PSTAT131_HW1

Justin Lee

10/14/2020

(1)

Reading the data and attaching packages

```
library(tidyverse)
library(dplyr)
library(ggplot2)
library(resample)
library(miscTools)
library(FIACH)

algae <- read_table2("algaeBloom.txt",col_names=c('season','size','speed','mxPH','mnO2','Cl','NO3','NH4
'oPO4','PO4','Chla','a1','a2','a3','a4','a5','a6','a7'),
na="XXXXXXX")

glimpse(algae)
```

```
## Rows: 200
## Columns: 18
## $ season <chr> "winter", "spring", "autumn", "spring", "autumn", "winter", ...
## $ size   <chr> "small", "small", "small", "small", "small", "small", "small...
## $ speed  <chr> "medium", "medium", "medium", "medium", "medium", "high", "h...
## $ mxPH   <dbl> 8.00, 8.35, 8.10, 8.07, 8.06, 8.25, 8.15, 8.05, 8.70, 7.93, ...
## $ mnO2   <dbl> 9.8, 8.0, 11.4, 4.8, 9.0, 13.1, 10.3, 10.6, 3.4, 9.9, 10.2, ...
## $ Cl     <dbl> 60.800, 57.750, 40.020, 77.364, 55.350, 65.750, 73.250, 59.0...
## $ NO3    <dbl> 6.238, 1.288, 5.330, 2.302, 10.416, 9.248, 1.535, 4.990, 0.8...
## $ NH4    <dbl> 578.000, 370.000, 346.667, 98.182, 233.700, 430.000, 110.000...
## $ oPO4   <dbl> 105.000, 428.750, 125.667, 61.182, 58.222, 18.250, 61.250, 4...
## $ PO4    <dbl> 170.000, 558.750, 187.057, 138.700, 97.580, 56.667, 111.750,...
## $ Chla   <dbl> 50.000, 1.300, 15.600, 1.400, 10.500, 28.400, 3.200, 6.900, ...
## $ a1     <dbl> 0.0, 1.4, 3.3, 3.1, 9.2, 15.1, 2.4, 18.2, 25.4, 17.0, 16.6, ...
## $ a2     <dbl> 0.0, 7.6, 53.6, 41.0, 2.9, 14.6, 1.2, 1.6, 5.4, 0.0, 0.0, 0....
## $ a3     <dbl> 0.0, 4.8, 1.9, 18.9, 7.5, 1.4, 3.2, 0.0, 2.5, 0.0, 0.0, 0.0,...
## $ a4     <dbl> 0.0, 1.9, 0.0, 0.0, 0.0, 0.0, 3.9, 0.0, 0.0, 2.9, 0.0, 0.0, ...
## $ a5     <dbl> 34.2, 6.7, 0.0, 1.4, 7.5, 22.5, 5.8, 5.5, 0.0, 0.0, 1.2, 0.0...
## $ a6     <dbl> 8.3, 0.0, 0.0, 0.0, 4.1, 12.6, 6.8, 8.7, 0.0, 0.0, 0.0, 0.0,...
## $ a7     <dbl> 0.0, 2.1, 9.7, 1.4, 1.0, 2.9, 0.0, 0.0, 0.0, 1.7, 6.0, 1.5, ...
```

(a)

```r
algae %>% group_by(season) %>% summarise(n = n())
```

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

From the data we see above, we can see the total count of the observations are:

- Autumn = 40
- Spring = 53
- Summer = 45
- Winter = 62

(b)

```r
missing = is.na(algae)
length(missing[missing == TRUE])
```

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

```r
chemicals = algae[, 4:11]
colMeans(chemicals,na.rm = TRUE)
```

```
##       mxPH       mnO2         Cl        NO3        NH4       oPO4        PO4
##   8.011734   9.117778  43.636279   3.282389 501.295828  73.590596 137.882101
##       Chla
##  13.971197
```

```r
#Using package 'Resample'
colVars(chemicals,na.rm= TRUE)
```

```
##         mxPH         mnO2           Cl          NO3          NH4         oPO4
## 3.579693e-01 5.718089e+00 2.193172e+03 1.426176e+01 3.851585e+06 8.305850e+03
##          PO4         Chla
## 1.663938e+04 4.200827e+02
```

We can confirm that there are missing values within the dataset using the function "is.na()". The total count of the missing values were 33. I used the length function to see how many counts of "TRUE" there were in the "missing" subset. The mean and variance is shown above as well using the *colMeans* function as well as *colvars* from a pacakge 'Resample' that I have learned from another class. For both the cases of mean and variance, the missing values were ignored. The magnitude is greater than the mean for most of the chemicals which can indicate that the data points are very spread out from the average.

(c)

```
colMedians(chemicals, na.rm = TRUE)
```

```
##     mxPH     mnO2       Cl      NO3      NH4     oPO4      PO4     Chla
##   8.0600   9.8000  32.7300   2.6750 103.1665  40.1500 103.2855   5.4750
```

```
#chemicals subset with the missing values replaced with mean for each chemical
chemicals_2 = chemicals
chemicals_2$mnO2[is.na(chemicals_2$mnO2)] = mean(chemicals$mnO2,na.rm = TRUE)
chemicals_2$Cl[is.na(chemicals_2$Cl)] = mean(chemicals$Cl,na.rm = TRUE)
chemicals_2$NO3[is.na(chemicals_2$NO3)] = mean(chemicals$NO3,na.rm = TRUE)
chemicals_2$NH4[is.na(chemicals_2$NH4)] = mean(chemicals$NH4,na.rm = TRUE)
chemicals_2$oPO4[is.na(chemicals_2$oPO4)] = mean(chemicals$oPO4,na.rm = TRUE)
chemicals_2$PO4[is.na(chemicals_2$PO4)] = mean(chemicals$PO4,na.rm = TRUE)
chemicals_2$Chla[is.na(chemicals_2$Chla)] = mean(chemicals$Chla,na.rm = TRUE)

missing2 = is.na(chemicals_2)
#Confirming there are no missing values, this should output zero if I did everythin correctly
length(missing2[missing2 == TRUE])
```

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

```
#Computing M.A.D. for each chemical using package 'FIACH'
colMad(chemicals_2)
```

```
## [1]   0.504084   1.979271  35.337771   2.153477 112.059356  45.466153 121.444955
## [8]   7.685057
```

Since the output for the Median Absolute Deviation is does not show which output for each chemical, I will state them neatly here:

- mxPH = 0.504084
- mnO2 = 1.979271
- Cl = 35.337771
- NO3 = 2.153477
- NH4 = 112.059356
- OPO4 = 45.466153
- PO4 = 121.444955
- Chla = 7.685057

