ProjectKNN

2023-11-27

#general background this dataset provides the cost the insurance has charged with the policy holder's info on sex, region, # of children, smoker, bmi, and age.

#motivation what motivated me to work on this project is that its interesting to see the charges that insurance will bill you based on different factors

#who cares? who cares about this project? people that have health issues and have to pay for insurance.

#what are we doing we are trying to fit both linear regression/ knn regression on the model to predict insurance costs.

#objectives we want to fit linear regression model see what are some features that affect insurance costs. fit knn regression model. compare the two and their performances this is a regression problem since we are trying to find out costs which is quantitative. Difference between scaled and unscaled knn regression.

#setup

```r
setwd("C:/Users/Kathy/Desktop/Stat/project")

insurance=read.csv("insurance.csv",stringsAsFactors = T)

#no missing values
sum(is.na(insurance))
```

```
## [1] 0
```

```r
insurance$children=as.factor(insurance$children)

summary(insurance)
```

```
##       age            sex           bmi        children smoker    
##  Min.   :18.00   female:662   Min.   :15.96   0:574    no :1064  
##  1st Qu.:27.00   male  :676   1st Qu.:26.30   1:324    yes: 274  
##  Median :39.00                Median :30.40   2:240              
##  Mean   :39.21                Mean   :30.66   3:157              
##  3rd Qu.:51.00                3rd Qu.:34.69   4: 25              
##  Max.   :64.00                Max.   :53.13   5: 18              
##        region       charges     
##  northeast:324   Min.   : 1122  
##  northwest:325   1st Qu.: 4740  
##  southeast:364   Median : 9382  
##  southwest:325   Mean   :13270  
##                  3rd Qu.:16640  
##                  Max.   :63770  
```

