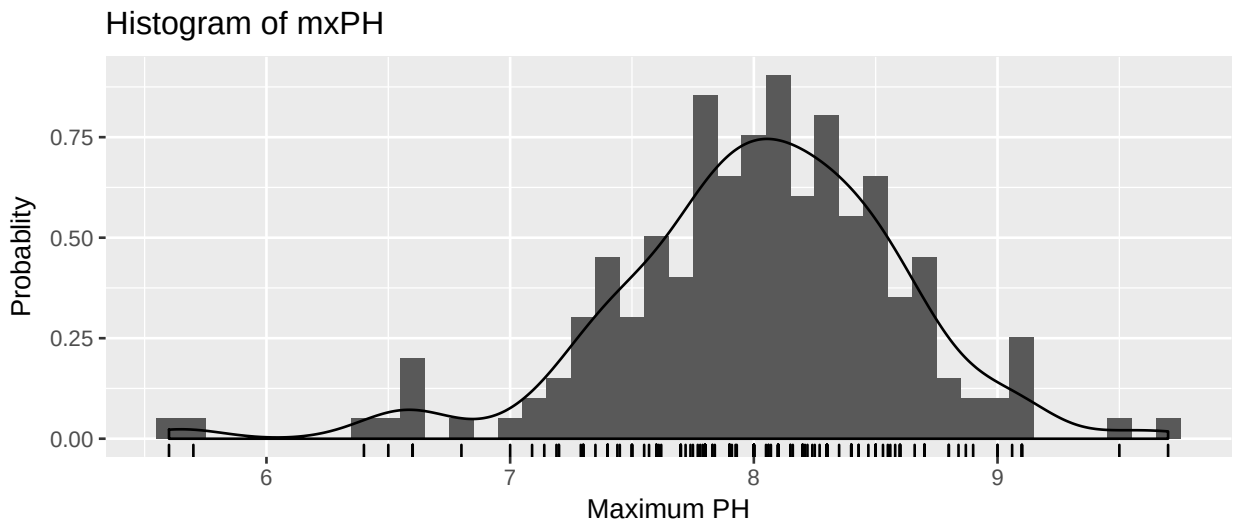
```
(myPlot <- ggplot(algae,aes(x=algae$mxPH))+ geom_histogram(aes(y=..density..),binwidth =
0.1)+ylab("Probablity") + xlab("Maximum PH") + ggtitle("Histogram of mxPH"))
```



The distribution resembles the typical bell-curve, but it appears to be skewed left.

(b)

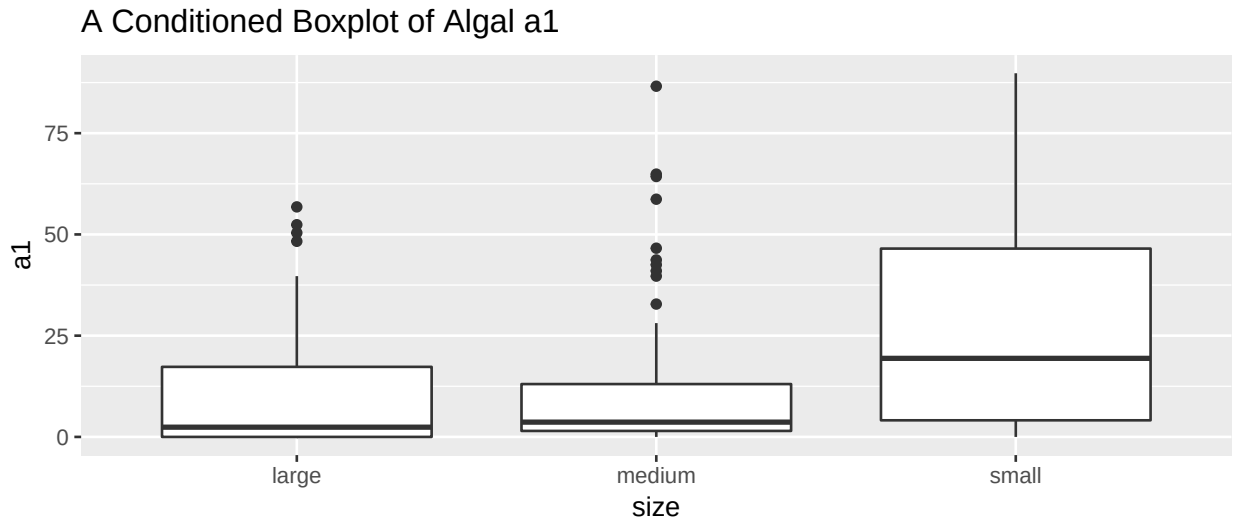
```
myPlot+geom_density() + geom_rug(aes(x=algae$mxPH)) # Add density and rug
```