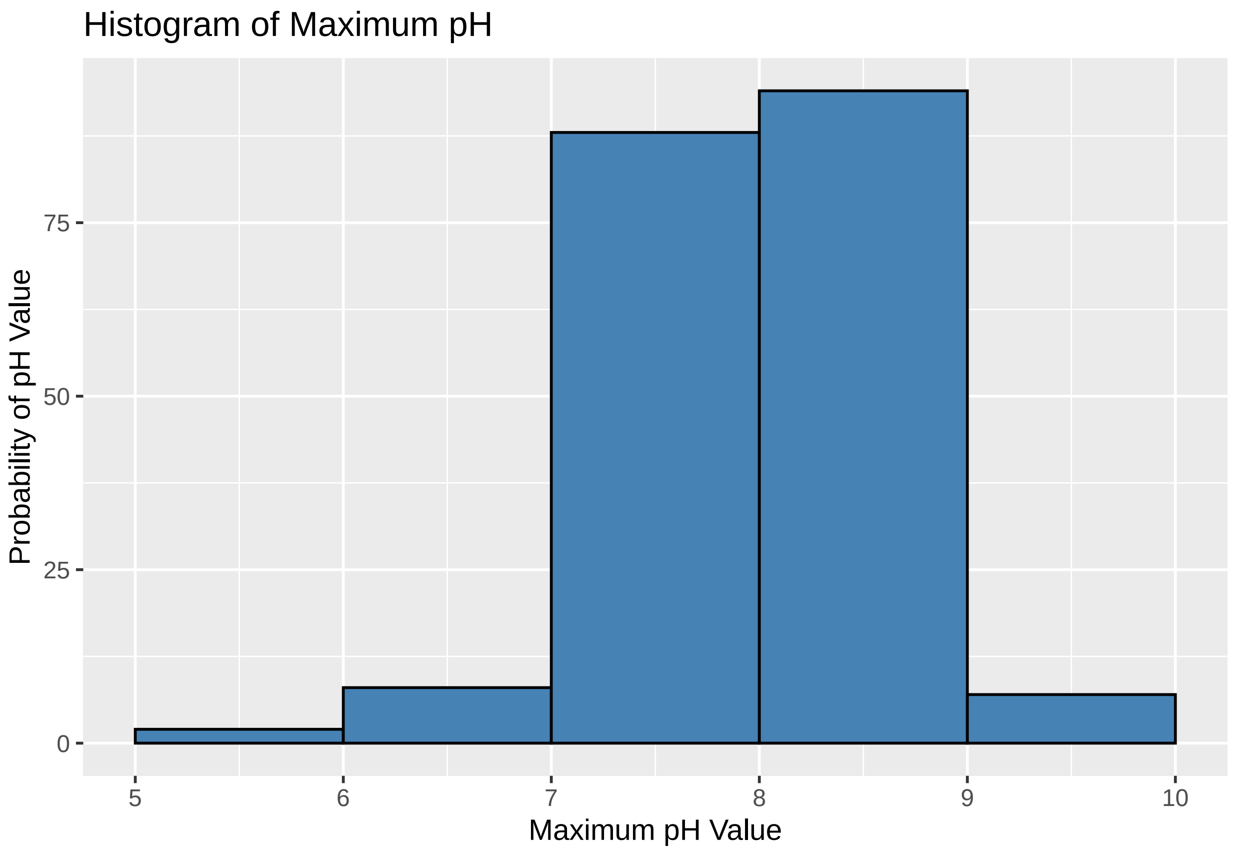
The median for each chemicals were found using the *colMedians* function from the package "miscTools". This provided a lot more of a simple way for me to output the name of the chemical and the median. Finding the median absolute deviation has many ways including using the mad() function but for simplicity I used *colMad* from the package "FIACH" that I learned in the past. Since missing values present needs to be replaced to find M.A.D., I used *is.na()* function to find the missing values and replaced them with the mean value for that chemical. This adjusted value should remain closer to the original, but just more accurate. We can see that the mean and variances have magnitudes with larger differences. We can see that the variance for mnO_2 is 5.718089 and the mean is 9.117778. This is almost two times from the variance and there are significant differences between chemicals in variance and mean. However, the medians and the median absolute deviations of the chemicals are very close to each other compared to the other. This pattern is true for all of the chemicals beside mnO_2.

(2)

(a)

```
algae %>% ggplot(aes(x=mxPH, stat = "density")) +
  geom_histogram(breaks = seq(5, 10, by = 1),col = "black", fill = "steelblue") +
  labs(title = "Histogram of Maximum pH", x = "Maximum pH Value", y = "Probability of pH Value")
```

```
## Warning: Removed 1 rows containing non-finite values (stat_bin).
```



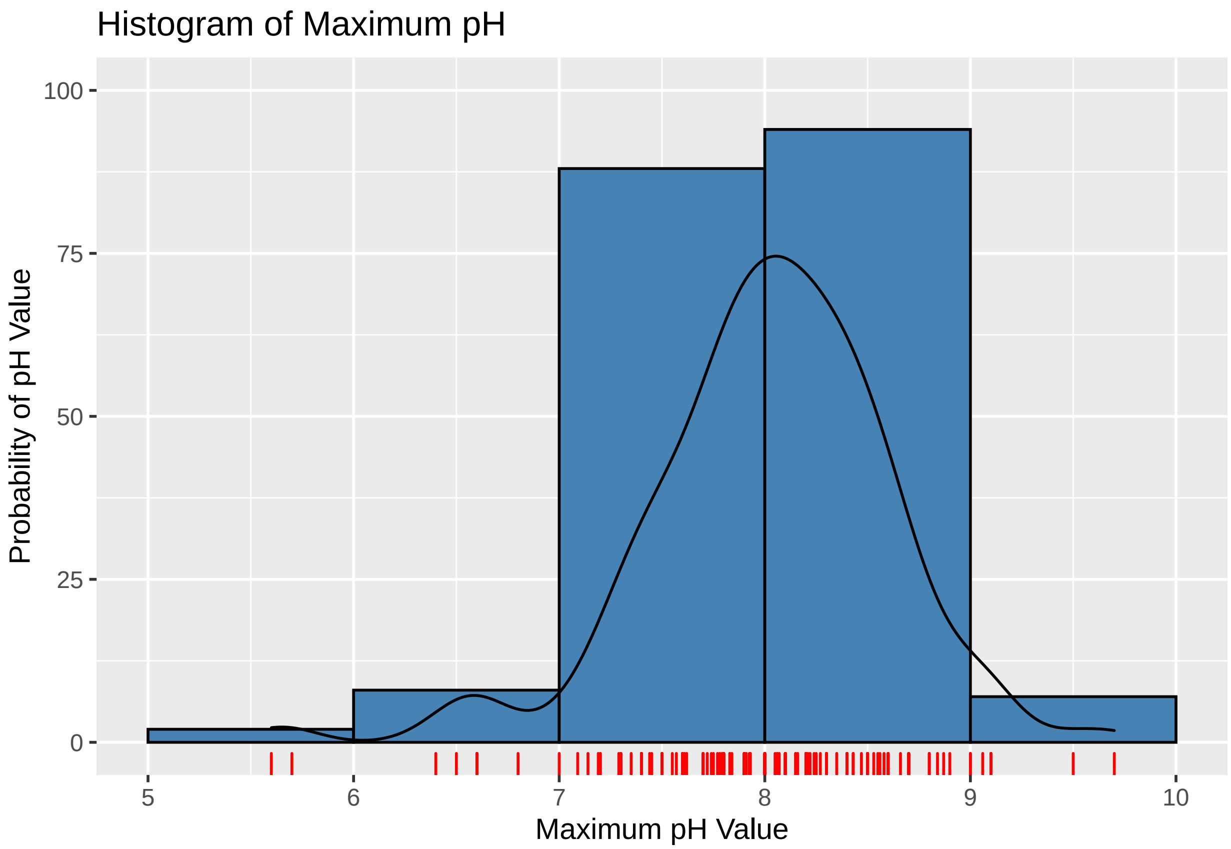
I used the ggplot() function to produce a histogram with probability on the vertical axis and the maximum pH on the horizontal axis. Using the statement *stat = “density”* gives us a histogram that contains a measure of density instead of frequency. The distribution seems to be skewed slightly to the left from the plot and the computed median of mxPH is larger than it’s mean.