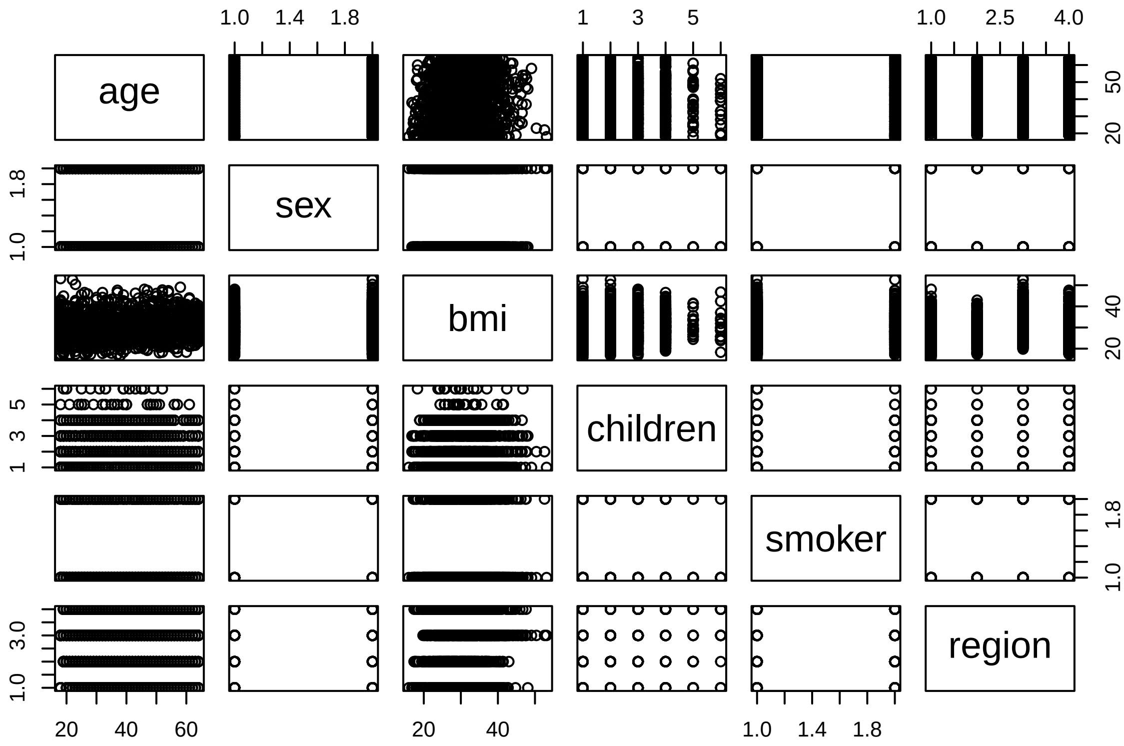
```r
attach(insurance)
```

#predictors

```r
#key predictors and significance
lm_model = lm(charges~.,data=insurance)
summary(lm_model)
```

```
##
## Call:
## lm(formula = charges ~ ., data = insurance)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -11689.4  -2902.6   -943.7   1492.2  30042.7
##
## Coefficients:
##                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)     -11927.17     993.66 -12.003  < 2e-16 ***
## age                257.19      11.91  21.587  < 2e-16 ***
## sexmale           -128.16     332.83  -0.385 0.700254
## bmi                336.91      28.61  11.775  < 2e-16 ***
## children1          390.98     421.35   0.928 0.353619
## children2         1635.78     466.67   3.505 0.000471 ***
## children3          964.34     548.10   1.759 0.078735 .
## children4         2947.37    1239.16   2.379 0.017524 *
## children5         1116.04    1456.02   0.767 0.443514
## smokeryes        23836.41     414.14  57.557  < 2e-16 ***
## regionnorthwest   -380.04     476.56  -0.797 0.425318
## regionsoutheast  -1033.14     479.14  -2.156 0.031245 *
## regionsouthwest   -952.89     478.15  -1.993 0.046483 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 6059 on 1325 degrees of freedom
## Multiple R-squared:  0.7519, Adjusted R-squared:  0.7497
## F-statistic: 334.7 on 12 and 1325 DF,  p-value: < 2.2e-16
```

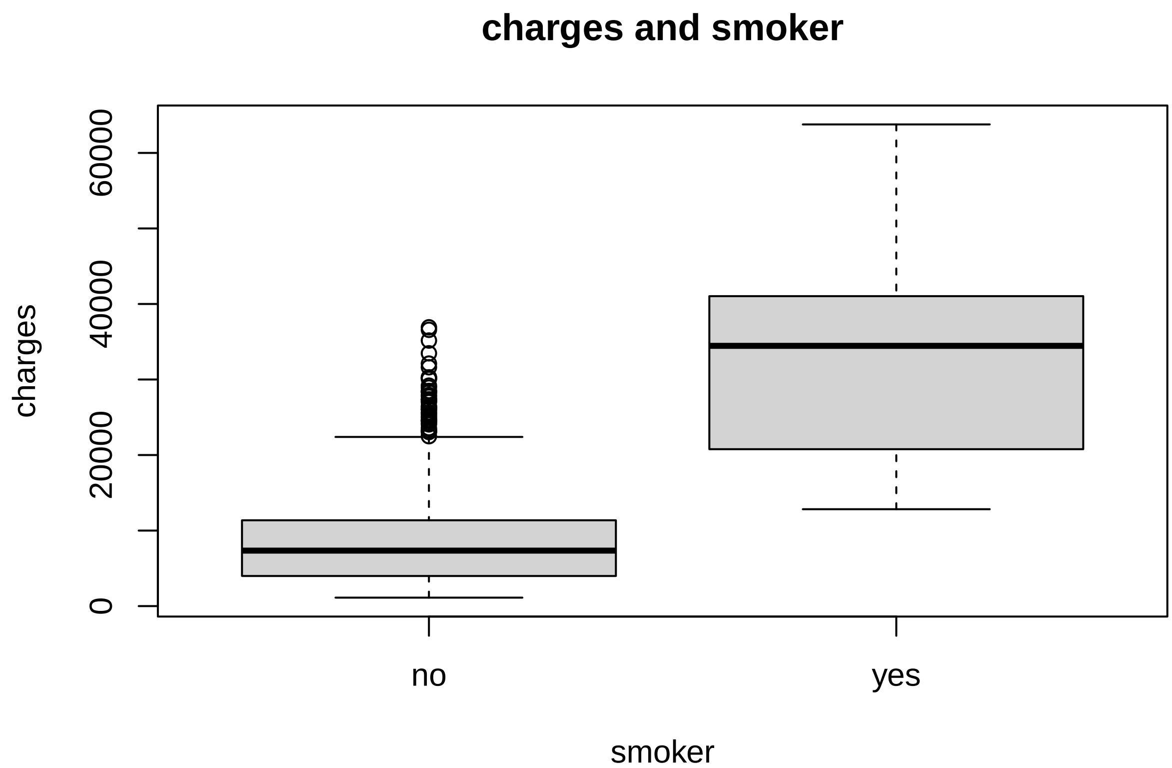
```r
#age, bmi, children, and if you are a smoker are very significant


#collinearity
pairs(insurance[,-7])
```