The rug-plot gives a more precise visualization of the data clusters. The density plot fits the data well, confirming a general bell-shape.

(c)

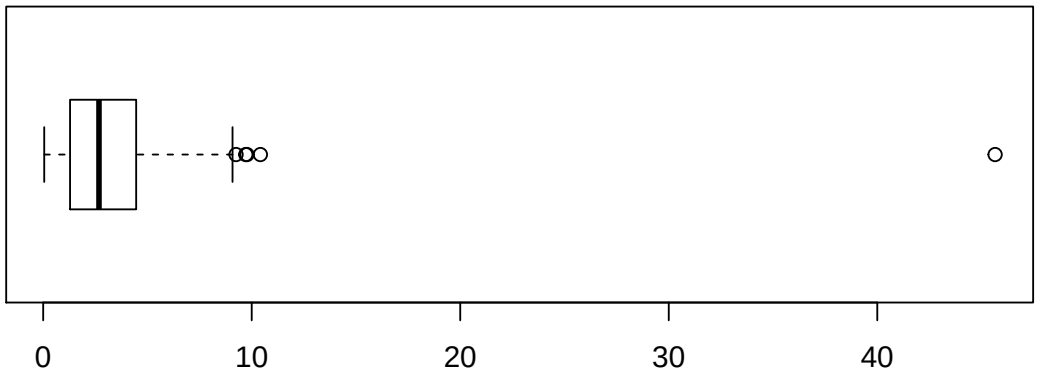
```
algsize<- algae %>% group_by(size) # Group algae by size
(plot2<-ggplot(algsize,aes(x=size,y=a1))+geom_boxplot()+ggtitle("A Conditioned Boxplot of Algal a1"))
```



The smallest samples appear to have the largest concentration of a1 algae, but the presence of outliers makes these plots difficult to compare.

(d)

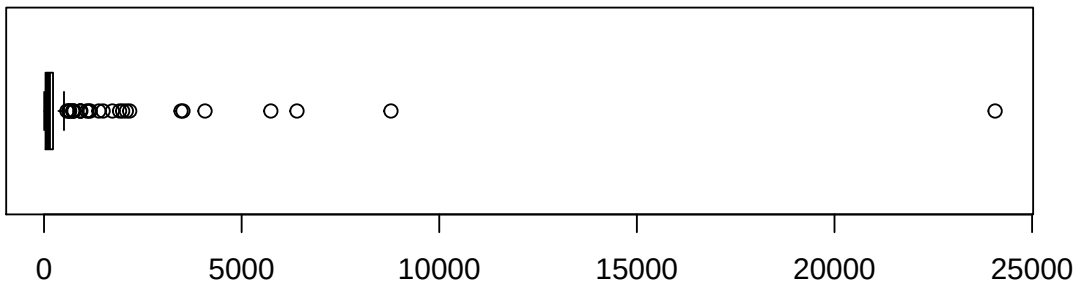
```
(plot3<-boxplot(algae$NO3,horizontal=TRUE)) # Make a boxplot of nitrate levels
```




```
## $stats
##       [,1]
## [1,] 0.050
## [2,] 1.288
## [3,] 2.675
## [4,] 4.459
## [5,] 9.080
##
## $n
## [1] 198
##
## $conf
##       [,1]
## [1,] 2.319
## [2,] 3.031
##
## $out
## [1] 10.416  9.248  9.773  9.715 45.650
##
## $group
## [1] 1 1 1 1 1
##
## $names
## [1] "1"
```