(b)

```
algae %>% ggplot(aes(x=mxPH, stat = "density")) +
  geom_histogram(breaks = seq(5, 10, by = 1),col = "black", fill = "steelblue") +
  geom_density(aes(y = ..density..*(100))) +
  geom_rug(col = "red") +
  labs(title = "Histogram of Maximum pH", x = "Maximum pH Value", y = "Probability of pH Value") +
  ylim(c(0,100))
```

```
## Warning: Removed 1 rows containing non-finite values (stat_bin).
```

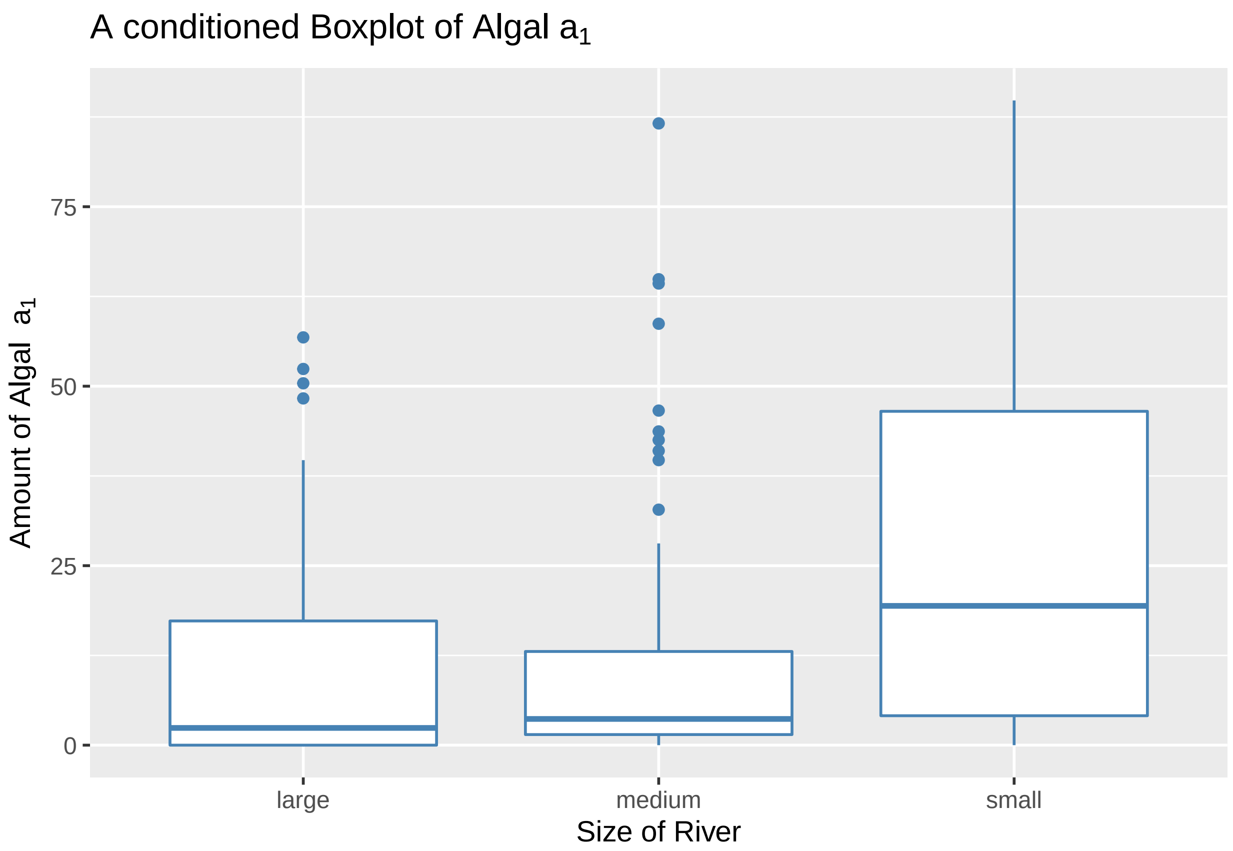
```
## Warning: Removed 1 rows containing non-finite values (stat_density).
```



Using geom_density() and geom_rug, we can add the density curve along with the rug plot. The geom_density() did give me an error in the beginning due to the fact that the probability was out of a hundred, but the density curve was out of one. However, I multipled the y-values of the density curve by a hundred to fix the problem.

(c)

```
algae %>% ggplot(aes(x = size, y = a1)) +
  geom_boxplot(col = "steelblue") +
  labs(title = expression(paste("A conditioned Boxplot of Algal ", "a"[1])), x = "Size of River", y = e
```



We use ggplot() along with geom_boxplot to creat a boxplot for a_1. We use the aes() statement to group them by size which are: small,medium,large. We also specify in the aes() statement to indicate that the y-axis will be the data from a_1 and the x-axis will the size.

(d)

```
x <- algae$NO3
y <- algae$NH4
x[which(x %in% boxplot.stats(x)$out)]
```

```
## [1] 10.416  9.248  9.773  9.715 45.650
```

```
y[which(y %in% boxplot.stats(y)$out)]
```

```
##  [1]   578.000  8777.600  1729.000  3515.000  6400.000  1911.000   647.570
##  [8]  1386.250  2082.850  2167.370   737.500   914.000  5738.330  4073.330
## [15]   758.750   931.833   723.667  3466.660   920.000  1990.160 24064.000
## [22]  1131.660  1495.000   643.000   627.273  1168.000  1081.660
```

To find the outliers we can use the range given by this formula: $[(Q1 - 1.5IQR), (Q3 + 1.5IQR)]$. IQR is the interquartile range and Q1 and Q3 are the quartiles. I assigned the variable NO_3 to to x and NH_4 to y. Using the operator, which(), and the function, boxplot.stats(). Through the function I wrote, we are able to find which data points are outliers. From the results we can see that there are 5 outliers for NO_3 and 27 outliers for NH_4.

(e)

From question 1 we know:
NO_3:
mean = 3.282389, variance = 14.26176, median = 2.6750, MAD = 22.153477
NH_4:
mean = 501.295828, variance = 3851585, median = 103.1665, MAD = 112.059356

We can see that both the variance of NO3 and NH4 is significantly larger than the mean. When we look at the values for the median and MAD we can see that they are very similar. In conclusion, the median and MAD values that are not influenced much form the outlier which means that they are more robust measurements than the mean and variance.

