Data Mining - Final Project

Stephanie Langeland

December 15, 2016

Abstract

I use exploratory data analysis to display and understand trends in weather data from Weather Underground. After exploring the data, I use actual mean temperature, actual minimum temperature, actual maximum temperature, average minimum temperature, average maximum temperature, record minimum temperature, record maximum temperature, actual precipitation, and record precipitation to predict whether average precipitation is above or below the overall average precipitation rate of 0.11. Linear and quadratic models are fit using lasso, logit, and partial least squares regression. The goal is to find a model with the lowest MSE (mean squared error) and most importantly, the highest rate of prediction accuracy in the testing data.

Weather Dataset:

The original source of this weather data is Weather Underground, an internet based weather service provider. The dataset was available and downloaded from GitHub.

Column/Variable	Description
date	The date of the weather record, formatted YYYY-M-D
actual_mean_temp	The measured average temperature for that day
actual_min_temp	The measured minimum temperature for that day
actual_max_temp	The measured maximum temperature for that day
average_min_temp	The average minimum temperature on that day since 1880
average_max_temp	The average maximum temperature on that day since 1880
record_min_temp	The lowest ever temperature on that day since 1880
record_max_temp	The highest ever temperature on that day since 1880
record_min_temp_year	The year that the lowest ever temperature occurred
record_max_temp_year	The year that the highest ever temperature occurred
actual_precipitation	The measured amount of rain or snow for that day
average_precipitation	The average amount of rain or snow on that day since 1880
record_precipitation	The highest amount of rain or snow on that day since 1880

Exploratory Data Analysis

```
weather <- read.csv("https://raw.githubusercontent.com/fivethirtyeight/data/master/us-weather-history/KCLT.csv") #path to data

weather$date <- as.Date(weather$date) #"YYYY-MM-DD" format
summary(is.na(weather)) #verify that there are no missing values
```

```
##     date         actual_mean_temp actual_min_temp actual_max_temp
##  Mode :logical   Mode :logical    Mode :logical   Mode :logical
##  FALSE:365       FALSE:365        FALSE:365       FALSE:365
##  NA's :0         NA's :0          NA's :0         NA's :0
##  average_min_temp average_max_temp record_min_temp record_max_temp
##  Mode :logical    Mode :logical    Mode :logical   Mode :logical
##  FALSE:365        FALSE:365        FALSE:365       FALSE:365
##  NA's :0          NA's :0          NA's :0         NA's :0
##  record_min_temp_year record_max_temp_year actual_precipitation
##  Mode :logical        Mode :logical        Mode :logical
##  FALSE:365            FALSE:365            FALSE:365
##  NA's :0              NA's :0              NA's :0
##  average_precipitation record_precipitation
##  Mode :logical         Mode :logical
##  FALSE:365             FALSE:365
##  NA's :0               NA's :0
```

```
dim(weather)
```

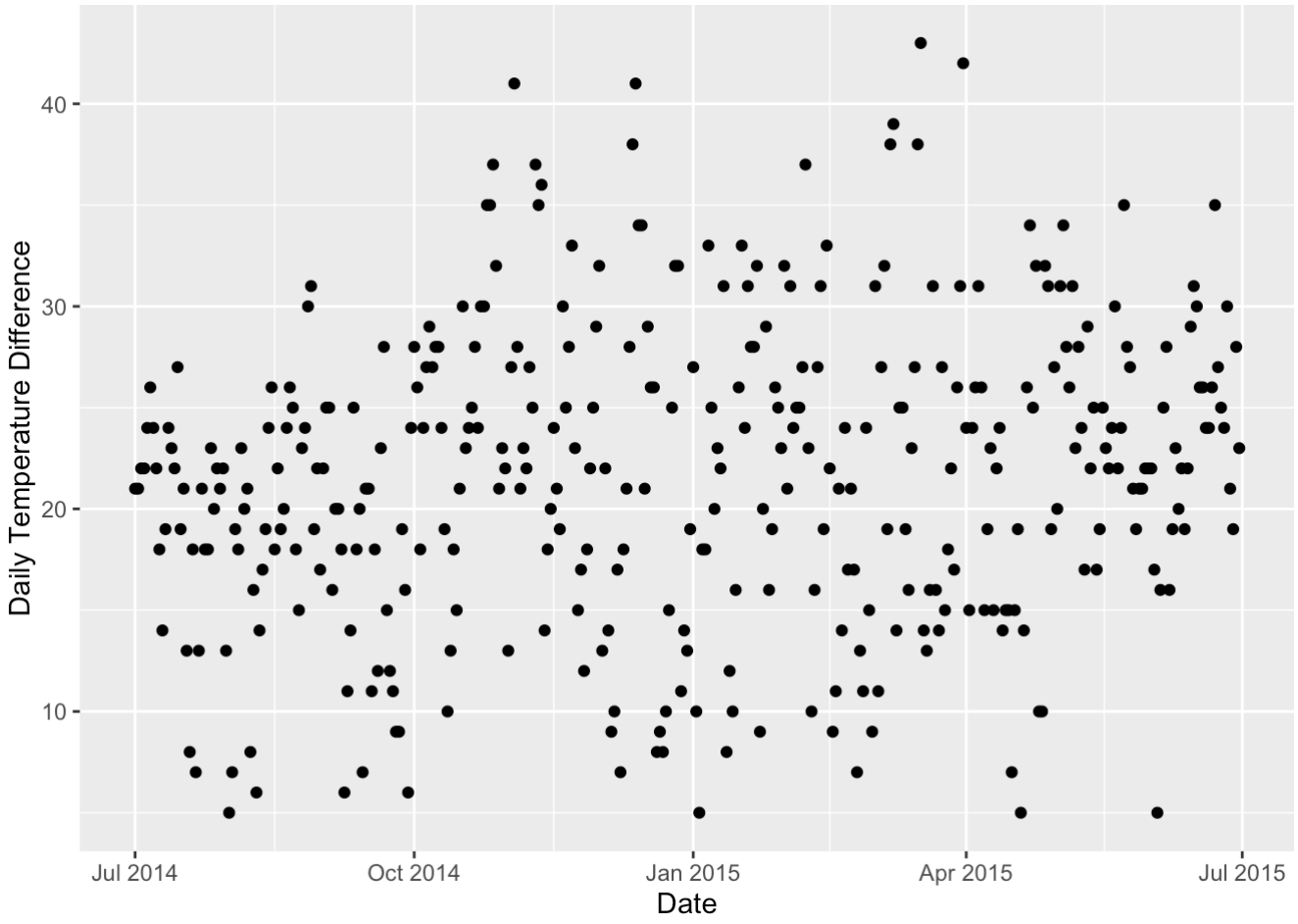
```
## [1] 365  13
```

```
str(weather)
```

```
## 'data.frame':    365 obs. of  13 variables:
##  $ date                 : Date, format: "2014-07-01" "2014-07-02" ...
##  $ actual_mean_temp     : int  81 85 82 75 72 74 79 83 80 78 ...
##  $ actual_min_temp      : int  70 74 71 64 60 61 67 72 71 71 ...
##  $ actual_max_temp      : int  91 95 93 86 84 87 91 94 89 85 ...
##  $ average_min_temp     : int  67 68 68 68 68 68 68 68 68 68 ...
##  $ average_max_temp     : int  89 89 89 89 89 89 89 89 89 89 ...
##  $ record_min_temp      : int  56 56 56 55 57 57 55 58 57 53 ...
##  $ record_max_temp      : int  104 101 99 99 100 99 100 101 101 101 ...
##  $ record_min_temp_year : int  1919 2008 2010 1933 1967 1964 1972 1892 1891 1961 ...
##  $ record_max_temp_year : int  2012 1931 1931 1955 1954 1948 1954 2010 1986 1926 ...
##  $ actual_precipitation : num  0 0 0.14 0 0 0 0 0 0.15 0 ...
##  $ average_precipitation: num  0.1 0.1 0.11 0.1 0.1 0.1 0.11 0.11 0.12 0.11 ...
##  $ record_precipitation : num  5.91 1.53 2.5 2.63 1.65 1.95 2.37 1.87 3.71 2.45 ...
```

```
weather$month <- weather$date #create month variable
weather$month <- format(weather$month, format = "%B") #format month variable
weather$year <- weather$date #create year variable
weather$year <- format(weather$year, format = "%Y") #format year variable
```

```
#Create a daily temperature difference column:
weather$daily_temp_diff <- weather$actual_max_temp - weather$actual_min_temp
weather$daily_temp_diff <- as.numeric(weather$daily_temp_diff)

#When is that most variable time of year in terms of daily temperature changes?
library(ggplot2)
ggplot(data = weather) +
  geom_point(mapping = aes(x = date, y = daily_temp_diff)) +
  xlab("Date") +
  ylab("Daily Temperature Difference")
```




```
#Although there are major outliers betwen October and April,
#this graph does not provide much clarity since the points are scattered.
#The median daily difference in temperature will be calculated, so
#we can see which days have greater daily temperature differences than the median.

#Median daily temperature difference:
( max(weather$daily_temp_diff) + min(weather$daily_temp_diff) ) / 2
```

```
## [1] 24
```

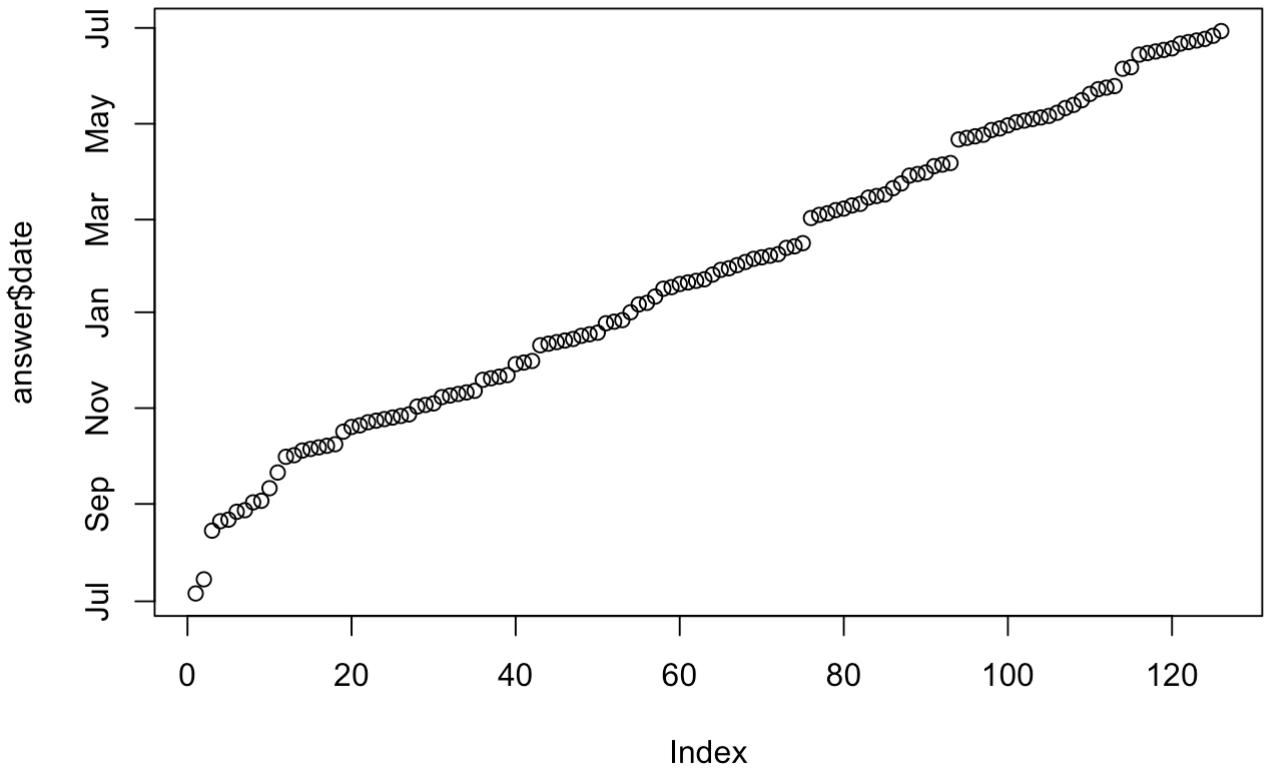
```
#The median the daily temperature change is 24 degrees.

#Which days have daily temperature differences above the median temperature (24 degree
s)?
library(dplyr)
```