There is one point which should be considered an outlier by any reasonable metric. There are four data points which R labels outliers because they are outside 1.5× the Interquartial Range (and therefore displays as dots outside the box-and-whisker plot above), but which seem close enough to the mass of points to call them non-outlying.

```
(plot4<-boxplot(algae$NH4,horizontal=TRUE)) # Make boxplot of ammonium levels
```




```
## $stats
##        [,1]
## [1,]   5.00
## [2,]  38.11
## [3,] 103.17
## [4,] 227.60
## [5,] 505.00
##
## $n
## [1] 198
##
## $conf
##        [,1]
## [1,]  81.89
## [2,] 124.44
##
## $out
##  [1]   578.0  8777.6  1729.0  3515.0  6400.0  1911.0   647.6  1386.2
##  [9]  2082.9  2167.4   737.5   914.0  5738.3  4073.3   758.8   931.8
## [17]   723.7  3466.7   920.0  1990.2 24064.0  1131.7  1495.0   643.0
## [25]   627.3  1168.0  1081.7
##
## $group
##  [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
##
## $names
## [1] "1"
```

There are 27 data points outside the 1.5× IQR rule-of-thumb, but their arrangement makes me suspect $NH4$ concentration should be evaluated on a log-scale or after some other transformation. The fourth quartile is by far the largest, and the outlying points suggest a clear "stretching-out" of the data's tail.

(e)

Looking back the results from 1b and 1c we see that the variance for $NO3$ is only three times larger than its mean, the smallest of the discrepencies between mean and variance. But for $NH4$, the variance is 7600 times larger than the mean! The MAD appears more robust with respect to outliers, as both $NH4$ and $NO3$ can be understood in the context of medians.

3

(a)

```
sum(is.na(algae)) # Number of NA values
```

```
## [1] 33
```

```
sum(rowSums(is.na(algae))>=1) # Number of rows with at least one NA
```

```
## [1] 16
```

There are 33 missing values. Some of these entries share a row, so there are only 16 rows (or "observations") containing an n/a.

Revisiting this table from Problem One, Part (b):

```
colSums(is.na(algae)) # Report missing values in each variable
```

```
## season   size  speed   mxPH   mnO2     Cl    NO3    NH4   oPO4    PO4
##      0      0      0      1      2     10      2      2      2      2
##   Chla     a1     a2     a3     a4     a5     a6     a7
##     12      0      0      0      0      0      0      0
```

Most of the variables have almost all of their values. Chloride and Clorophyll have the most missing entries.

(b)

```
algae.del <- filter(algae,rowSums(is.na(algae))==0) # Make algae.del using rows of algae with no NA
nrow(algae.del) # Print number of rows
```

```
## [1] 184
```

There are 184 observations in *algae.del*. This is to be expected, as there are 200 observations in *algae* and 16 rows with n/a entries (found in part a).

(c)

```
algae.med <- algae %>% mutate_at(vars(mxPH:Chla),funs(ifelse(is.na(.),median(algae.del$.),.)))
  # Uses mutate_at instead of mutate_each for clearer syntax.
  # Mutates each column featuring NAs (found in part a) to either use the real value if available
  # or the median of the cleaned data if not
nrow(algae.med) # Print number of rows
```

```
## [1] 200
```

This has 200 observations, which is equal to the number of entries in *algae*, as expected.

```
algae.med[c(48,62,199),] # Print the rows specified
```

```
## # A tibble: 3 x 18
##   season  size  speed  mxPH  mnO2    Cl   NO3   NH4  oPO4   PO4  Chla
##    <chr> <chr>  <chr> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1 winter small    low   8.1 12.60  9.00  0.23  10.0  5.00   6.0 1.100
## 2 summer small medium   6.4  9.75 35.08  2.82 115.7 46.28  14.0 5.522
## 3 winter large medium   8.0  7.60 35.08  2.82 115.7 46.28 115.6 5.522
## # ... with 7 more variables: a1 <dbl>, a2 <dbl>, a3 <dbl>, a4 <dbl>,
## #   a5 <dbl>, a6 <dbl>, a7 <dbl>
```

As expected, these rows have no n/a entries. (In the *algae* tibble, these three rows account for 13 of the n/as.)