(3)

(a)

```r
summary(algae)
```

```
##     season              size              speed                mxPH      
##  Length:200         Length:200         Length:200         Min.   :5.600  
##  Class :character   Class :character   Class :character   1st Qu.:7.700  
##  Mode  :character   Mode  :character   Mode  :character   Median :8.060  
##                                                           Mean   :8.012  
##                                                           3rd Qu.:8.400  
##                                                           Max.   :9.700  
##                                                           NA's   :1      
##       mnO2              Cl               NO3              NH4          
##  Min.   : 1.500   Min.   :  0.222   Min.   : 0.050   Min.   :    5.00  
##  1st Qu.: 7.725   1st Qu.: 10.981   1st Qu.: 1.296   1st Qu.:   38.33  
##  Median : 9.800   Median : 32.730   Median : 2.675   Median :  103.17  
##  Mean   : 9.118   Mean   : 43.636   Mean   : 3.282   Mean   :  501.30  
##  3rd Qu.:10.800   3rd Qu.: 57.824   3rd Qu.: 4.446   3rd Qu.:  226.95  
##  Max.   :13.400   Max.   :391.500   Max.   :45.650   Max.   :24064.00  
##  NA's   :2        NA's   :10        NA's   :2        NA's   :2         
##       oPO4             PO4              Chla               a1       
##  Min.   :  1.00   Min.   :  1.00   Min.   :  0.200   Min.   : 0.00  
##  1st Qu.: 15.70   1st Qu.: 41.38   1st Qu.:  2.000   1st Qu.: 1.50  
##  Median : 40.15   Median :103.29   Median :  5.475   Median : 6.95  
##  Mean   : 73.59   Mean   :137.88   Mean   : 13.971   Mean   :16.92  
##  3rd Qu.: 99.33   3rd Qu.:213.75   3rd Qu.: 18.308   3rd Qu.:24.80  
##  Max.   :564.60   Max.   :771.60   Max.   :110.456   Max.   :89.80  
##  NA's   :2        NA's   :2        NA's   :12                       
##        a2               a3               a4               a5        
##  Min.   : 0.000   Min.   : 0.000   Min.   : 0.000   Min.   : 0.000  
##  1st Qu.: 0.000   1st Qu.: 0.000   1st Qu.: 0.000   1st Qu.: 0.000  
##  Median : 3.000   Median : 1.550   Median : 0.000   Median : 1.900  
##  Mean   : 7.458   Mean   : 4.309   Mean   : 1.992   Mean   : 5.064  
##  3rd Qu.:11.375   3rd Qu.: 4.925   3rd Qu.: 2.400   3rd Qu.: 7.500  
##  Max.   :72.600   Max.   :42.800   Max.   :44.600   Max.   :44.400  
##                                                                     
##        a6               a7        
##  Min.   : 0.000   Min.   : 0.000  
##  1st Qu.: 0.000   1st Qu.: 0.000  
##  Median : 0.000   Median : 1.000  
##  Mean   : 5.964   Mean   : 2.495  
##  3rd Qu.: 6.925   3rd Qu.: 2.400  
##  Max.   :77.600   Max.   :31.600  
## 
```

```r
sum(is.na(algae))
```

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

Using the *summary()* function, we can see which variables have missing values and how many there are. We can see that every predictors have missing values starting from mxPH to Chla. From this we can see that *mxPH* has 1 missing value, *mnO2* has 2 missing values, *Cl* has 10 missing values, *NO3* has 2 missing values, *NH4* has 2 missing values, *OPO4* has 2 missing values, *PO4* has 2 missing values, and *Chla* has 12 missing values. This brings us to a total of 33 missing values.

(b)

```
algae.del = filter(algae,  !is.na(mxPH), !is.na(mnO2), !is.na(Cl), !is.na(NO3), !is.na(NH4), !is.na(oPO4
summary(algae.del)
```

```
##     season               size               speed                mxPH      
##  Length:184         Length:184         Length:184         Min.   :7.000  
##  Class :character   Class :character   Class :character   1st Qu.:7.777  
##  Mode  :character   Mode  :character   Mode  :character   Median :8.100  
##                                                           Mean   :8.078  
##                                                           3rd Qu.:8.400  
##                                                           Max.   :9.500  
##       mnO2              Cl               NO3              NH4          
##  Min.   : 1.500   Min.   :  0.80   Min.   : 0.050   Min.   :    5.80  
##  1st Qu.: 7.675   1st Qu.: 11.85   1st Qu.: 1.364   1st Qu.:   49.38  
##  Median : 9.750   Median : 35.08   Median : 2.820   Median :  115.71  
##  Mean   : 9.019   Mean   : 44.88   Mean   : 3.384   Mean   :  537.67  
##  3rd Qu.:10.700   3rd Qu.: 58.52   3rd Qu.: 4.540   3rd Qu.:  235.25  
##  Max.   :13.400   Max.   :391.50   Max.   :45.650   Max.   :24064.00  
##       oPO4             PO4              Chla               a1       
##  Min.   :  1.25   Min.   :  2.50   Min.   :  0.200   Min.   : 0.00  
##  1st Qu.: 18.56   1st Qu.: 50.34   1st Qu.:  2.075   1st Qu.: 1.40  
##  Median : 46.28   Median :115.60   Median :  5.522   Median : 4.85  
##  Mean   : 78.27   Mean   :146.58   Mean   : 13.883   Mean   :15.32  
##  3rd Qu.:102.83   3rd Qu.:220.25   3rd Qu.: 18.308   3rd Qu.:19.32  
##  Max.   :564.60   Max.   :771.60   Max.   :110.456   Max.   :89.80  
##        a2               a3               a4               a5        
##  Min.   : 0.000   Min.   : 0.000   Min.   : 0.000   Min.   : 0.000  
##  1st Qu.: 0.000   1st Qu.: 0.000   1st Qu.: 0.000   1st Qu.: 0.000  
##  Median : 3.600   Median : 1.700   Median : 0.000   Median : 2.650  
##  Mean   : 7.777   Mean   : 4.613   Mean   : 1.846   Mean   : 5.493  
##  3rd Qu.:11.700   3rd Qu.: 5.525   3rd Qu.: 2.425   3rd Qu.: 8.000  
##  Max.   :72.600   Max.   :42.800   Max.   :44.600   Max.   :44.400  
##        a6               a7        
##  Min.   : 0.000   Min.   : 0.000  
##  1st Qu.: 0.000   1st Qu.: 0.000  
##  Median : 0.000   Median : 1.000  
##  Mean   : 6.447   Mean   : 2.665  
##  3rd Qu.: 7.975   3rd Qu.: 2.700  
##  Max.   :77.600   Max.   :31.600
```

```
str(algae.del)
```

```
## tibble [184 x 18] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
##  $ season: chr [1:184] "winter" "spring" "autumn" "spring" ...
##  $ size  : chr [1:184] "small" "small" "small" "small" ...
##  $ speed : chr [1:184] "medium" "medium" "medium" "medium" ...
##  $ mxPH  : num [1:184] 8 8.35 8.1 8.07 8.06 8.25 8.15 8.05 8.7 7.93 ...
##  $ mnO2  : num [1:184] 9.8 8 11.4 4.8 9 13.1 10.3 10.6 3.4 9.9 ...
##  $ Cl    : num [1:184] 60.8 57.8 40 77.4 55.4 ...
##  $ NO3   : num [1:184] 6.24 1.29 5.33 2.3 10.42 ...
##  $ NH4   : num [1:184] 578 370 346.7 98.2 233.7 ...
##  $ oPO4  : num [1:184] 105 428.8 125.7 61.2 58.2 ...
##  $ PO4   : num [1:184] 170 558.8 187.1 138.7 97.6 ...
##  $ Chla  : num [1:184] 50 1.3 15.6 1.4 10.5 ...
##  $ a1    : num [1:184] 0 1.4 3.3 3.1 9.2 15.1 2.4 18.2 25.4 17 ...
##  $ a2    : num [1:184] 0 7.6 53.6 41 2.9 14.6 1.2 1.6 5.4 0 ...
##  $ a3    : num [1:184] 0 4.8 1.9 18.9 7.5 1.4 3.2 0 2.5 0 ...
##  $ a4    : num [1:184] 0 1.9 0 0 0 0 3.9 0 0 2.9 ...
##  $ a5    : num [1:184] 34.2 6.7 0 1.4 7.5 22.5 5.8 5.5 0 0 ...
##  $ a6    : num [1:184] 8.3 0 0 0 4.1 12.6 6.8 8.7 0 0 ...
##  $ a7    : num [1:184] 0 2.1 9.7 1.4 1 2.9 0 0 0 1.7 ...
##  - attr(*, "spec")=
##   .. cols(
##   ..   season = col_character(),
##   ..   size = col_character(),
##   ..   speed = col_character(),
##   ..   mxPH = col_double(),
##   ..   mnO2 = col_double(),
##   ..   Cl = col_double(),
##   ..   NO3 = col_double(),
##   ..   NH4 = col_double(),
##   ..   oPO4 = col_double(),
##   ..   PO4 = col_double(),
##   ..   Chla = col_double(),
##   ..   a1 = col_double(),
##   ..   a2 = col_double(),
##   ..   a3 = col_double(),
##   ..   a4 = col_double(),
##   ..   a5 = col_double(),
##   ..   a6 = col_double(),
##   ..   a7 = col_double()
##   .. )
```