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

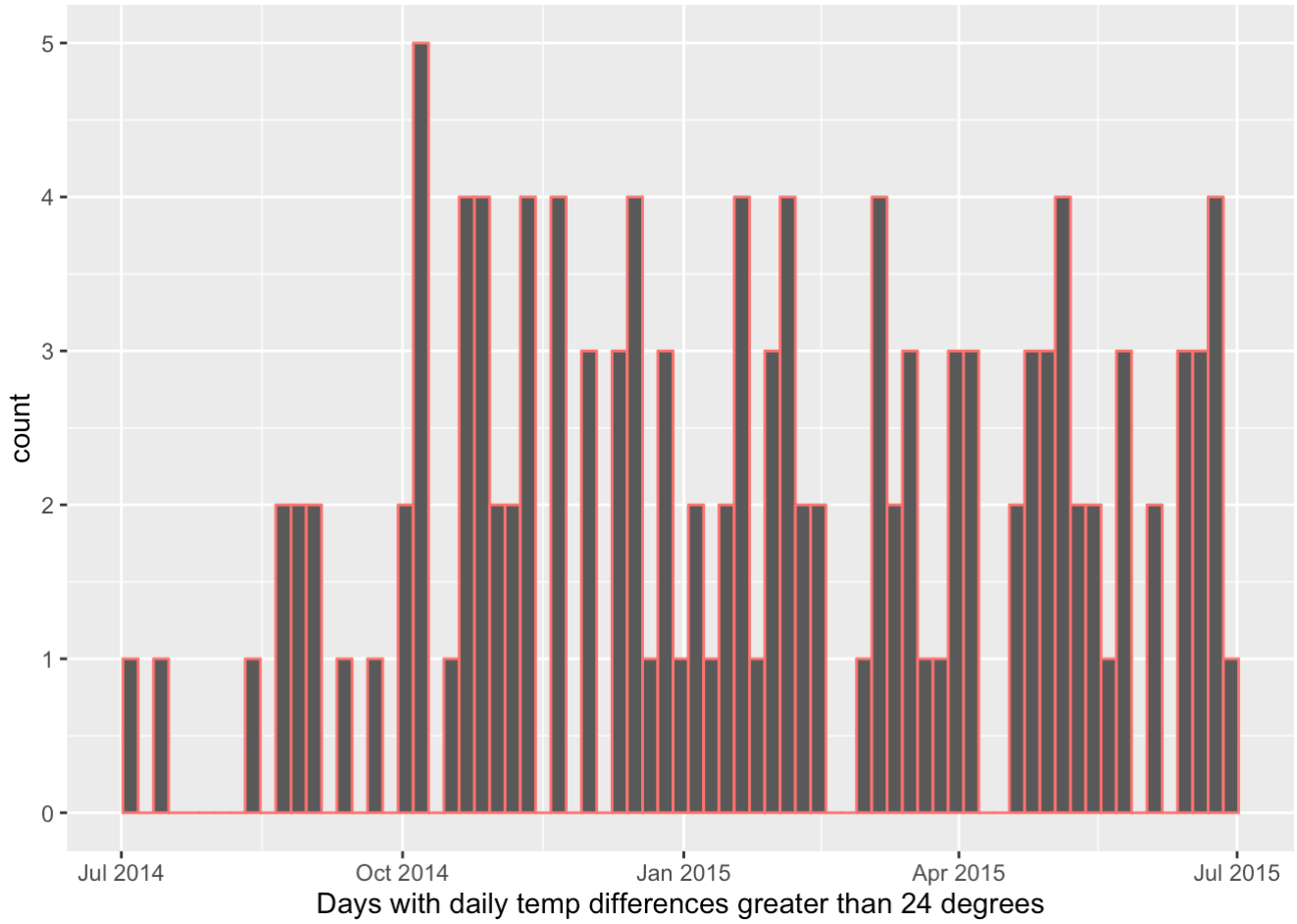
```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

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

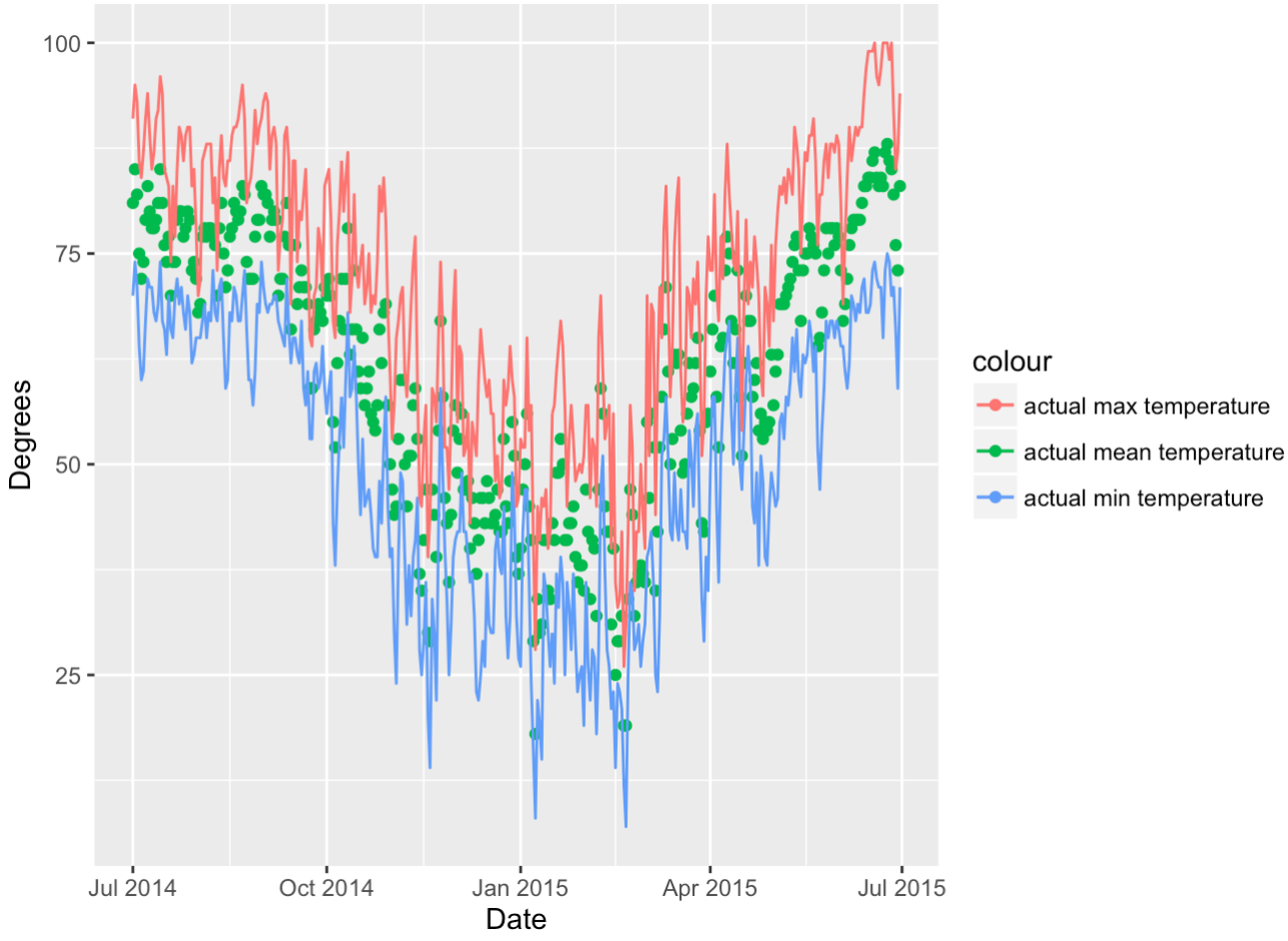
```
greater24 <- group_by(weather, date) %>%
  summarize(new = sort(daily_temp_diff > 24))
truerows <- which(greater24$new == TRUE)
answer <- greater24[truerows, ]
plot(x = answer$date)
```



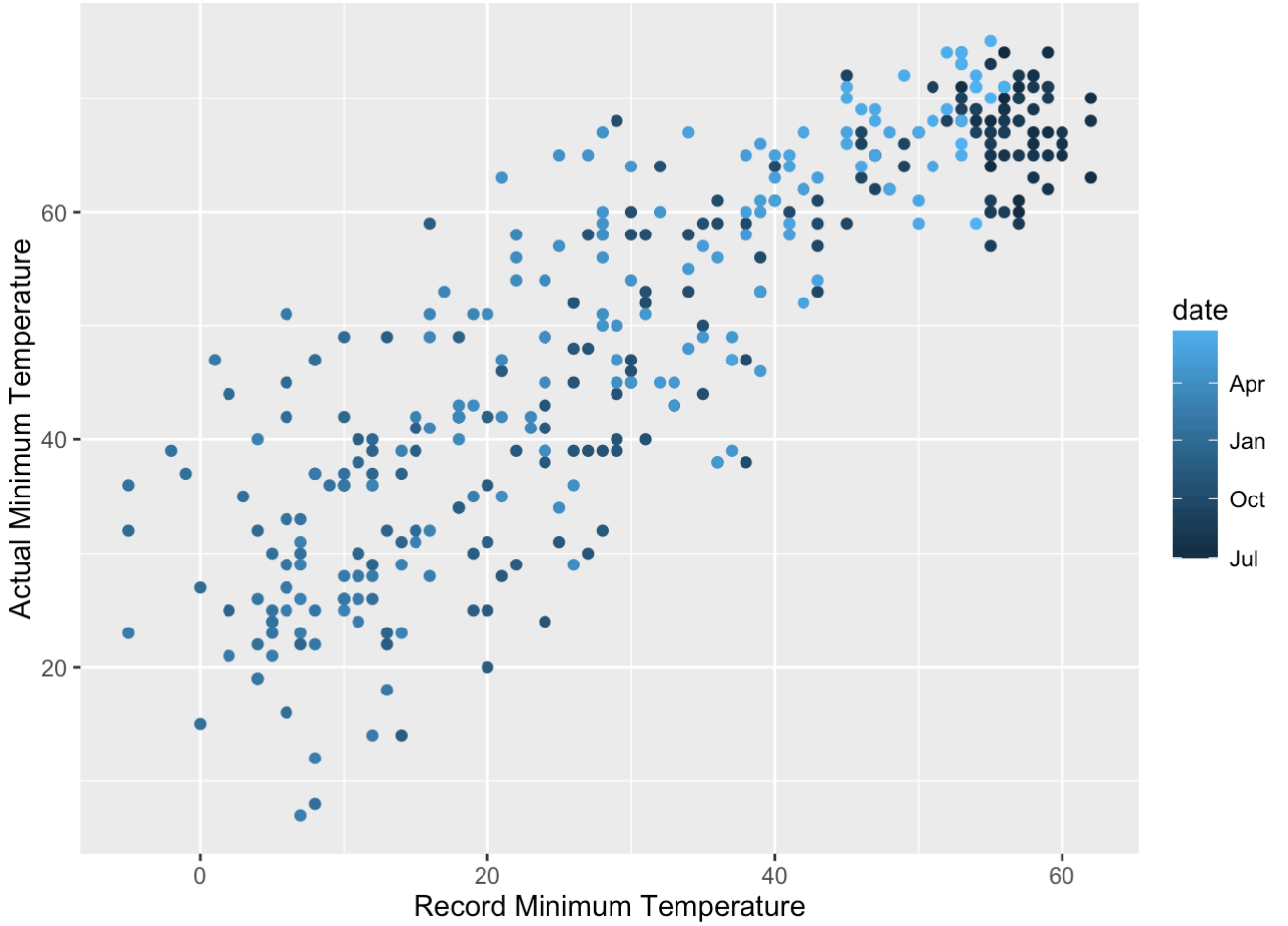
```
#This plot does not show much granularity, a histogram should represent the
#data more clearly.