(d)

```
cor(select(algae.del,c(mxPH:a7))) # Finds correlation of numeric variables of algae.del
```

```
##          mxPH     mnO2       Cl      NO3      NH4      oPO4      PO4
## mxPH  1.00000 -0.10269  0.14710 -0.17213 -0.15430  0.090229  0.10133
## mnO2 -0.10269  1.00000 -0.26325  0.11791 -0.07827 -0.393753 -0.46396
## Cl    0.14710 -0.26325  1.00000  0.21096  0.06598  0.379256  0.44519
## NO3  -0.17213  0.11791  0.21096  1.00000  0.72468  0.133015  0.15703
## NH4  -0.15430 -0.07827  0.06598  0.72468  1.00000  0.219311  0.19940
## oPO4  0.09023 -0.39375  0.37926  0.13301  0.21931  1.000000  0.91196
## PO4   0.10133 -0.46396  0.44519  0.15703  0.19940  0.911965  1.00000
## Chla  0.43182 -0.13122  0.14296  0.14549  0.09120  0.106915  0.24849
## a1   -0.16263  0.24998 -0.35924 -0.24724 -0.12361 -0.394574 -0.45817
## a2    0.33502 -0.06848  0.07845  0.01997 -0.03790  0.123811  0.13267
## a3   -0.02716 -0.23523  0.07653 -0.09182 -0.11290  0.005705  0.03219
## a4   -0.18435 -0.37983  0.14147 -0.01449  0.27452  0.382481  0.40884
## a5   -0.10731  0.21001  0.14535  0.21214  0.01544  0.122027  0.15549
## a6   -0.17274  0.18863  0.16904  0.54404  0.40119  0.003340  0.05320
## a7   -0.17027 -0.10455 -0.04495  0.07505 -0.02539  0.026150  0.07978
##          Chla       a1       a2        a3       a4       a5       a6
## mxPH  0.43182 -0.16263  0.33502 -0.027160 -0.18435 -0.10731 -0.17274
## mnO2 -0.13122  0.24998 -0.06848 -0.235228 -0.37983  0.21001  0.18863
## Cl    0.14296 -0.35924  0.07845  0.076530  0.14147  0.14535  0.16904
## NO3   0.14549 -0.24724  0.01997 -0.091822 -0.01449  0.21214  0.54404
## NH4   0.09120 -0.12361 -0.03790 -0.112905  0.27452  0.01544  0.40119
## oPO4  0.10691 -0.39457  0.12381  0.005705  0.38248  0.12203  0.00334
## PO4   0.24849 -0.45817  0.13267  0.032194  0.40884  0.15549  0.05320
```

```
## Chla  1.00000 -0.26601  0.36672 -0.063301 -0.08601 -0.07343  0.01033
## a1   -0.26601  1.00000 -0.26267 -0.108178 -0.09338 -0.26973 -0.26156
## a2    0.36672 -0.26267  1.00000  0.009760 -0.17629 -0.18676 -0.13352
## a3   -0.06330 -0.10818  0.00976  1.000000  0.03337 -0.14161 -0.19690
## a4   -0.08601 -0.09338 -0.17629  0.033369  1.00000 -0.10132 -0.08488
## a5   -0.07343 -0.26973 -0.18676 -0.141611 -0.10132  1.00000  0.38861
## a6    0.01033 -0.26156 -0.13352 -0.196900 -0.08488  0.38861  1.00000
## a7    0.01761 -0.19306  0.03621  0.039060  0.07115 -0.05149 -0.03033
##            a7
## mxPH -0.17027
## mnO2 -0.10455
## Cl   -0.04495
## NO3   0.07505
## NH4  -0.02539
## oPO4  0.02615
## PO4   0.07978
## Chla  0.01761
## a1   -0.19306
## a2    0.03621
## a3    0.03906
## a4    0.07115
## a5   -0.05149
## a6   -0.03033
## a7    1.00000
```

$PO4$ and $oPO4$ are highly correlated (at .991965). Therefore it makes sense to try to fill missing entries of $PO4$ using $oPO4$.

```
algae[28,c("PO4","oPO4")] # Shows missing value and value to reconstruct from
```

```
## # A tibble: 1 x 2
##     PO4  oPO4
##   <dbl> <dbl>
## 1    NA     4
```

```
myPrediction <- lm(PO4 ~ oPO4,algae.del) # Use algae.del to make a linear model
summary(myPrediction)
```

```
##
## Call:
## lm(formula = PO4 ~ oPO4, data = algae.del)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -109.2  -37.9  -13.8   25.8  219.0
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  47.0802     5.1323    9.17   <2e-16 ***
## oPO4          1.2712     0.0424   29.99   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 53.1 on 182 degrees of freedom
## Multiple R-squared:  0.832,  Adjusted R-squared:  0.831
## F-statistic:  899 on 1 and 182 DF,  p-value: <2e-16
```