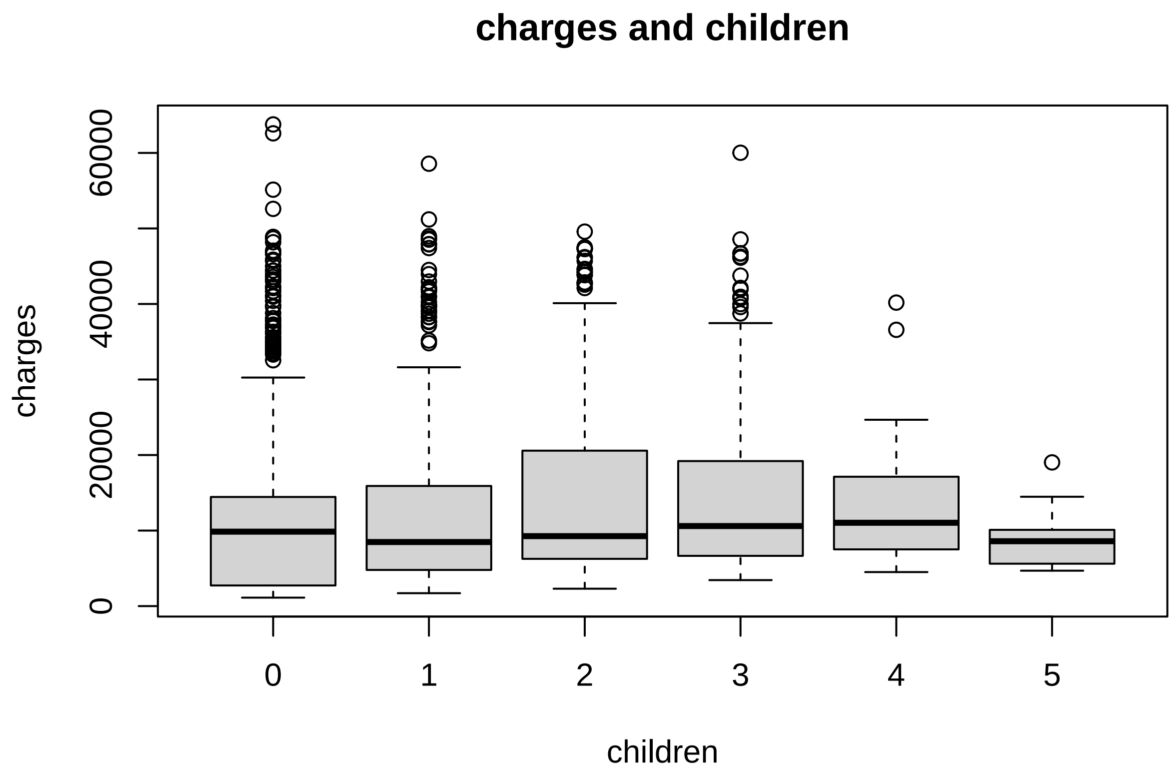
sex,smoker,region, children are categorical (qualitative) and the rest are quantitative our response is charges which is quantitative our predictors are sex, smoker, region, bmi, age, and children age, bmi, children2, and if you are a smoker are very significant