ggplot(answer) +
  geom_histogram(mapping = aes(x = answer$date,
                               col = "red"),
                 show.legend = FALSE,
                 binwidth =  5) +
  xlab("Days with daily temp differences greater than 24 degrees")
```



```
#October - July seem to have the most erratic temperature changes up to 4
#degrees above the median.  October has the highest count of days 5 degrees
#above the median temperature.  October shows the most variability in
#temperature differences per day.

#Next, an analysis of actual maximum, minimum, and average temperatures.
#Are there any major outliers?
#Temperatures by Date:
ggplot(weather) +
  geom_point(mapping = aes(x = date, y = actual_mean_temp,
                           color = "actual mean temperature"),
             show.legend = TRUE) +
  geom_line(mapping = aes(x = date, y = actual_min_temp,
                          color = "actual min temperature"),
            show.legend = TRUE) +
  geom_line(mapping = aes(x = date, y = actual_max_temp,
                          color = "actual max temperature"),
            show.legend = TRUE) +
  xlab("Date") +
  ylab("Degrees")
```



```
#We see outliers between January and April when mean temperatures overlap with
#minimum temperatures rather than floating between maximum and minimum temperatures.
#This may be caused by unseasonably cold days in those months dragging down the average.

#A similiar pattern occurred in April when the average temperature overlapped with the
#maximum temperatures.  This too may have been caused by unseasonably warm days bringing
#up the mean.

#Next, a comparison of record versus actual minimum temperatures:
ggplot(data = weather) +
  geom_point(mapping = aes(x = record_min_temp, y = actual_min_temp,
                           col = date)) +
  xlab("Record Minimum Temperature") +
  ylab("Actual Minimum Temperature")
```



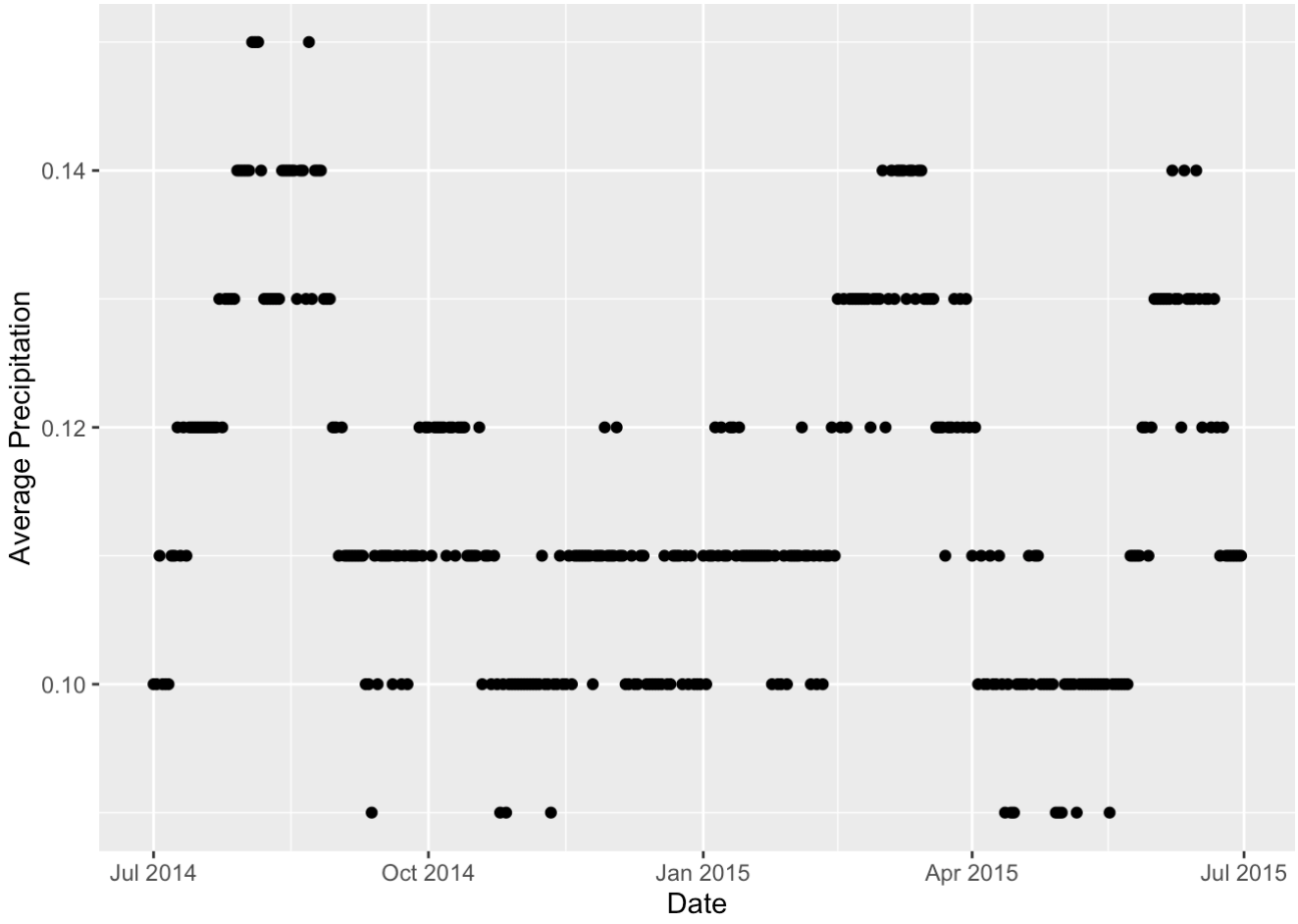

```
#The relationship between record and actual minimum temperatures seems
#to be intuitive in that they have a positive relationship.  There aren't
#any major outliers. These variables are likely correlated.

#Next, what are the top 10 warmest days?
t <- order(weather$actual_max_temp, decreasing = TRUE)
head(t, n = 10)
```

```
##  [1] 353 357 358 359 361 350 351 352 360 349
```

```
weather[c("353", "357", "358", "359", "361", "350", "351", "352", "360", "349"), c("dat
e")]
```

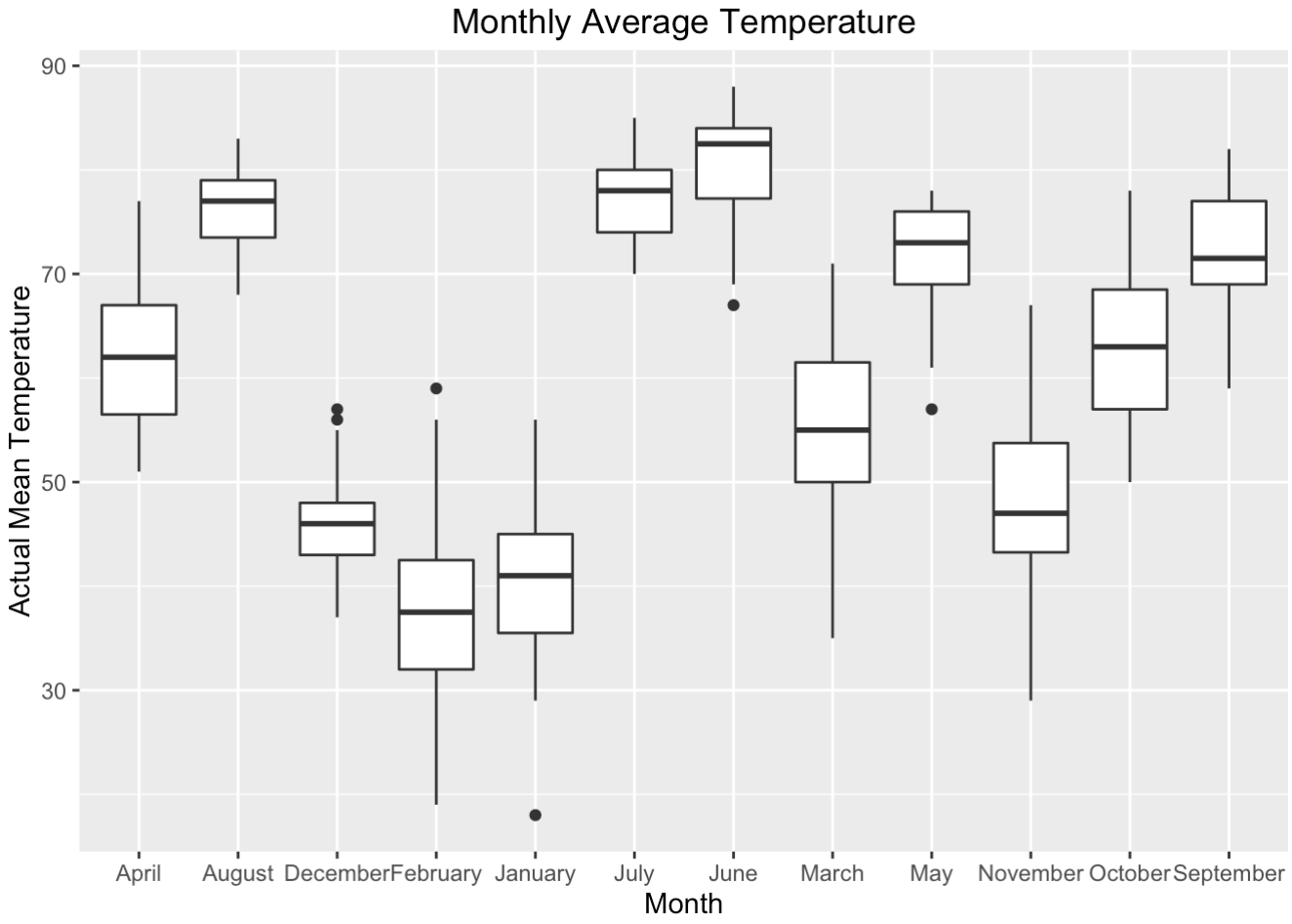
```
##  [1] "2015-06-18" "2015-06-22" "2015-06-23" "2015-06-24" "2015-06-26"
##  [6] "2015-06-15" "2015-06-16" "2015-06-17" "2015-06-25" "2015-06-14"
```

```
#There seems to have been a heatwave in June 2015.