Our linear model has an intercept of 47.0802 and a slope of 1.2712. Knowing the $oPO4$ in this row is 4, we may fill $PO4$ using the model's prediction:

```
algae[28,"PO4"] <- predict(myPrediction,algae[28,"oPO4"]) # Replace the missing value with the model.
algae[28,"PO4"] # Shows value after modeling
```

```
## # A tibble: 1 x 1
##     PO4
##   <dbl>
## 1 52.17
```

4

(a)

```
set.seed(100) # For reproducibility
(fold<-cut(1:nrow(algae.med),breaks=5,labels=FALSE) %>% sample()) # Cut data into five
```

```
##   [1] 2 2 3 1 3 3 4 2 3 1 3 5 2 2 4 4 1 2 5 4 3 4 3 4 2 1 4 4 5 2 3 4 2 4 3
##  [36] 4 5 3 5 1 5 4 4 4 5 2 4 4 1 2 2 5 1 2 3 1 1 5 3 4 2 3 5 3 2 2 5 2 4 3
##  [71] 2 2 5 5 5 2 3 5 3 5 2 5 3 3 1 2 4 1 1 3 3 1 1 5 5 5 5 1 1 2 4 3 5 1 2
## [106] 5 5 4 1 5 2 5 1 5 3 4 4 5 1 2 4 2 4 2 1 1 5 2 1 3 3 4 2 3 3 3 1 5 3 4
## [141] 5 1 1 5 1 3 2 2 4 1 1 2 3 2 1 3 5 3 1 4 5 3 1 5 5 3 2 1 1 4 1 5 2 3 4
## [176] 2 2 1 2 4 4 4 4 4 5 1 3 2 3 1 1 4 4 5 5 4 4 3 1 3
```

(b)

This section utylizes the code given in the problem.

```
do.chunk <- function(chunkid, chunkdef, dat) { # function argument
  train = (chunkdef!= chunkid)
  Xtr = dat[train,1:11] # get training set
  Ytr = dat[train,12] # get true response values in trainig set
  Xvl = dat[!train,1:11] # get validation set
  Yvl = dat[!train,12] # get true response values in validation set
  lm.a1 <- lm(a1~., data = dat[train,1:12])
  predYtr = predict(lm.a1) # predict training values
  predYvl = predict(lm.a1,Xvl) # predict validation values
  data.frame(fold = chunkid,train.error = mean((predYtr - Ytr)^2), # compute and store training error
  val.error = mean((predYvl - Yvl)^2)) # compute and store test error
} # The code above is copied from the homework

lapply(1:5, function(z) {do.chunk(z,fold,algae.med)})
```

```
## [[1]]
##   fold train.error val.error
## 1    1       271.6     391.1
##
## [[2]]
##   fold train.error val.error
## 1    2       291.2     279.1
##
```

```
## [[3]]
##   fold train.error val.error
## 1    3       290.5     292.6
##
## [[4]]
##   fold train.error val.error
## 1    4         283     314.3
##
## [[5]]
##   fold train.error val.error
## 1    5       273.6     354.1
```

```
  # For each chunk's model, find test/training error
```

The average test error is 326. The training errors are generally consistent, ranging from 271.6 to 291.2. This is because all models fit the training data, as expected.

5

Having loaded the new dataset, we use it to evaluate the 'true' test error of the model approximated in part four.

```
Xtr = algae.med[1:11]    # The trainest set
Ytr = algae.med[12]      # Get real values for training data
Xvl = algae.test[1:11]       # Predictors for test data
Yvl = algae.test[12]         # Real values for test data
ourModel <- lm(a1~., data = algae.med[1:12])       # Make model
predYtr = predict(ourModel,newdata = algae.med[1:11])    # Predict
predYvl = predict(ourModel,newdata=algae.test[1:11]) # predict validation values
data.frame(train.error = mean((predYtr - Ytr)^2), # compute and store training error
val.error = mean((predYvl - Yvl)^2))
```

```
##   train.error val.error
## 1       286.1       250
```

This test error is actually lower than when we predicted it should be given cross-validation! It decreased more than 75 points.