There are a total of *184 observations* in algae.del.

(c)

```
algae.med = algae %>%
mutate_at(vars(mxPH,mnO2,Cl,NO3,NH4,oPO4,PO4,Chla),
          funs(ifelse(is.na(.)==TRUE,median(algae$.,na.rm = TRUE),.)))
```

```
## Warning: funs() is soft deprecated as of dplyr 0.8.0
## Please use a list of either functions or lambdas:
##
##   # Simple named list:
```

```
##   list(mean = mean, median = median)
##
##   # Auto named with `tibble::lst()`:
##   tibble::lst(mean, median)
##
##   # Using lambdas
##   list(~ mean(., trim = .2), ~ median(., na.rm = TRUE))
## This warning is displayed once per session.
```

```
str(algae.med)
```

```
## tibble [200 x 18] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
##  $ season: chr [1:200] "winter" "spring" "autumn" "spring" ...
##  $ size  : chr [1:200] "small" "small" "small" "small" ...
##  $ speed : chr [1:200] "medium" "medium" "medium" "medium" ...
##  $ mxPH  : num [1:200] 8 8.35 8.1 8.07 8.06 8.25 8.15 8.05 8.7 7.93 ...
##  $ mnO2  : num [1:200] 9.8 8 11.4 4.8 9 13.1 10.3 10.6 3.4 9.9 ...
##  $ Cl    : num [1:200] 60.8 57.8 40 77.4 55.4 ...
##  $ NO3   : num [1:200] 6.24 1.29 5.33 2.3 10.42 ...
##  $ NH4   : num [1:200] 578 370 346.7 98.2 233.7 ...
##  $ oPO4  : num [1:200] 105 428.8 125.7 61.2 58.2 ...
##  $ PO4   : num [1:200] 170 558.8 187.1 138.7 97.6 ...
##  $ Chla  : num [1:200] 50 1.3 15.6 1.4 10.5 ...
##  $ a1    : num [1:200] 0 1.4 3.3 3.1 9.2 15.1 2.4 18.2 25.4 17 ...
##  $ a2    : num [1:200] 0 7.6 53.6 41 2.9 14.6 1.2 1.6 5.4 0 ...
##  $ a3    : num [1:200] 0 4.8 1.9 18.9 7.5 1.4 3.2 0 2.5 0 ...
##  $ a4    : num [1:200] 0 1.9 0 0 0 0 3.9 0 0 2.9 ...
##  $ a5    : num [1:200] 34.2 6.7 0 1.4 7.5 22.5 5.8 5.5 0 0 ...
##  $ a6    : num [1:200] 8.3 0 0 0 4.1 12.6 6.8 8.7 0 0 ...
##  $ a7    : num [1:200] 0 2.1 9.7 1.4 1 2.9 0 0 0 1.7 ...
```

```
algae.med[48,]
```

```
## # A tibble: 1 x 18
##   season size  speed  mxPH  mnO2    Cl   NO3   NH4  oPO4   PO4  Chla    a1    a2
##   <chr>  <chr> <chr> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1 winter small low    8.06  12.6     9  0.23    10     5     6   1.1  35.5     0
## # ... with 5 more variables: a3 <dbl>, a4 <dbl>, a5 <dbl>, a6 <dbl>, a7 <dbl>
```

```
algae.med[62,]
```

```
## # A tibble: 1 x 18
##   season size  speed  mxPH  mnO2    Cl   NO3   NH4  oPO4   PO4  Chla    a1    a2
##   <chr>  <chr> <chr> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1 summer small medi~   6.4   9.8  32.7  2.68  103.  40.2    14  5.48  19.4     0
## # ... with 5 more variables: a3 <dbl>, a4 <dbl>, a5 <dbl>, a6 <dbl>, a7 <dbl>
```

```
algae.med[199,]
```

```
## # A tibble: 1 x 18
##   season size  speed  mxPH  mnO2    Cl   NO3   NH4  oPO4   PO4  Chla    a1    a2
##   <chr>  <chr> <chr> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1 winter large medi~     8   7.6  32.7  2.68  103.  40.2  103.  5.48     0  12.5
## # ... with 5 more variables: a3 <dbl>, a4 <dbl>, a5 <dbl>, a6 <dbl>, a7 <dbl>
```