#Finally, precipitation is examined in order to understand whether prediction models
#can be built using these data:
ggplot(weather) +
  geom_point(mapping = aes(x = date, y = average_precipitation)) +
  ylab("Average Precipitation") +
  xlab("Date")
```




```
#This graph using daily data is not very clear.  The next section focuses on monthly
#data for both precipitation and temperature, which may help spot trends more easily.
```

```
#Are monthly average temperatures wide ranging?
monthly_avg_temp <- aggregate(actual_mean_temp ~ month, data = weather, mean)
monthly_avg_temp$actual_mean_temp <- round(monthly_avg_temp$actual_mean_temp, digits =
3)

ggplot(weather) +
  geom_boxplot(mapping = aes(x = month, y = actual_mean_temp)) +
  ggtitle("Monthly Average Temperature") +
  xlab("Month") +
  ylab("Actual Mean Temperature")
```



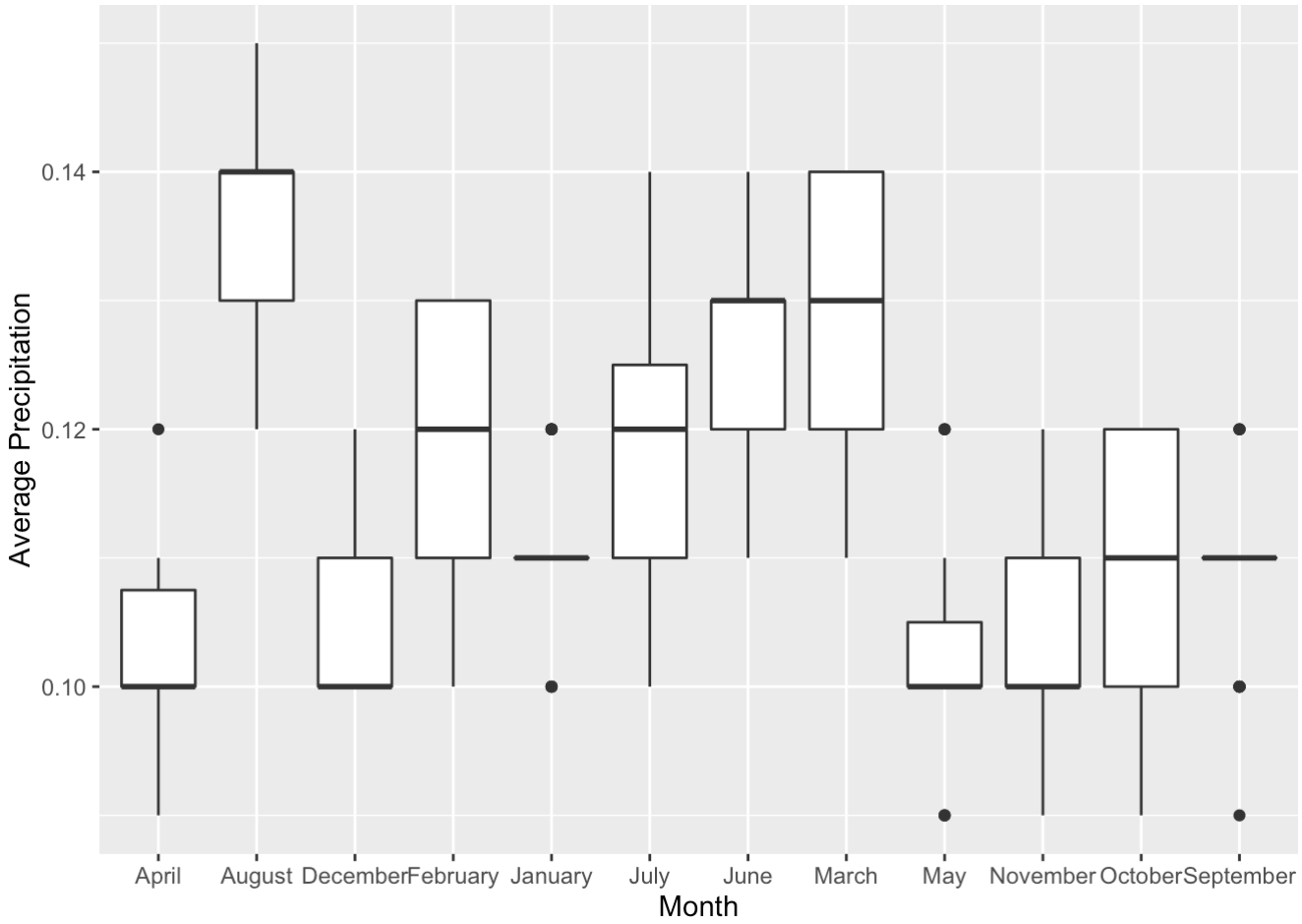

```
#February, March, November, and October have higher Interquartile Ranges than the
#other months, denoted by longer whiskers on the box plot above. These months have
#the most wide ranging average temperatures.

#Mean temperature per month:
matrix(cbind(c("April", "August", "December", "February", "January", "July", "June",
               "March", "May", "November", "October", "September"),
             monthly_avg_temp$actual_mean_temp), ncol = 2)
```

```
##       [,1]        [,2]
##  [1,] "April"     "62.467"
##  [2,] "August"    "76.581"
##  [3,] "December"  "45.968"
##  [4,] "February"  "37.714"
##  [5,] "January"   "40.516"
##  [6,] "July"      "77.742"
##  [7,] "June"      "80.333"
##  [8,] "March"     "54.387"
##  [9,] "May"       "71.935"
## [10,] "November"  "47.533"
## [11,] "October"   "63.226"
## [12,] "September" "72.533"
```

```
#The overall average for the actual mean temparature variable is:
mean_temp <- round(mean(weather$actual_mean_temp), digits = 2)
mean_temp
```

```
## [1] 61.05
```

```
#Next, let's examine monthly precipitation and whether it can be used as a predictor:
monthly_avg_precip <- aggregate(average_precipitation ~ month, data = weather, mean)
monthly_avg_precip$average_precipitation <- round(monthly_avg_precip$average_precipitati
on,
                                                  digits = 3)

ggplot(weather) +
  geom_boxplot(mapping = aes(x = month, y = average_precipitation)) +
  xlab("Month") +
  ylab("Average Precipitation")
```




```
#Monthly average precipitation data are much more skewed than monthly average temperatur
es.
#We can see the skewed data by looking at the median line of the box plots.  For exampl
e,
#we can see that April precipitation is heavily skewed right whereas the August precipit
ation
#data are skewed left, and February's data seem to be relatively normally distributed.

#Mean precipitation per month:
matrix(cbind(c("April", "August", "December", "February", "January", "July", "June",
               "March", "May", "November", "October", "September"),
             monthly_avg_precip$average_precipitation), ncol = 2)
```

```
##       [,1]        [,2]
##  [1,] "April"     "0.101"
##  [2,] "August"    "0.136"
##  [3,] "December"  "0.105"
##  [4,] "February"  "0.119"
##  [5,] "January"   "0.11"
##  [6,] "July"      "0.119"
##  [7,] "June"      "0.125"
##  [8,] "March"     "0.129"
##  [9,] "May"       "0.103"
## [10,] "November"  "0.105"
## [11,] "October"   "0.11"
## [12,] "September" "0.108"
```

```
#The overall average for the average precipitation variable is:
mean_precip <- round(mean(weather$average_precipitation), digits = 2)
mean_precip
```

```
## [1] 0.11
```

Predictions

Can actual mean temperature, actual minimum temperature, actual maximum temperature, average minimum temperature, average maximum temperature, record minimum temperature, record maximum temperature, actual precipitation, and record precipitation predict whether average precipitation is above or below the overall average precipitation rate of 0.11?

Training and Testing Datasets

```
set.seed(12345)
train <- sample(nrow(weather), 182)
#Exclude date, record_min_temp_year, record_max_temp_year, month, year, and
#daily_temp_diff:
training <- weather[train, -c(1, 9:10, 14:16)]
testing <- weather[-train, -c(1, 9:10, 14:16)]
```

Note: All `ols` models are linear and all `qlm` models are quadatric.

Models

Explore whether the model is linear or takes on a different form.

A) Linear `ols` Model

```
ols <- lm(average_precipitation ~ ., data = training, y = TRUE)
ols
```

```
##
## Call:
## lm(formula = average_precipitation ~ ., data = training, y = TRUE)
##
## Coefficients:
##          (Intercept)      actual_mean_temp       actual_min_temp
##            0.1348612             0.0006298            -0.0003971
##      actual_max_temp      average_min_temp      average_max_temp
##           -0.0004811             0.0031694            -0.0024900
##      record_min_temp       record_max_temp  actual_precipitation
##           -0.0003081             0.0003192             0.0004735
## record_precipitation
##           -0.0013061
```

```
MSE_ols <- mean( (testing$average_precipitation - predict(ols, newdata = testing)) ^ 2)
MSE_ols
```

```
## [1] 0.0001710528
```

```
ols_tbl <- table(testing$average_precipitation, predict(ols, newdata = testing) > 0.11)
ols_tbl
```