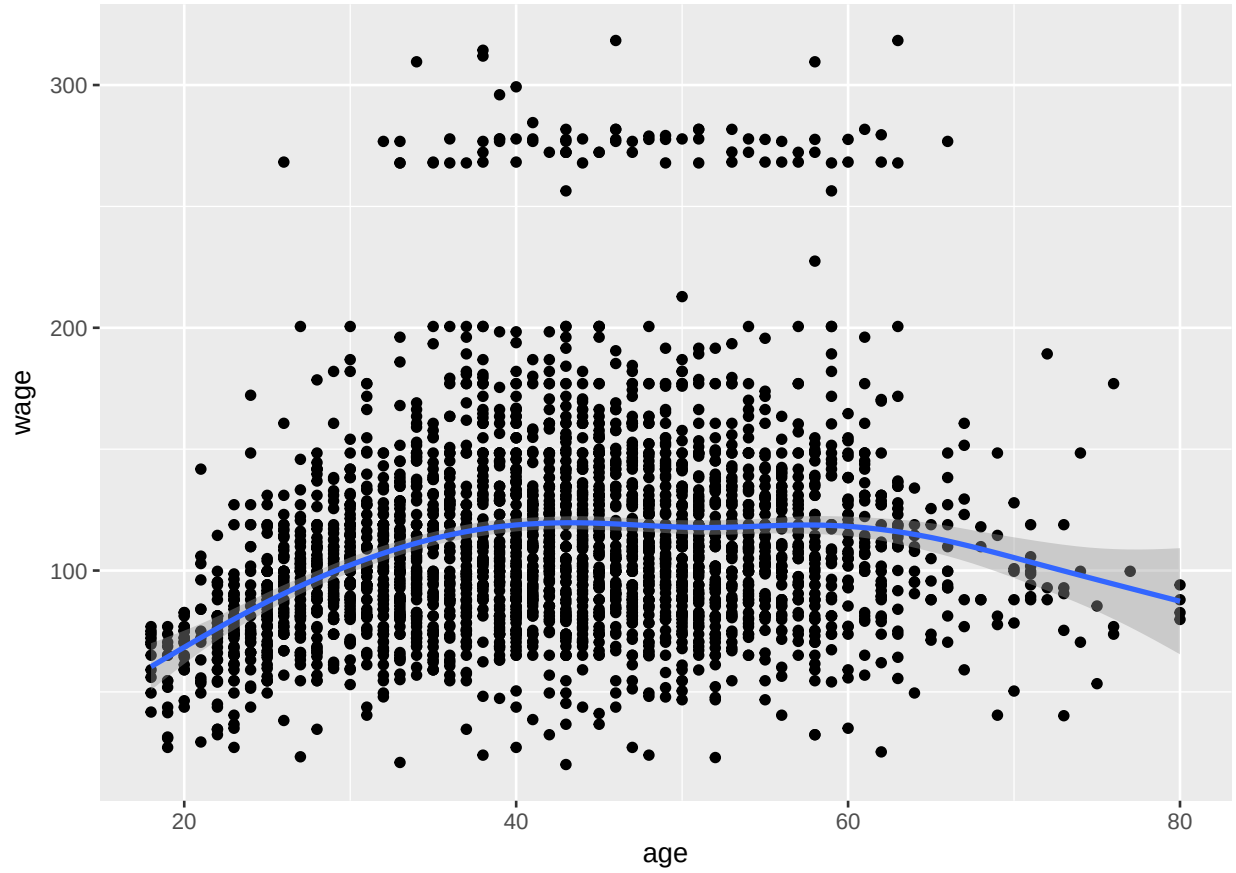
6

```
# install.packages("ISLR")
library(ISLR)
```

(a)

```
(wagePlot <- ggplot(Wage, aes(x=age,y=wage))+geom_point() + geom_smooth())
```

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



The first noticable trait in this data is significant banding. There are two major clusters of wage-level, one from almost no wages to 200 units, and another around 275 wage units. This matches my personal expectation, that age would be relevant to wage but that wages would also be distributed along class lines and race factors. Perhaps this banding could be due to sampling error; maybe wages above a certain level were rounded down to 275. It's difficult to tell with just the plot as it is.