#visuals for characteristics of the dataset

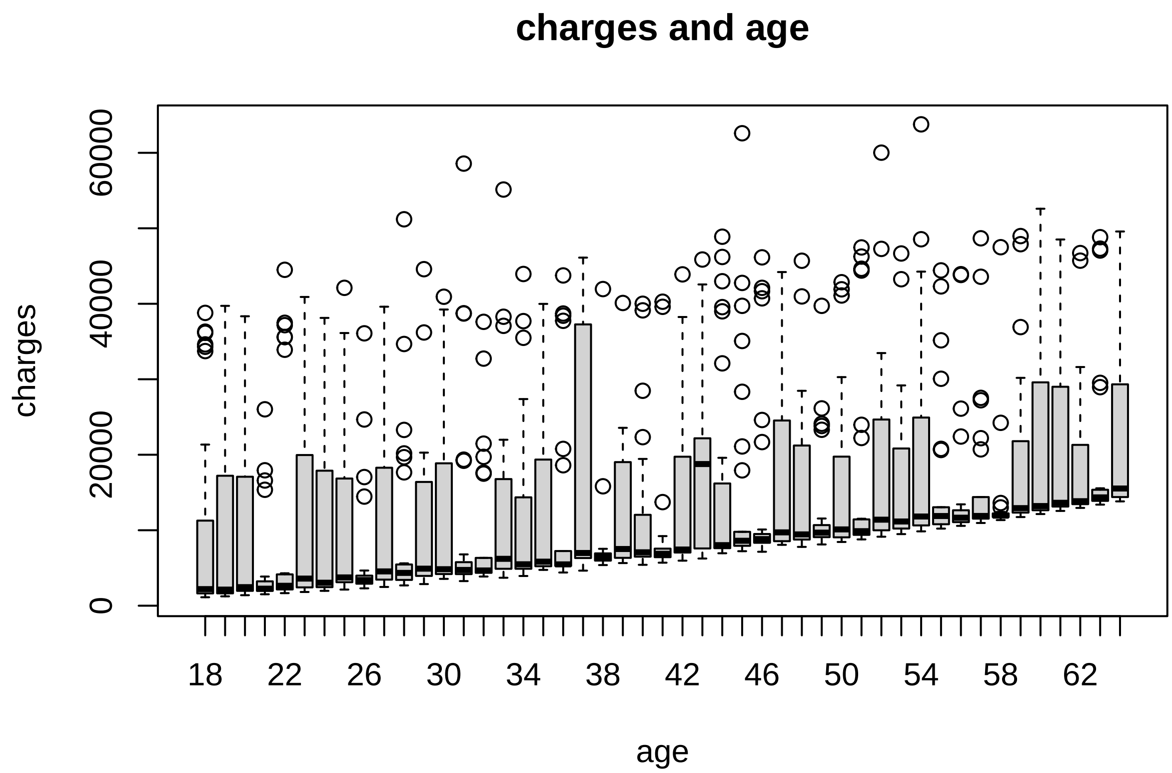
```r
#visuals and characteristics
boxplot(charges~smoker, data=insurance,main="charges and smoker")
```