```
##
##        FALSE TRUE
##   0.09     2    2
##   0.1     26   16
##   0.11    13   40
##   0.12     8   28
##   0.13     6   23
##   0.14     5   11
##   0.15     0    3
```

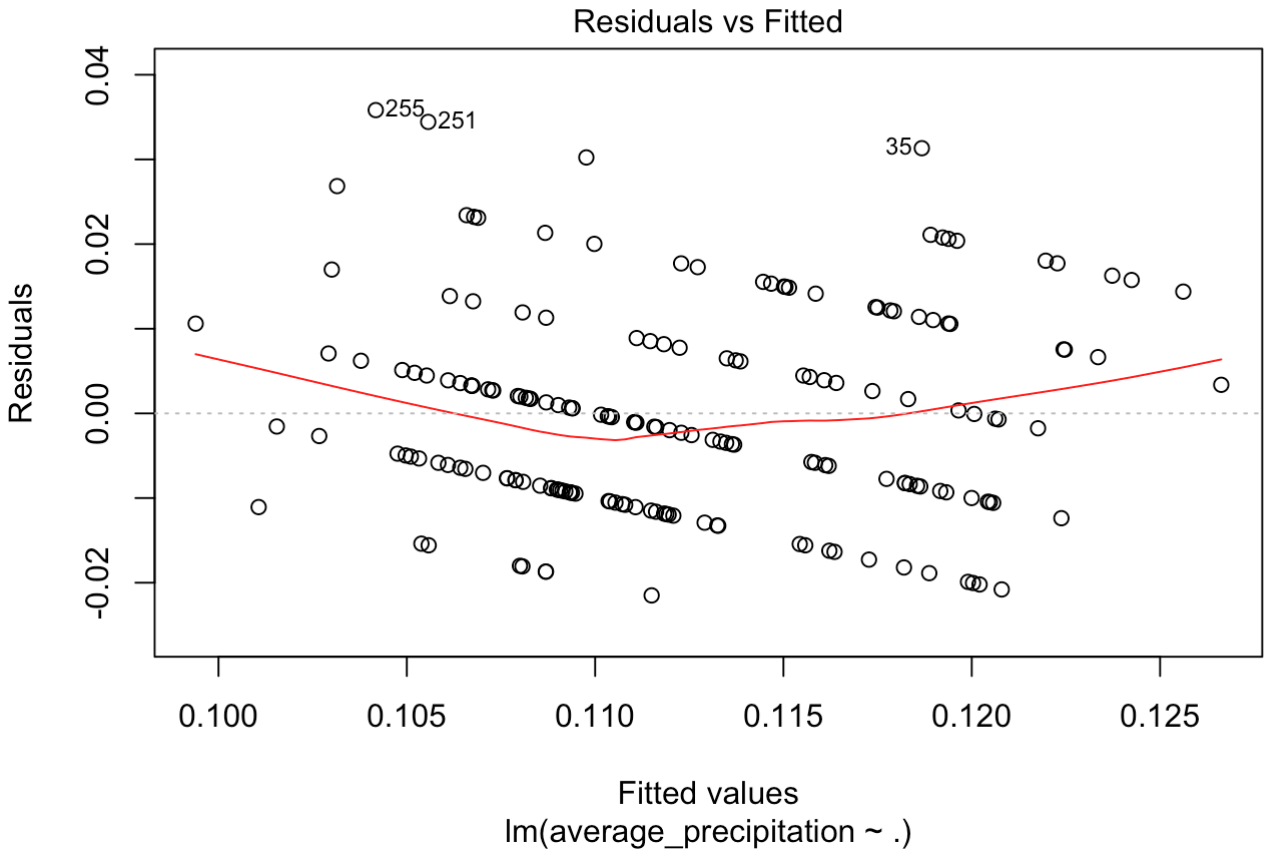
```
table(predict(ols, newdata = testing) > 0.11)
```

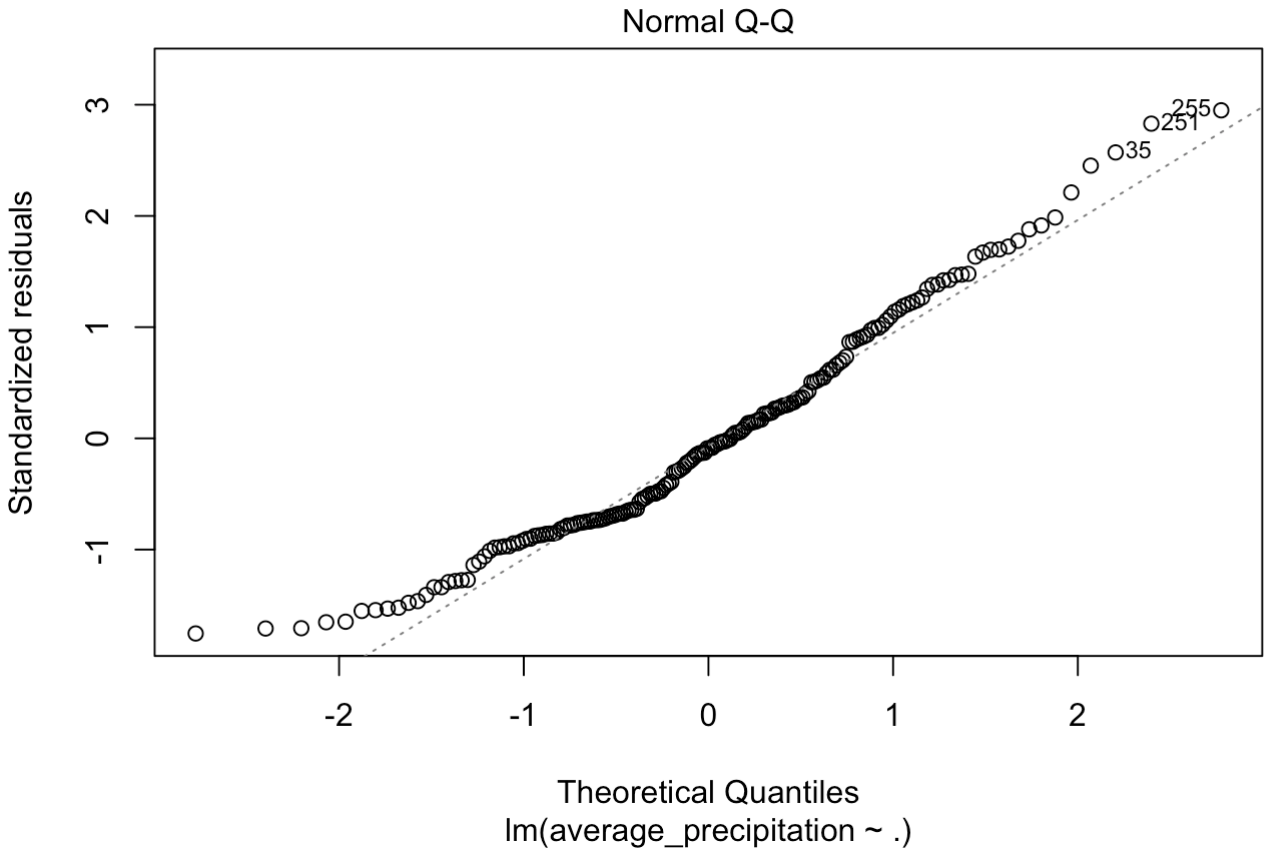
```
##
## FALSE  TRUE
##    60   123
```

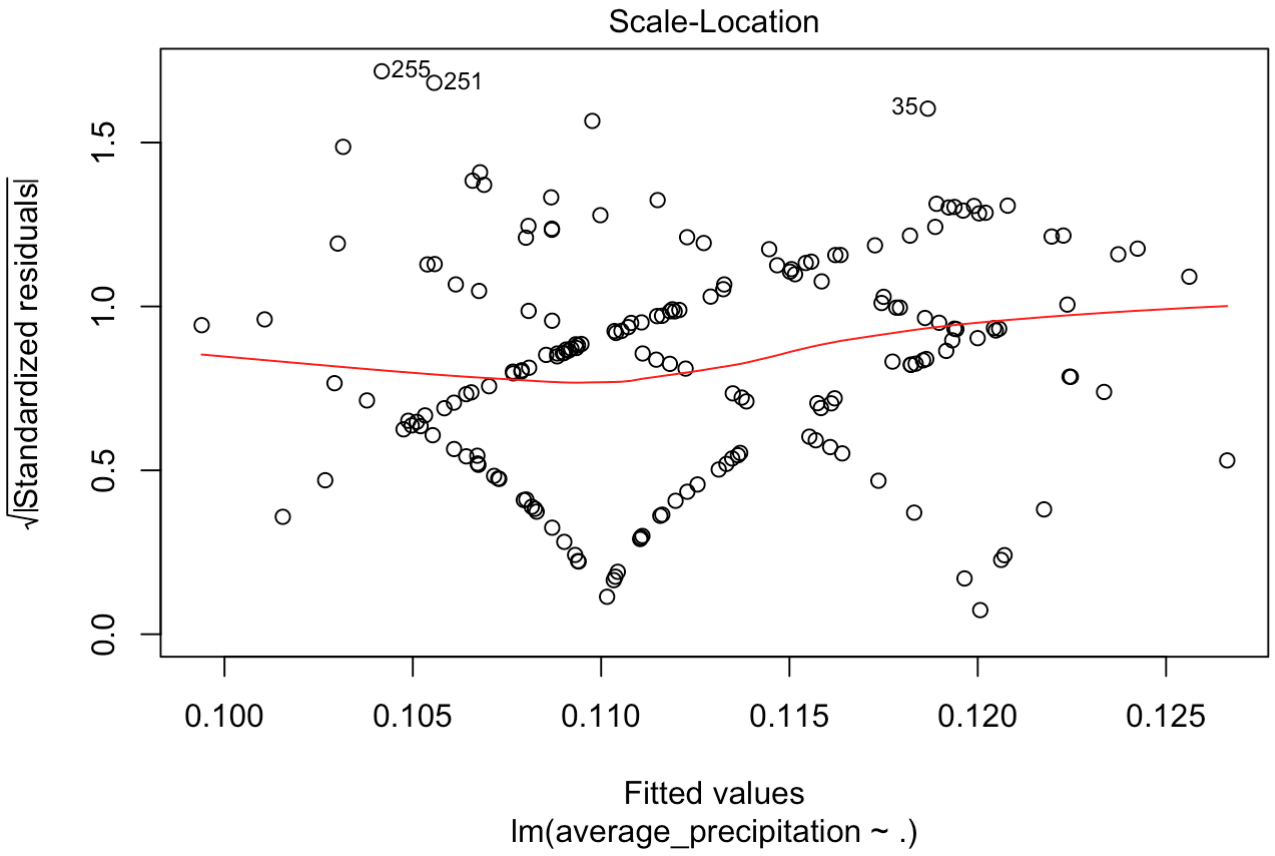
```
ols_pred_acc <- 123 / (60 + 123)
ols_pred_acc #67% prediction accuracy in the testing data
```