There are a total or *200 observations* in algae.med

(d)

```
df = data.frame(algae.del[, 4:11])

cor(df, use = "pairwise.complete.obs" )
```

```
##             mxPH        mnO2          Cl        NO3         NH4        oPO4
## mxPH  1.00000000 -0.10269374  0.14709539 -0.1721302 -0.15429757  0.09022909
## mnO2 -0.10269374  1.00000000 -0.26324536  0.1179077 -0.07826816 -0.39375269
## Cl    0.14709539 -0.26324536  1.00000000  0.2109583  0.06598336  0.37925596
## NO3  -0.17213024  0.11790769  0.21095831  1.0000000  0.72467766  0.13301452
## NH4  -0.15429757 -0.07826816  0.06598336  0.7246777  1.00000000  0.21931121
## oPO4  0.09022909 -0.39375269  0.37925596  0.1330145  0.21931121  1.00000000
## PO4   0.10132957 -0.46396073  0.44519118  0.1570297  0.19939575  0.91196460
## Chla  0.43182377 -0.13121671  0.14295776  0.1454929  0.09120406  0.10691478
##             PO4        Chla
## mxPH  0.1013296  0.43182377
## mnO2 -0.4639607 -0.13121671
## Cl    0.4451912  0.14295776
## NO3   0.1570297  0.14549290
## NH4   0.1993958  0.09120406
## oPO4  0.9119646  0.10691478
## PO4   1.0000000  0.24849223
## Chla  0.2484922  1.00000000
```

```
model = lm(PO4~oPO4, data = algae)
summary(model)
```

```
##
## Call:
## lm(formula = PO4 ~ oPO4, data = algae)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -110.12  -36.34  -12.68   23.26  216.98
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   42.897      4.808   8.922 3.34e-16 ***
## oPO4           1.293      0.041  31.535  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 52.37 on 195 degrees of freedom
##   (3 observations deleted due to missingness)
## Multiple R-squared:  0.8361, Adjusted R-squared:  0.8352
## F-statistic: 994.5 on 1 and 195 DF,  p-value: < 2.2e-16
```

```r
predict(model,algae[28,"oPO4"])
```

```
##        1
## 48.06929
```

```r
#filling in the missing value in 'algae'
algae[28,"PO4"] = predict(model,algae[28,"oPO4"])
```

The value that we obtain for the missing value for PO4 based on oPO4 in the 28th observation is 48.06929.

(e)

We know from lecture that survivorship bias favors the values that appear but it ignores the values that did not appear. The bullet holes on planes example studies the concentration of the holes and the lack of bullet holes on the planes that did survive. We are not considerting the fact that missing values may be indicative of an outlying phenomenon by using imputed values.

(4)

(a)

```
set.seed(123)
cv = sample(cut(1:200,breaks = 5, label = FALSE))
cv
```

```
##   [1] 4 5 1 5 5 2 3 2 5 5 4 3 3 5 5 3 4 3 2 1 1 5 5 5 2 3 5 3 3 2 4 1 3 5 5 2 1
##  [38] 4 2 4 5 5 1 2 5 5 2 4 3 5 1 1 2 5 2 1 3 3 1 3 5 1 5 5 4 1 1 4 4 2 1 3 5 3
##  [75] 1 3 1 1 3 1 1 1 4 4 2 5 5 3 3 2 2 5 2 3 1 1 2 5 5 3 5 4 5 3 2 1 2 4 2 4 3
## [112] 1 4 5 4 5 5 4 3 1 2 1 2 4 4 3 4 2 1 4 5 1 2 2 3 1 4 5 4 4 4 2 3 5 1 2 1 4
## [149] 1 1 4 2 4 2 2 4 3 3 1 4 4 2 2 3 3 2 3 1 2 4 1 4 3 1 5 4 3 2 5 2 2 1 2 4 4
## [186] 4 3 5 5 4 3 1 2 3 3 3 4 3 2 1
```

(b)

```
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$a1)^2), # compute and store training error
             val.error = mean((predYvl - Yvl$a1)^2)) # compute and store test error
}

#My code
lapply(1:5, FUN = do.chunk, chunkdef = cv, dat = algae.med)
```

```
## [[1]]
##   fold train.error val.error
## 1    1    280.7503  322.0817
##
## [[2]]
##   fold train.error val.error
## 1    2    305.6518  229.1328
##
## [[3]]
##   fold train.error val.error
## 1    3    272.3432  360.8609
##
```

```
## [[4]]
##   fold train.error val.error
## 1    4    281.0626   422.842
##
## [[5]]
##   fold train.error val.error
## 1    5    270.3345  386.2931
```

(5)

```
algae.Test <- read_table2('algaeTest.txt',
                          col_names=c('season','size','speed','mxPH','mnO2','Cl','NO3',
                                      'NH4','oPO4','PO4','Chla','a1'),
                          na=c('XXXXXXX'))
```

```
## Parsed with column specification:
## cols(
##   season = col_character(),
##   size = col_character(),
##   speed = col_character(),
##   mxPH = col_double(),
##   mnO2 = col_double(),
##   Cl = col_double(),
##   NO3 = col_double(),
##   NH4 = col_double(),
##   oPO4 = col_double(),
##   PO4 = col_double(),
##   Chla = col_double(),
##   a1 = col_double()
## )
```

```
firstdata = algae.med[12]
newdata = algae.Test[12]

fit = lm(a1 ~ ., data = algae.med[1:12])

firstpredict = predict(fit, algae.med[1:11])
newpredict = predict(fit, algae.Test[1:11])

data.frame(train.error = mean((firstpredict - firstdata$a1)^2), val.error = mean((newpredict - newdata$
```

```
##   train.error val.error
## 1    286.2661  250.1794
```

Yes, this is what is roughly expected based of the CV estimated test error from number 4. The *train.error is 286.2661* which is very close to train.error predicted in number 4. The *val.error is 250.1794* which is not close to the predicted val.error besides the 2nd fold.

(6)

```r
library(ISLR)
```

```
## Warning: package 'ISLR' was built under R version 3.6.3
```