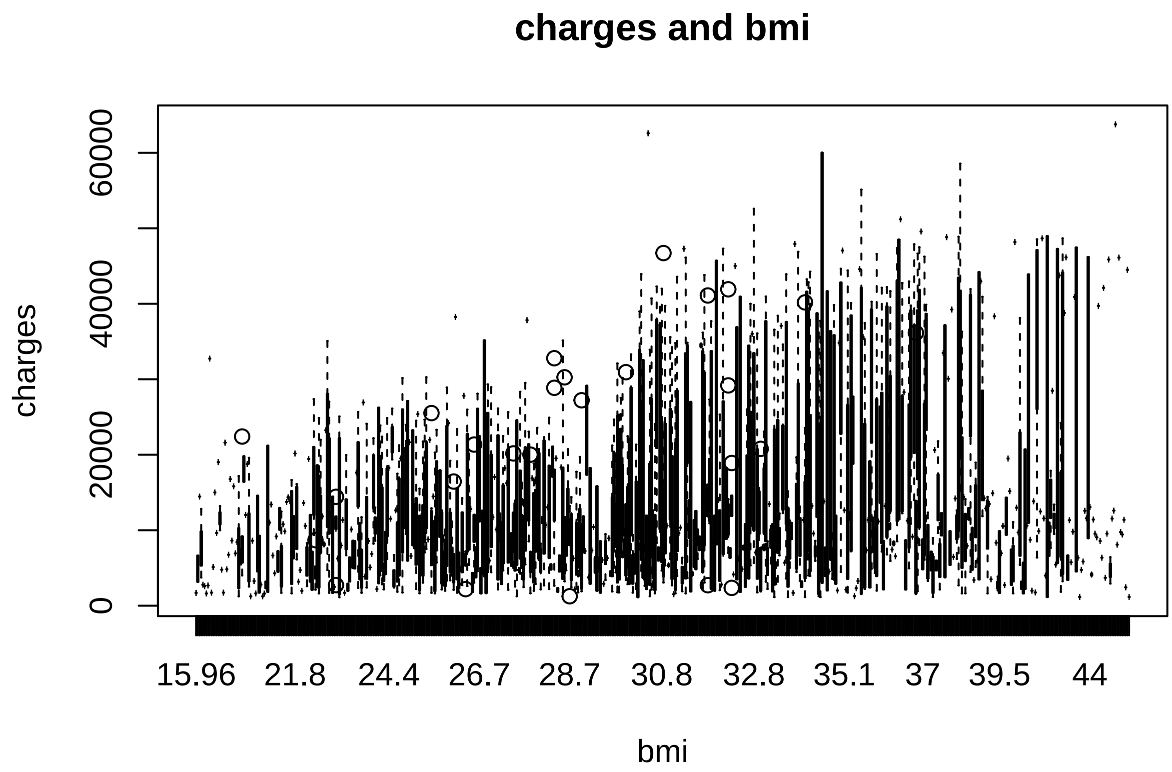

```
boxplot(charges~children, data=insurance,main="charges and children")
```



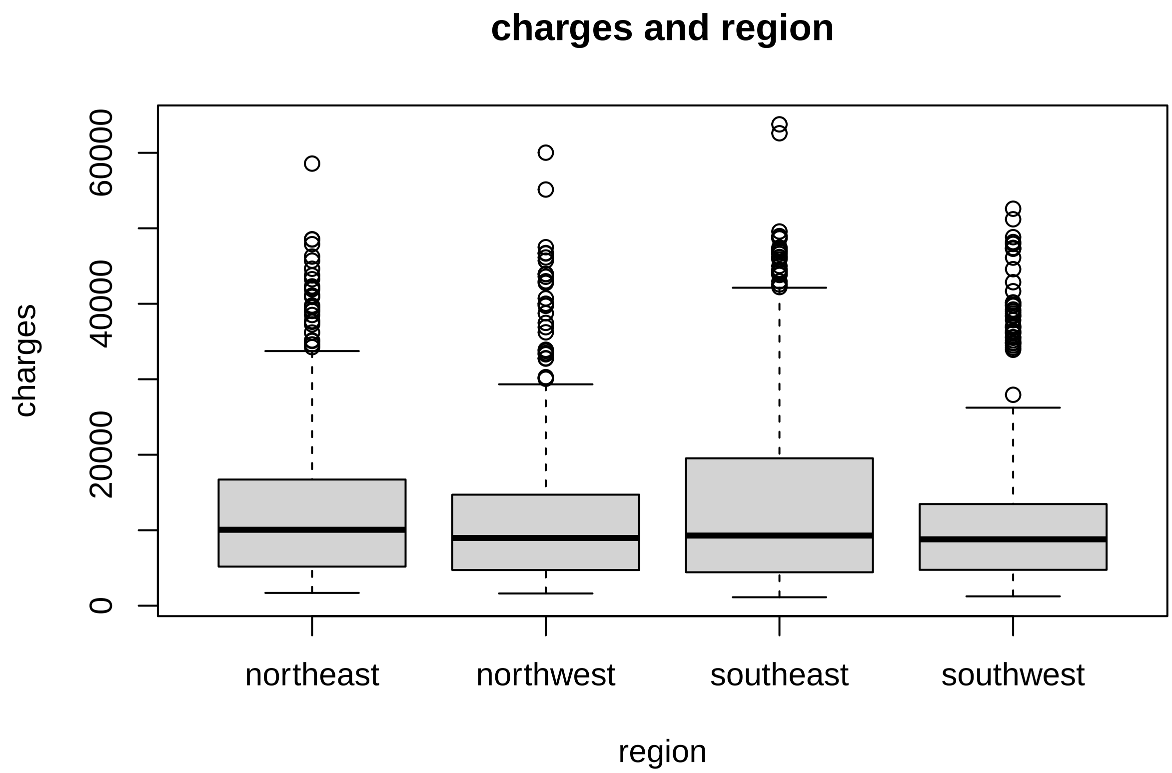

```r
boxplot(charges~age, data=insurance,main="charges and age")
```