```
## [1] 0.6721311
```

```
plot(ols)
```


Residuals vs Fitted
Residuals
0.04
0.02
0.00
-0.02
255
251
35
0.100
0.105
0.110
0.115
0.120
0.125
Fitted values
lm(average_precipitation ~ .)


Normal Q-Q
Standardized residuals
3
2
1
0
-1
255
251
35
-2
-1
0
1
2
Theoretical Quantiles
lm(average_precipitation ~ .)

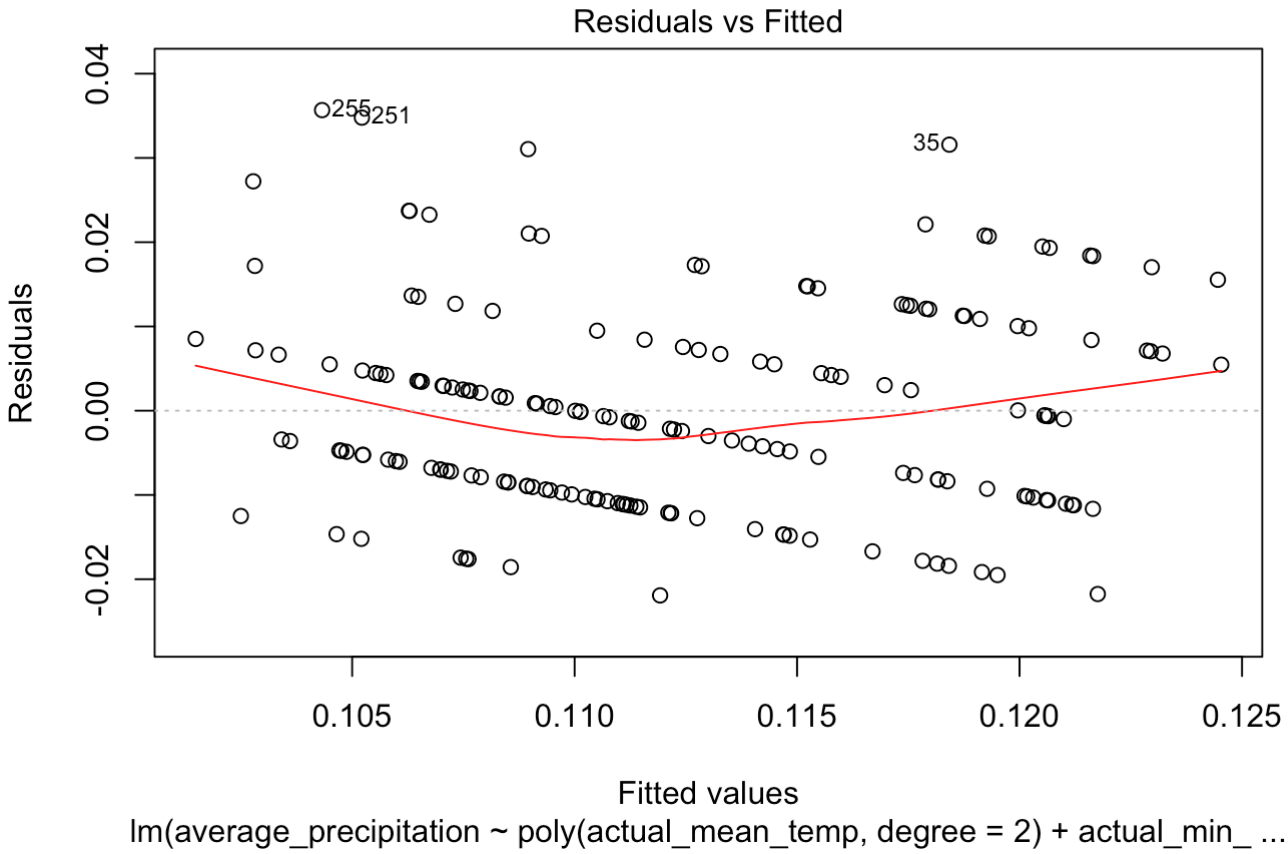

Scale-Location
255
251
35
√|Standardized residuals|
0.0
0.5
1.0
1.5
0.100
0.105
0.110
0.115
0.120
0.125
Fitted values
lm(average_precipitation ~ .)

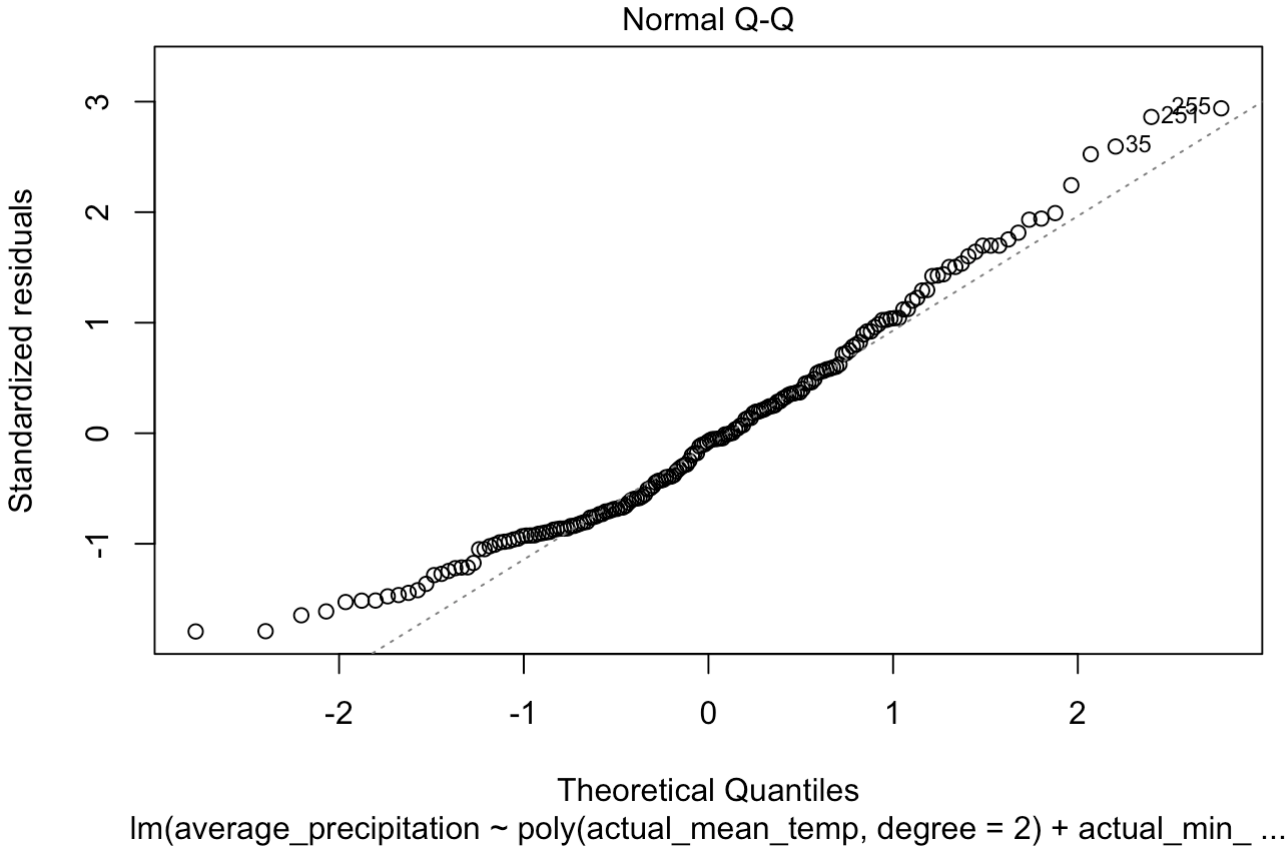

Residuals vs Leverage
255
260
23
Standardized residuals
-2
-1
0
1
2
3
Cook's distance
0.00
0.05
0.10
0.15
0.20
Leverage
lm(average_precipitation ~ .)

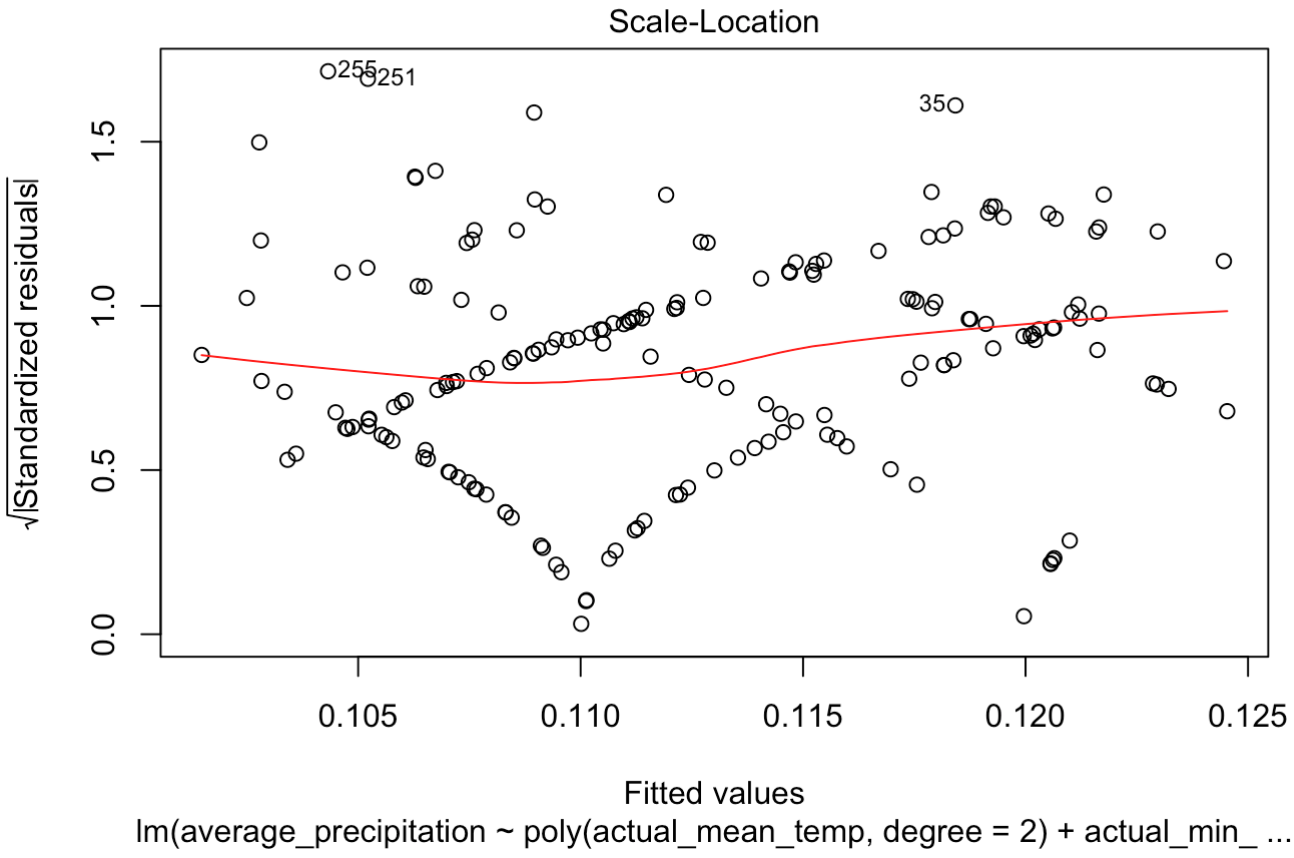
```
#the fitted values versus residuals plot shows us that the
#relationship may be quadratic rather than linear
```

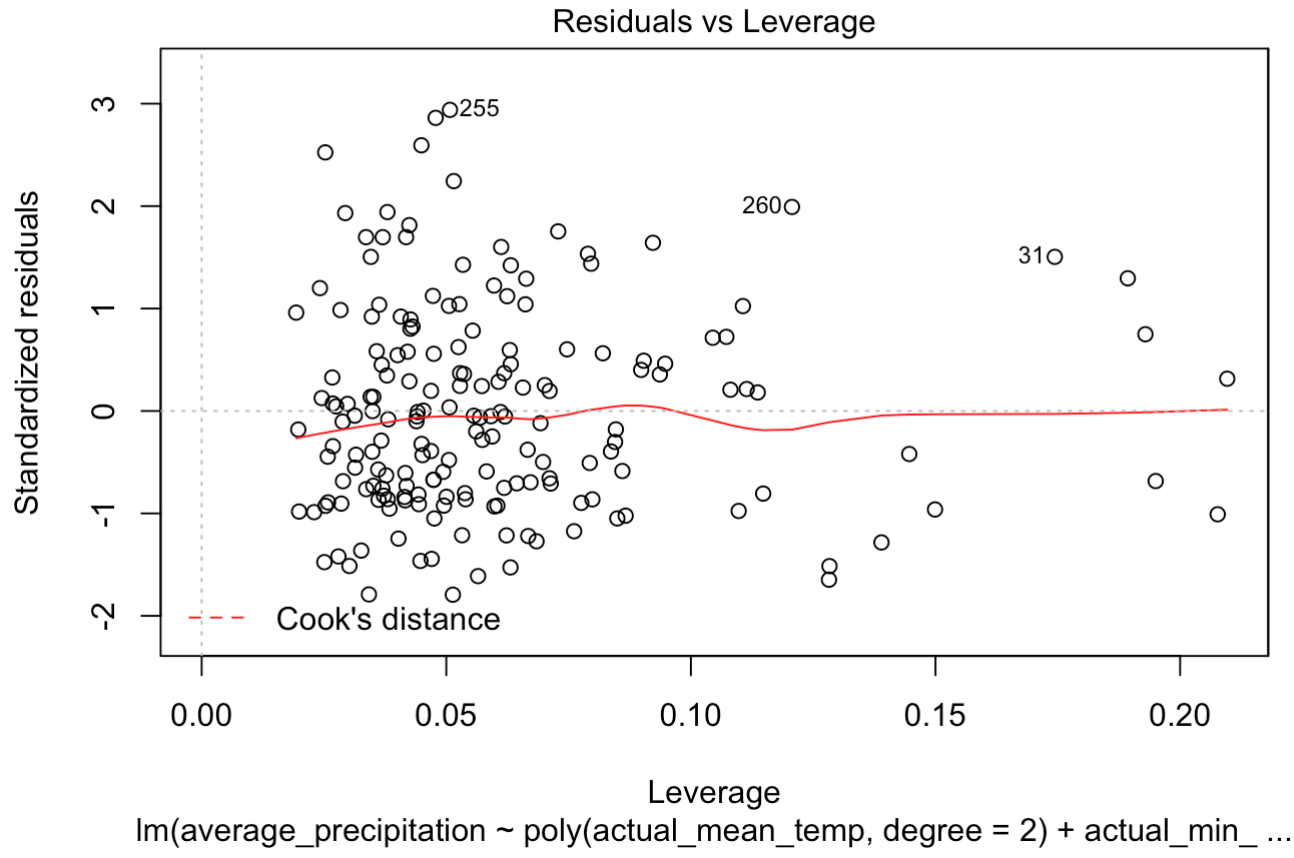
B) Quadratic `qlm` Model