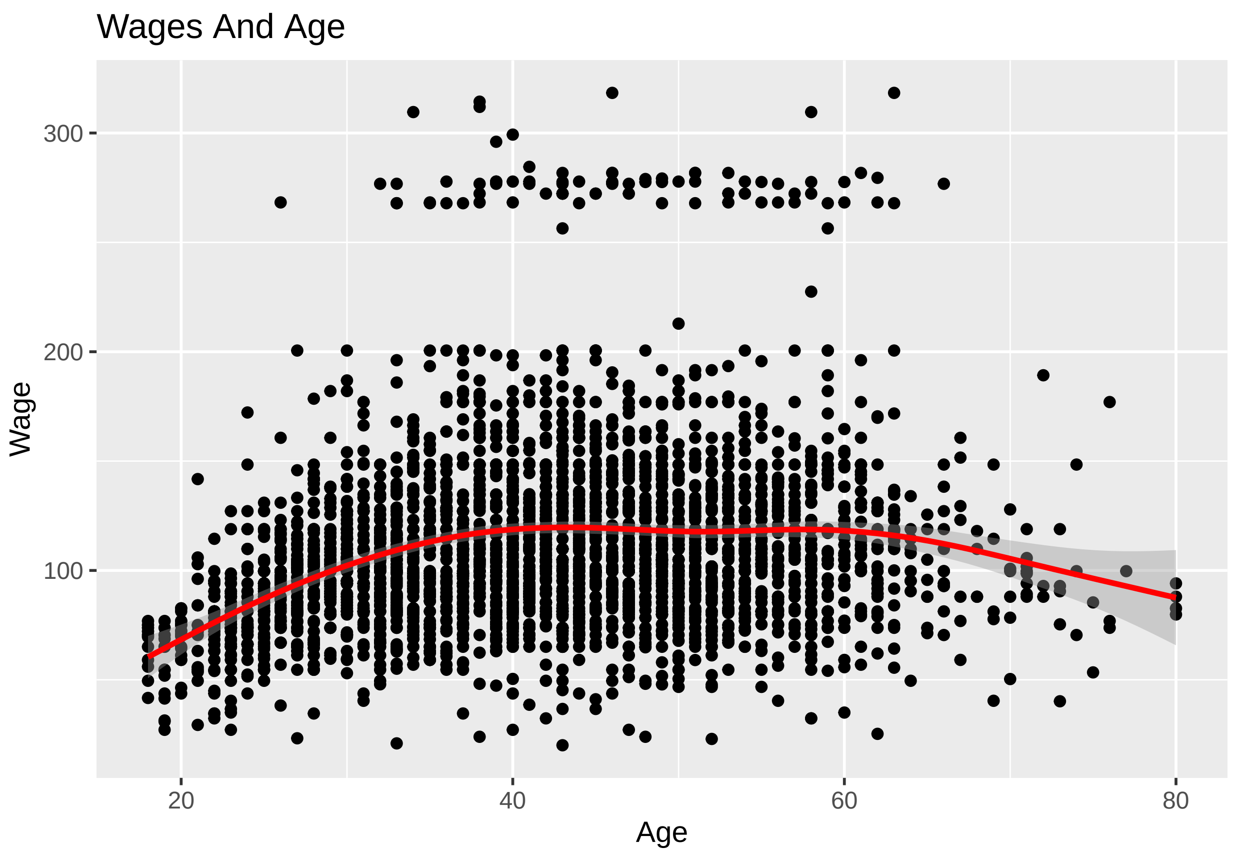
```r
head(Wage)
```

```
##        year age            maritl     race       education             region
## 231655 2006  18 1. Never Married 1. White    1. < HS Grad 2. Middle Atlantic
## 86582  2004  24 1. Never Married 1. White 4. College Grad 2. Middle Atlantic
## 161300 2003  45       2. Married 1. White 3. Some College 2. Middle Atlantic
## 155159 2003  43       2. Married 3. Asian 4. College Grad 2. Middle Atlantic
## 11443  2005  50      4. Divorced 1. White      2. HS Grad 2. Middle Atlantic
## 376662 2008  54       2. Married 1. White 4. College Grad 2. Middle Atlantic
##              jobclass         health health_ins  logwage      wage
## 231655  1. Industrial      1. <=Good      2. No 4.318063  75.04315
## 86582  2. Information 2. >=Very Good      2. No 4.255273  70.47602
## 161300  1. Industrial      1. <=Good     1. Yes 4.875061 130.98218
## 155159 2. Information 2. >=Very Good     1. Yes 5.041393 154.68529
## 11443  2. Information      1. <=Good     1. Yes 4.318063  75.04315
## 376662 2. Information 2. >=Very Good     1. Yes 4.845098 127.11574
```

(a)

```r
Wage %>% ggplot(aes(x = age, y = wage)) +
  geom_point() +
  geom_smooth(color = "red") +
  labs(title = "Wages And Age", x = "Age", y = "Wage")
```

```
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```



To plot this graph I used ggplot() along with geom_point() and geom_smooth(). I used the red color for the fit so that it'll stick out more. There is a pattern that we can observe and we see that the wages steadily increases until the age hits around 40 then remains constant until about age 60. Then we can see that the wage goes down slowly until the end. This matches exactly what I expected because as time goes on from you twenties, we tend to work on our skills which will give us promotions or better jobs. Then as our age passes 60, we have to start thinking about retirement or even retire.

(b)

(i):

```
attach(Wage)
x = lm(wage ~ 1 + age + I(age^2) + I(age^3) + I(age^4) + I(age^5) + I(age^6) + I(age^7)+ I(age^8)+ I(ag
summary(x)
```

```
##
## Call:
## lm(formula = wage ~ 1 + age + I(age^2) + I(age^3) + I(age^4) +
##     I(age^5) + I(age^6) + I(age^7) + I(age^8) + I(age^9) + I(age^10))
##
## 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)  1.773e+04  2.636e+04   0.672    0.501
## age         -4.259e+03  6.821e+03  -0.624    0.532
## I(age^2)     4.412e+02  7.726e+02   0.571    0.568
## I(age^3)    -2.585e+01  5.048e+01  -0.512    0.609
## I(age^4)     9.471e-01  2.109e+00   0.449    0.653
## I(age^5)    -2.257e-02  5.894e-02  -0.383    0.702
## I(age^6)     3.513e-04  1.117e-03   0.315    0.753
## I(age^7)    -3.476e-06  1.418e-05  -0.245    0.806
## I(age^8)     2.031e-08  1.156e-07   0.176    0.860
## I(age^9)    -5.868e-11  5.468e-10  -0.107    0.915
## I(age^10)    4.646e-14  1.141e-12   0.041    0.968
##
## Residual standard error: 39.89 on 2989 degrees of freedom
## Multiple R-squared:  0.08912,    Adjusted R-squared:  0.08607
## F-statistic: 29.24 on 10 and 2989 DF,  p-value: < 2.2e-16
```

(ii):

```
set.seed(123)
library(plyr)
```

```
## Warning: package 'plyr' was built under R version 3.6.3
```

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

```
## 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
```

```
## The following object is masked from 'package:purrr':
##
##     compact
```

```
do.chunk_2 <- function(chunkid, chunkdef, dat, a){ # function argument

  train = (chunkdef != chunkid)

  Xtr = dat[train,1:10] # get training set
  Ytr = dat[train,11] # get true response values in trainig set

  Xvl = dat[!train,1:10] # get validation set
  Yvl = dat[!train,11] # get true response values in validation set

  if (a == 0){
    lm.x = lm(wage~1, data = dat[train, 1:11])
  }
  else {
    lm.x = lm(wage~poly(x = age, degree = a, raw = FALSE),        data = dat[train, 1:11])
  }

  predYtr = predict(lm.x) # predict training values
  predYvl = predict(lm.x,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
}




cv = sample(cut(1:nrow(Wage),breaks = 5, label = FALSE))





df <- data.frame()
for (i in 0:10){
  ldply_out <- ldply(1:5, .fun = do.chunk_2,chunkdef = cv, dat = Wage, a = i)
  df <- rbind(df, ldply_out)
}