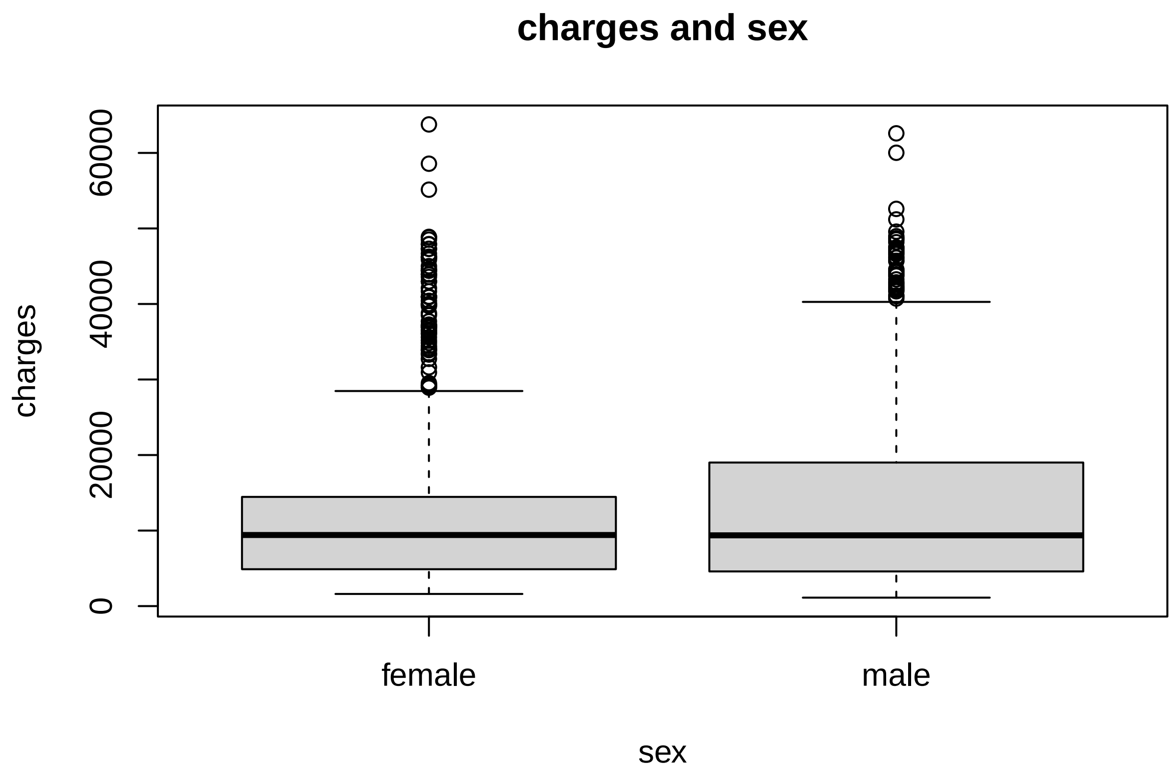

```r
boxplot(charges~bmi, data=insurance,main="charges and bmi")
```




```r
boxplot(charges~region,data=insurance,main="charges and region")
```




```
boxplot(charges~sex,data=insurance,main="charges and sex")
```