```
qlm <- lm(average_precipitation ~ poly(actual_mean_temp, degree = 2)  + actual_min_temp
+
                                            actual_max_temp + average_min_temp +
                                            average_max_temp + record_min_temp +
                                            record_max_temp + actual_precipitation +
                                            record_precipitation,
              data = training, y = TRUE)
#Note that many different versions of this model were run placing the quadratic
#term on different variables.  This model seemed to have the best fit.

plot(qlm)
```


Residuals vs Fitted
Residuals
0.04
0.02
0.00
-0.02
255
251
35
0.105
0.110
0.115
0.120
0.125
Fitted values
lm(average_precipitation ~ poly(actual_mean_temp, degree = 2) + actual_min_ ...


Normal Q-Q
Standardized residuals
3
2
1
0
-1
255
251
35
-2
-1
0
1
2
Theoretical Quantiles
lm(average_precipitation ~ poly(actual_mean_temp, degree = 2) + actual_min_ ...


Scale-Location
255
251
35
√|Standardized residuals|
0.0
0.5
1.0
1.5
0.105
0.110
0.115
0.120
0.125
Fitted values
lm(average_precipitation ~ poly(actual_mean_temp, degree = 2) + actual_min_ ...


Residuals vs Leverage
255
260
31
Standardized residuals
-2
-1
0
1
2
3
Cook's distance
0.00
0.05
0.10
0.15
0.20
Leverage
lm(average_precipitation ~ poly(actual_mean_temp, degree = 2) + actual_min_ ...

```
#the fitted values versus residuals plot shows us that the
#relationship is quadratic
qlm
```

```
##
## Call:
## lm(formula = average_precipitation ~ poly(actual_mean_temp, degree = 2) +
##     actual_min_temp + actual_max_temp + average_min_temp + average_max_temp +
##     record_min_temp + record_max_temp + actual_precipitation +
##     record_precipitation, data = training, y = TRUE)
##
## Coefficients:
##                          (Intercept)  poly(actual_mean_temp, degree = 2)1
##                            0.1958452                            0.2397236
## poly(actual_mean_temp, degree = 2)2                       actual_min_temp
##                            0.0164384                           -0.0005916
##                       actual_max_temp                    average_min_temp
##                           -0.0006839                            0.0027679
##                    average_max_temp                     record_min_temp
##                           -0.0022505                           -0.0002443
##                     record_max_temp                 actual_precipitation
##                            0.0003354                            0.0006553
##                 record_precipitation
##                           -0.0011503
```

```
MSE_qlm <- mean( (testing$average_precipitation - predict(qlm, newdata = testing)) ^ 2)
MSE_qlm
```

```
## [1] 0.0001680992
```

```
qlm_tbl <- table(testing$average_precipitation, predict(qlm, newdata = testing) > 0.11)
qlm_tbl
```

```
##
##        FALSE TRUE
##   0.09     2    2
##   0.1     29   13
##   0.11    16   37
##   0.12     9   27
##   0.13     6   23
##   0.14     5   11
##   0.15     0    3
```

```
table(predict(qlm, newdata = testing) > 0.11)
```

```
##
## FALSE  TRUE
##    67   116
```

```
qlm_pred_acc <- 116 / (67 + 166)
qlm_pred_acc #50% prediction accuracy in the testing data
```

```
## [1] 0.4978541
```

Linear and Quadratic Models

C) Lasso `ols` Model

```
X <- model.matrix(formula(ols), data = training)
library(lars)
```

```
## Loaded lars 1.2
```

```
lasso <- lars(X, ols$y)
new_X <- model.matrix(formula(ols), data = testing)
MSE_lasso <- colMeans( (testing$average_precipitation - predict(lasso, newx =
new_X)$fit) ^ 2)
best_lasso <- which.min(MSE_lasso)
MSE_lasso[best_lasso]
```

```
## [1] 0.0001703469
```

```
b <- coef(lasso)[best_lasso,]
b <- b[b != 0]
b
```

```
##      actual_min_temp      actual_max_temp     average_min_temp
##        -5.881308e-05        -1.773217e-04         2.198216e-03
##     average_max_temp      record_min_temp      record_max_temp
##        -1.518463e-03        -1.638855e-04         4.647529e-05
## record_precipitation
##        -8.673963e-04
```

```
library(glmpath)
```

```
## Loading required package: survival
```

```
y <- training$average_precipitation
path1 <- glmpath(X, y, nopenalty.subset = 1,
                 family = 'binomial')
path1
```

```
## Call:
## glmpath(x = X, y = y, nopenalty.subset = 1, family = "binomial")
## Step 1 :  (Intercept)  average_min_temp
## Step 3 :  actual_max_temp
## Step 6 :  record_precipitation
## Step 8 :  actual_mean_temp
## Step 9 :  average_max_temp
## Step 11 :  record_min_temp
## Step 13 :  record_max_temp
## Step 14 :  actual_min_temp
## Step 16 :  actual_precipitation
## Step 17 : - actual_mean_temp
## Step 19 :  actual_mean_temp
```

```
y_hat_path1 <- predict(path1, newx = new_X,
                       type = "response",
                       s = 11)
z_path1 <- as.integer(y_hat_path1 > 0.11)
y <- testing$average_precipitation
table(y, z_path1)
```

```
##       z_path1
## y       0  1
##   0.09  1  3
##   0.1  19 23
##   0.11 18 35
##   0.12  6 30
##   0.13  5 24
##   0.14  6 10
##   0.15  0  3
```

```
table(z_path1)
```

```
## z_path1
##   0   1
##  55 128
```

```
lasso_ols_pred_acc <- 128 / (55 +128)
lasso_ols_pred_acc #70% prediction accuracy in the testing data
```

```
## [1] 0.6994536
```

D) Lasso `qlm` Model

```
X <- model.matrix(formula(qlm), data = training)
library(lars)
lasso1 <- lars(X, qlm$y)
new_X <- model.matrix(formula(qlm), data = testing)
MSE_lasso1 <- colMeans( (testing$average_precipitation - predict(lasso1,
                                                                 newx = new_X)$fit) ^ 2)
best_lasso1 <- which.min(MSE_lasso1)
MSE_lasso1[best_lasso1]
```

```
## [1] 0.0001656807
```

```
b1 <- coef(lasso1)[best_lasso1,]
b1 <- b1[b1 != 0]
b1
```

```
## poly(actual_mean_temp, degree = 2)2                    actual_min_temp
##                        2.765210e-02                      -3.128199e-06
##                    actual_max_temp                    average_min_temp
##                      -1.457693e-04                       1.136639e-03
##                    average_max_temp               actual_precipitation
##                      -7.525967e-04                       1.720654e-04
##               record_precipitation
##                      -4.325534e-04
```

```
library(glmpath)
y <- training$average_precipitation
path2 <- glmpath(X, y, nopenalty.subset = 1,
                 family = 'binomial')
path2
```