df
```

```
##    fold train.error val.error
## 1     1    1721.594  1817.527
## 2     2    1658.553  2069.315
## 3     3    1743.628  1729.083
## 4     4    1774.148  1606.896
## 5     5    1805.085  1484.879
## 6     1    1655.070  1750.247
## 7     2    1582.324  2042.891
## 8     3    1683.087  1639.366
## 9     4    1710.395  1529.071
## 10    5    1738.332  1419.121
## 11    1    1577.534  1680.028
## 12    2    1511.439  1945.377
## 13    3    1604.291  1572.308
## 14    4    1632.043  1461.176
## 15    5    1662.475  1341.647
## 16    1    1572.410  1673.979
## 17    2    1507.355  1935.783
## 18    3    1599.535  1565.365
## 19    4    1626.202  1458.256
## 20    5    1655.864  1342.242
## 21    1    1570.350  1672.057
## 22    2    1505.834  1931.884
## 23    3    1597.794  1562.194
## 24    4    1624.176  1456.262
## 25    5    1652.975  1344.246
## 26    1    1569.490  1673.654
## 27    2    1505.115  1932.817
## 28    3    1597.749  1561.237
## 29    4    1623.832  1455.656
## 30    5    1652.459  1344.168
## 31    1    1567.805  1673.954
## 32    2    1504.501  1929.448
## 33    3    1596.183  1560.627
## 34    4    1623.032  1452.553
## 35    5    1650.304  1346.741
## 36    1    1566.402  1675.732
## 37    2    1503.501  1929.308
## 38    3    1595.927  1558.254
## 39    4    1622.292  1451.201
## 40    5    1649.215  1346.675
## 41    1    1566.388  1676.108
## 42    2    1503.474  1929.203
## 43    3    1594.927  1564.856
## 44    4    1621.855  1456.666
## 45    5    1648.872  1348.656
## 46    1    1564.032  1673.956
## 47    2    1501.270  1926.369
## 48    3    1592.954  1560.655
## 49    4    1619.690  1453.864
## 50    5    1645.779  1350.076
## 51    1    1563.755  1676.266
## 52    2    1501.270  1926.371
## 53    3    1592.941  1560.768
## 54    4    1619.586  1454.893
## 55    5    1645.736  1350.627
```

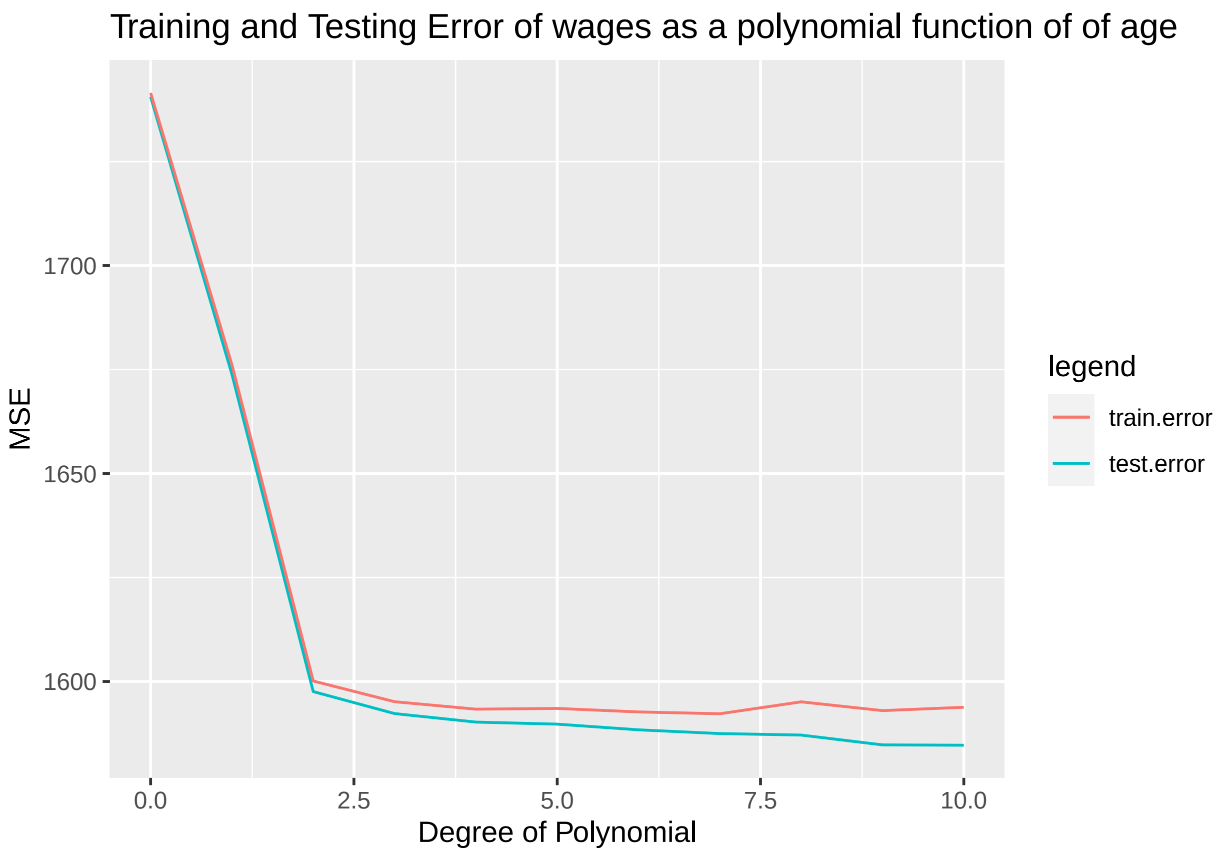
```
degree_0 <- colMeans(ldply(1:5, .fun = do.chunk_2, chunkdef = cv, dat = Wage, a= 0))

degree_1 <- colMeans(ldply(1:5, .fun = do.chunk_2, chunkdef = cv, dat = Wage, a= 1))

degree_2 <- colMeans(ldply(1:5, .fun = do.chunk_2, chunkdef = cv, dat = Wage, a= 2))

degree_3 <- colMeans(ldply(1:5, .fun = do.chunk_2, chunkdef = cv, dat = Wage, a= 3))

degree_4 <- colMeans(ldply(1:5, .fun = do.chunk_2, chunkdef = cv, dat = Wage, a= 4))

degree_5 <- colMeans(ldply(1:5, .fun = do.chunk_2, chunkdef = cv, dat = Wage, a= 5))

degree_6 <- colMeans(ldply(1:5, .fun = do.chunk_2, chunkdef = cv, dat = Wage, a= 6))
```

```
degree_7 <- colMeans(ldply(1:5, .fun = do.chunk_2, chunkdef = cv, dat = Wage, a= 7))

degree_8 <- colMeans(ldply(1:5, .fun = do.chunk_2, chunkdef = cv, dat = Wage, a= 8))

degree_9 <- colMeans(ldply(1:5, .fun = do.chunk_2, chunkdef = cv, dat = Wage, a= 9))

degree_10 <- colMeans(ldply(1:5, .fun = do.chunk_2, chunkdef = cv, dat = Wage, a= 10))




df2 = as.data.frame(rbind(degree_0, degree_1, degree_2, degree_3, degree_4, degree_5, degree_6, degree_
df2$degree = 0:10
df2[-1]
```

```
##           train.error val.error degree
## degree_0     1740.601  1741.540      0
## degree_1     1673.842  1676.139      1
## degree_2     1597.556  1600.107      2
## degree_3     1592.273  1595.125      3
## degree_4     1590.226  1593.329      4
## degree_5     1589.729  1593.506      5
## degree_6     1588.365  1592.664      6
## degree_7     1587.467  1592.234      7
## degree_8     1587.103  1595.098      8
## degree_9     1584.745  1592.984      9
## degree_10    1584.658  1593.785     10
```

(c):

```
df2 %>% ggplot() +
  geom_line(aes(x = degree, y = train.error,color = "red"), na.rm   = TRUE) +
  geom_line(aes(x = degree, y = val.error, color = "blue"))+
  scale_color_discrete("legend",labels = c("train.error", "test.error")) +
  labs(title = "Training and Testing Error of wages as a polynomial function of of age", x = "Degree of
```



As p increases the training error and the testing error both goes significantly down after 1. However we can see that the training error has a higher error than the test error. At around Degree 10 we can see that it has the minimum test error. So we choose the model at degree *10* because we want to minimize the test errors.