Moreover, examining the smooth curve, wage generally increases through a person's twenties but levels off in their forties. After age sixty wage seems to decline, but the variance of these points is far higher. This may be because of a smaller sample size for the elderly.

(b)

(i)

Fit the linear regression

```
linearpred<-lm(wage~poly(age,10,raw = FALSE),Wage)
summary(linearpred)
```

```
##
## Call:
## lm(formula = wage ~ poly(age, 10, raw = FALSE), data = Wage)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -100.38  -24.45   -4.97   15.49  199.61
##
## Coefficients:
##                               Estimate Std. Error t value Pr(>|t|)
## (Intercept)                    111.704      0.728  153.37   <2e-16 ***
## poly(age, 10, raw = FALSE)1    447.068     39.892   11.21   <2e-16 ***
## poly(age, 10, raw = FALSE)2   -478.316     39.892  -11.99   <2e-16 ***
## poly(age, 10, raw = FALSE)3    125.522     39.892    3.15   0.0017 **
## poly(age, 10, raw = FALSE)4    -77.911     39.892   -1.95   0.0509 .
## poly(age, 10, raw = FALSE)5    -35.813     39.892   -0.90   0.3694
## poly(age, 10, raw = FALSE)6     62.708     39.892    1.57   0.1161
## poly(age, 10, raw = FALSE)7     50.550     39.892    1.27   0.2052
## poly(age, 10, raw = FALSE)8    -11.255     39.892   -0.28   0.7779
## poly(age, 10, raw = FALSE)9    -83.692     39.892   -2.10   0.0360 *
## poly(age, 10, raw = FALSE)10     1.624     39.892    0.04   0.9675
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 39.9 on 2989 degrees of freedom
## Multiple R-squared:  0.0891, Adjusted R-squared:  0.0861
## F-statistic: 29.2 on 10 and 2989 DF,  p-value: <2e-16
```

(ii)

The following is adapted from the hint posted on Piazza

```
do.chunk.poly <- function(chunkid, chunkdef, dat, p){  # function argument

  train = (chunkdef != chunkid)

  Xtr = dat[train,]$age  # get training set
  Ytr = dat[train,]$wage  # get true response values in trainig set

  Xvl = dat[!train,]$age  # get validation set
  Yvl = dat[!train,]$wage  # get true response values in validation set

  ## at the end of this function you should return p, which fold you are testing on
  ## and the training error and test error
  ## This is an empty data frame which will include your results at the end
```

```
  res = data.frame(degree=integer(), fold=integer(),
                   train.error=double(), val.error=double())


   if (p==0) {
    lm.wage <- lm(wage~1, data = dat[train,])
}
  else {

    lm.wage<-lm(wage~poly(age, degree = p,raw=FALSE), data= dat[train,])

  }

    predYvl = predict(lm.wage, newdata=dat[!train,])
    predYtr = predict(lm.wage) # predict training values
    fold = chunkid


    data.frame(degree=p, fold = chunkid, train.error = mean((predYtr - Ytr)^2),
               val.error = mean((predYvl - Yvl)^2))

}

# compute and store test error
## return training and test error for current chunk / polynomial p
# get training/validation error for each fold/each model
test.list = rep(0,11)
training.list = rep(0,11)
for (p in 0:10) {
    result<-lapply(1:5, function(z) {do.chunk.poly(z,fold2,Wage,p)})
  ## call do.chunk.poly on each fold for order p (follow problem 4 as an example)
    test.list[p+1] <- (result[[1]]$val.error + result[[2]]$val.error + result[[3]]$val.error +
                         result[[4]]$val.error + result[[5]]$val.error)/5
    training.list[p+1] <- (result[[1]]$train.error + result[[2]]$train.error +
                             result[[3]]$train.error + result[[4]]$train.error +
                             result[[5]]$train.error)/5
}
## plot and test and train errors from result of cross validation

print(test.list)
```

```
##  [1] 1741 1676 1600 1596 1594 1595 1594 1595 1596 1595 1595
```