```
## Call:
## glmpath(x = X, y = y, nopenalty.subset = 1, family = "binomial")
## Step 1 :  (Intercept)  average_min_temp
## Step 3 :  poly(actual_mean_temp, degree = 2)2
## Step 4 :  actual_max_temp
## Step 8 :  actual_precipitation
## Step 11 :  record_precipitation
## Step 13 :  average_max_temp
## Step 14 :  actual_min_temp
## Step 16 :  record_min_temp
## Step 18 :  record_max_temp
## Step 20 :  poly(actual_mean_temp, degree = 2)1
```

```
y_hat_path2 <- predict(path2, newx = new_X,
                       type = "response",
                       s = 10)
z_path2 <- as.integer(y_hat_path2 > 0.11)
y <- testing$average_precipitation
table(y, z_path2)
```

```
##       z_path2
## y       0  1
##   0.09  1  3
##   0.1  24 18
##   0.11 19 34
##   0.12  7 29
##   0.13  6 23
##   0.14  6 10
##   0.15  0  3
```

```
table(z_path2)
```

```
## z_path2
##   0   1
##  63 120
```

```
lasso1_pred_acc <- 120 / (63 + 120)
lasso1_pred_acc #66% prediction accuracy in the testing data
```

```
## [1] 0.6557377
```

E) Partial Least Squares (PLS) `ols` Model

```
library(pls)
```

```
##
## Attaching package: 'pls'
```

```
## The following object is masked from 'package:stats':
##
##     loadings
```

```
pls.options(parallel = parallel::detectCores()) #does corss-validation in parallel
PLS <- plsr(formula(ols), data = training)
MSE_PLS <- colMeans( (testing$average_precipitation -
                        predict(PLS, newdata = testing)) ^ 2)
best_PLS <- which.min(MSE_PLS)
best_PLS
```

```
## [1] 6
```

```
MSE_PLS[best_PLS]
```

```
## [1] 0.0001678042
```

```
coef(PLS)
```

```
## , , 9 comps
##
##                       average_precipitation
## actual_mean_temp              0.0006298136
## actual_min_temp              -0.0003970600
## actual_max_temp              -0.0004811139
## average_min_temp              0.0031694297
## average_max_temp             -0.0024900305
## record_min_temp              -0.0003081003
## record_max_temp               0.0003192480
## actual_precipitation          0.0004735205
## record_precipitation         -0.0013060781
```

```
PLS_pred <- predict(PLS, newdata = testing, ncomp = 9, type = "response") > 0.11
table(testing$average_precipitation, PLS_pred)
```

```
##       PLS_pred
##        FALSE TRUE
##   0.09     2    2
##   0.1     26   16
##   0.11    13   40
##   0.12     8   28
##   0.13     6   23
##   0.14     5   11
##   0.15     0    3
```

```
table(PLS_pred)
```

```
## PLS_pred
## FALSE  TRUE
##    60   123
```

```
pls_ols_pred_acc <- 123 / (60 + 123)
pls_ols_pred_acc #67% prediction accuracy in the testing data
```

```
## [1] 0.6721311
```

F) Partial Least Squares (PLS) `qlm` Model

```
library(pls)
pls.options(parallel = parallel::detectCores()) #does corss-validation in parallel
PLS1 <- plsr(formula(qlm), data = training)
MSE_PLS1 <- colMeans( (testing$average_precipitation -
                         predict(PLS1, newdata = testing)) ^ 2)
best_PLS1 <- which.min(MSE_PLS1)
best_PLS1
```

```
## [1] 6
```

```
MSE_PLS1[best_PLS1]
```

```
## [1] 0.0001677366
```

```
coef(PLS1)
```

```
## , , 10 comps
##
##                                     average_precipitation
## poly(actual_mean_temp, degree = 2)1          0.2397236443
## poly(actual_mean_temp, degree = 2)2          0.0164384333
## actual_min_temp                             -0.0005916156
## actual_max_temp                             -0.0006839473
## average_min_temp                             0.0027678603
## average_max_temp                            -0.0022505462
## record_min_temp                             -0.0002442992
## record_max_temp                              0.0003354228
## actual_precipitation                         0.0006552620
## record_precipitation                        -0.0011503218
```

```
PLS_pred1 <- predict(PLS1, newdata = testing, ncomp = 10, type = "response") > 0.11
table(testing$average_precipitation, PLS_pred1)
```

```
##       PLS_pred1
##        FALSE TRUE
##   0.09     2    2
##   0.1     29   13
##   0.11    16   37
##   0.12     9   27
##   0.13     6   23
##   0.14     5   11
##   0.15     0    3
```

```
table(PLS_pred1)
```

```
## PLS_pred1
## FALSE  TRUE
##    67   116
```

```
PLS1_pred_acc <- 116 / (67 + 116)
PLS1_pred_acc #63% prediction accuracy in the testing data
```

```
## [1] 0.6338798
```

G) Logit `ols` Model

```
logit <- glm(ols, data = training,
             family = binomial) #estimate a logit model in the training data
```

```
## Warning: non-integer #successes in a binomial glm!
```

```
MSE_logit <- mean( (testing$average_precipitation - predict(logit, newdata = testing)) ^
 2)
MSE_logit
```

```
## [1] 4.740276
```

```
logit_tbl <- table(testing$average_precipitation, predict(logit, newdata = testing,
                                                          type = "response") > 0.11)
logit_tbl
```

```
##
##        FALSE TRUE
##   0.09     2    2
##   0.1     27   15
##   0.11    13   40
##   0.12     9   27
##   0.13     6   23
##   0.14     5   11
##   0.15     0    3
```

```
table(predict(logit, newdata = testing, type = "response") > 0.11)
```

```
##
## FALSE  TRUE
##    62   121
```

```
logit_pred_acc <- 121 / (62 + 121)
logit_pred_acc #66% prediction accuracy in the testing data
```

```
## [1] 0.6612022
```

H) Logit `qlm` Model

```
logit1 <- glm(qlm, data = training,
              family = binomial) #estimate a logit model in the training data
```

```
## Warning: non-integer #successes in a binomial glm!
```

```
MSE_logit1 <- mean( (testing$average_precipitation - predict(logit1, newdata = testing))
 ^ 2)
MSE_logit1
```

```
## [1] 4.74836
```

```
logit_tbl1 <- table(testing$average_precipitation, predict(logit1, newdata = testing,
                                                 type = "response") > 0.11)
logit_tbl1
```

```
##
##         FALSE TRUE
##   0.09     2    2
##   0.1     29   13
##   0.11    16   37
##   0.12     9   27
##   0.13     6   23
##   0.14     5   11
##   0.15     0    3
```

```
table(predict(logit1, newdata = testing, type = "response") > 0.11)
```

```
##
## FALSE  TRUE
##    67   116
```

```
logit1_pred_acc <- 116 / (116 + 67)
logit1_pred_acc #63% prediction accuracy in the testing data
```

```
## [1] 0.6338798
```

Conclusion

Summary of each model's MSE in order from least to greatest:

```
matrix(cbind(c("Model",
               "Lasso qlm",
               "Partial Least Squares qlm",
               "Partial Least Squares ols",
               "Quadratic qlm",
               "Lasso ols",
               "Linear ols",
               "Logit ols",
               "Logit qlm",
               "MSE",
               MSE_lasso1[best_lasso1],
               MSE_PLS1[best_PLS1] ,
               MSE_PLS[best_PLS],
               MSE_qlm,
               MSE_lasso[best_lasso],
               MSE_ols,
               MSE_logit,
               MSE_logit1)),
             nrow = 9,
             ncol = 2)
```

```
##       [,1]                         [,2]
##  [1,] "Model"                      "MSE"
##  [2,] "Lasso qlm"                  "0.000165680692293469"
##  [3,] "Partial Least Squares qlm" "0.000167736632997151"
##  [4,] "Partial Least Squares ols" "0.000167804178988959"
##  [5,] "Quadratic qlm"              "0.000168099245477251"
##  [6,] "Lasso ols"                  "0.000170346916988519"
##  [7,] "Linear ols"                 "0.000171052773532335"
##  [8,] "Logit ols"                  "4.74027608693709"
##  [9,] "Logit qlm"                  "4.74836024930511"
```

Here we see that the model with the lowest MSE is the `lasso qlm` model, which is quadratic. This is expected since the fitted values versus residuals plot above did show (refer to section B under Models) a quadratic pattern. Hence, we would expect a low MSE associated with the `qlm` models since the data may fit the model well.

Summary of each model's prediction accuracy in the testing data in order from

greatest to least:

```
matrix(cbind(c("Model",
               "Lasso ols",
               "Linear ols",
               "Partial Least Squares ols",
               "Logit ols",
               "Lasso qlm",
               "Partial Least Squares qlm",
               "Logit qlm",
               "Quadratic qlm",
               "Prediction Accuracy",
               lasso_ols_pred_acc,
               ols_pred_acc,
               pls_ols_pred_acc,
               logit_pred_acc,
               lasso1_pred_acc,
               PLS1_pred_acc,
               logit1_pred_acc,
               qlm_pred_acc)),
             nrow = 9,
             ncol = 2)
```

```
##       [,1]                         [,2]
##  [1,] "Model"                      "Prediction Accuracy"
##  [2,] "Lasso ols"                  "0.699453551912568"
##  [3,] "Linear ols"                 "0.672131147540984"
##  [4,] "Partial Least Squares ols" "0.672131147540984"
##  [5,] "Logit ols"                  "0.66120218579235"
##  [6,] "Lasso qlm"                  "0.655737704918033"
##  [7,] "Partial Least Squares qlm" "0.633879781420765"
##  [8,] "Logit qlm"                  "0.633879781420765"
##  [9,] "Quadratic qlm"              "0.497854077253219"
```

We see that some of the models have the same prediction statistics. However, the `lasso ols` linear model predicts best in the testing data although the plot above (refer to section A under Models) suggested a quadratic rather than linear relationship among the variables. This model also suffers from a high MSE.
The high MSE supports the plot above of fitted values versus residuals for the `ols` model in that the data do not closely fit the linear line. Although the fit of this model is poor, it predicts best in the testing data compared to the other models. It is unclear whether this may be the best model to use out of all possibilities, however, it is the best prediction model among the models created for this project. Therefore, we can conclude that the `lasso ols` model has a high MSE but does achieve the goal to predict whether the overall average precipitation was above or below 0.11. In fact, we see that all of the linear `ols` models make better predictions in the testing data as opposed to the quadratic `qlm` models.

Bibliography

Data Source:
https://raw.githubusercontent.com/fivethirtyeight/data/master/us-weather-history/KCLT.csv
(https://raw.githubusercontent.com/fivethirtyeight/data/master/us-weather-history/KCLT.csv)

Data Documentation:
https://github.com/fivethirtyeight/data/tree/master/us-weather-history
(https://github.com/fivethirtyeight/data/tree/master/us-weather-history)