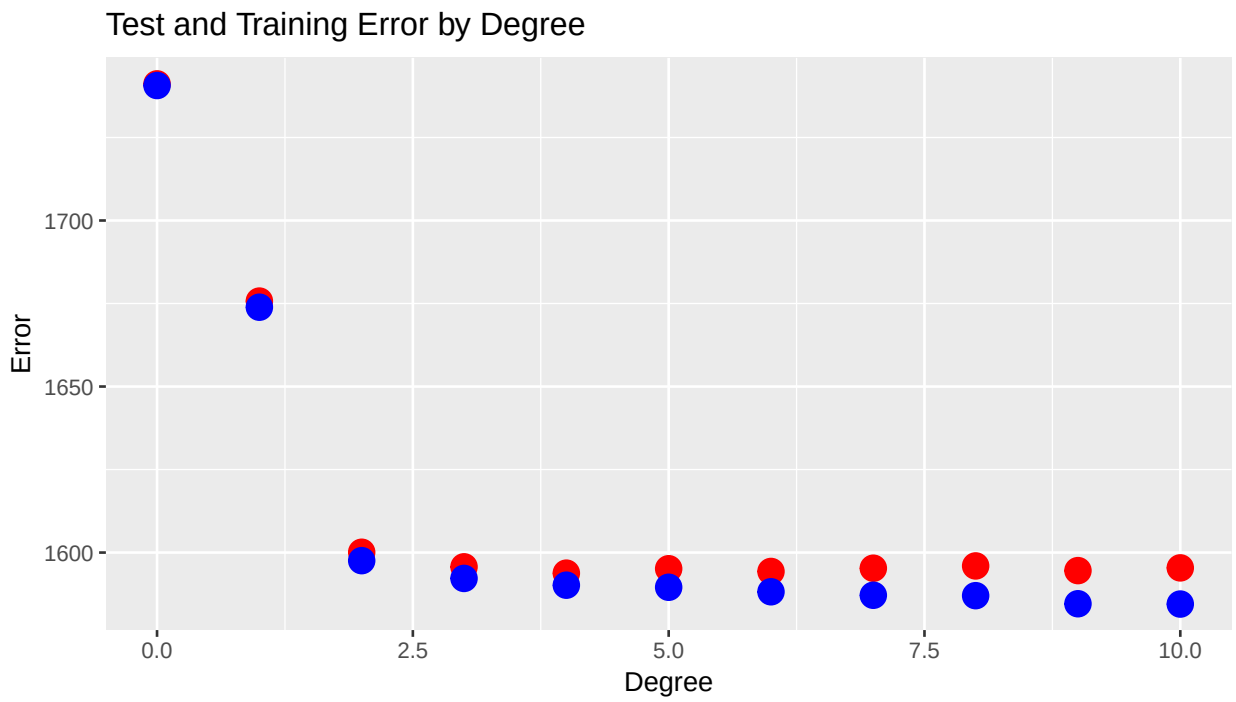
```
print(training.list)
```

```
##  [1] 1741 1674 1598 1592 1590 1590 1588 1587 1587 1585 1584
```

The first row shows the average test error in 5-fold cross-validation. We notice it decreases as the degree p increases until $p = 4$, then barely budges. The second row shows the average training error in 5-fold cross validation. We notice it strictly decreases as p increases (which makes sense, because over-fitting would lower the training error while increasing the test error). Interestingly, at the $p = 1$ model, the average test data equals the average training data.

(c)

```
zero2ten = 0:10 # Quick enumerating for indexing
myerror<-do.call(rbind.data.frame,Map('c',zero2ten,test.list,training.list))
      # Format data-frame for plotting
(myPlot <- ggplot(aes(x=zero2ten),data =myerror) + geom_point(aes(y=test.list,size=10),col="Red") +
      geom_point(aes(y=training.list,size=10),col="blue") + xlab("Degree") + ylab("Error") + ggtitle("T
```



The training error and test error appear similar, but test error levels off asymptotically when p is high even as training error continues to decrease. We recommend choosing the model where $p = 4$, the point at which increasing p no longer significantly reduces the test error.