#unscaled knn model

```
#using all predictors, unscaled
set.seed(2002)
#split indices for training and testing
train_indices = sample(1:nrow(insurance),0.8*nrow(insurance))

training=insurance[train_indices,]
testing=insurance[-train_indices,]


library(class)
library(FNN)
```

```
##
## Attaching package: 'FNN'

## The following objects are masked from 'package:class':
##
##     knn, knn.cv
```

```
library(caret)
```

```
## Loading required package: ggplot2
```

```
## Loading required package: lattice
```

```r
set.seed(2002)

#convert predictors to numeric
training$sex = as.numeric(training$sex)
testing$sex = as.numeric(testing$sex)
training$smoker = as.numeric(training$smoker)
testing$smoker = as.numeric(testing$smoker)
training$region = as.numeric(training$region)
testing$region = as.numeric(testing$region)
training$children=as.numeric(training$children)
testing$children=as.numeric(testing$children)


predictors = setdiff(names(insurance),"charges")

#cross validation
ctrl = trainControl(method = "cv", number = 5)

#knn model which is the optimal k
knn_model2 = train(
  x = training[, -6],
  y = training$charges,
  method = "knn",
  tuneGrid = expand.grid(k = 1:50),
  trControl = ctrl
)
knn_model2
```

```
## k-Nearest Neighbors
##
## 1070 samples
##    6 predictor
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 855, 857, 856, 857, 855
## Resampling results across tuning parameters:
##
##   k   RMSE       Rsquared   MAE
##    1   96.49931  0.9999357   31.42289
##    2  205.81988  0.9995954   39.02040
##    3  275.62132  0.9992374   48.41356
##    4  334.70157  0.9988883   58.62306
##    5  372.24208  0.9986286   63.15506
##    6  396.23011  0.9984279   67.57008
##    7  415.49651  0.9982595   68.71161
##    8  434.54856  0.9981114   71.77898
##    9  449.17080  0.9979916   74.19276
##   10  463.40976  0.9978955   77.20615
##   11  479.00570  0.9977904   81.16261
##   12  493.34580  0.9976901   84.85716
##   13  507.80319  0.9975898   88.54639
```

```
##   14  521.13156  0.9975012   91.09277
##   15  532.89716  0.9974194   93.04631
##   16  543.63668  0.9973309   97.18199
##   17  554.50833  0.9972396  100.54309
##   18  566.35126  0.9971651  103.21954
##   19  575.44898  0.9970894  107.13285
##   20  588.74083  0.9970147  112.96651
##   21  598.28457  0.9969423  117.65589
##   22  611.77847  0.9968455  124.10261
##   23  622.48753  0.9967526  125.77370
##   24  631.87449  0.9966730  130.37564
##   25  645.65026  0.9965591  135.21941
##   26  658.58696  0.9964580  138.84414
##   27  669.68337  0.9963626  142.05555
##   28  680.78223  0.9962644  144.22907
##   29  694.96525  0.9961596  149.70187
##   30  701.55083  0.9961122  153.98355
##   31  715.62088  0.9960293  158.95504
##   32  729.75471  0.9959223  165.77264
##   33  742.00207  0.9958417  170.22403
##   34  754.42768  0.9957402  173.93181
##   35  767.21720  0.9956354  178.47191
##   36  781.40723  0.9955233  183.94499
##   37  793.00718  0.9954424  188.02900
##   38  805.67615  0.9953399  191.96741
##   39  814.49092  0.9952604  196.81758
##   40  826.29398  0.9951554  201.84253
##   41  838.09185  0.9950455  206.23539
##   42  851.23353  0.9949111  210.74336
##   43  864.32672  0.9947894  216.09864
##   44  875.51374  0.9946949  219.77500
##   45  887.98444  0.9945913  223.93443
##   46  902.29521  0.9944863  230.13821
##   47  915.94571  0.9943684  236.62079
##   48  928.55163  0.9942442  240.92562
##   49  942.47191  0.9941298  247.08051
##   50  954.38642  0.9939887  251.15248
##
## RMSE was used to select the optimal model using the smallest value.
## The final value used for the model was k = 1.
```

```r
#knn regression model
knn_model=knn.reg(train=training[,predictors],test=testing[,predictors],y=training$charges, k=1)
#predicting
predict=knn_model$pred

actual = testing$charges
#table to compare
compare= data.frame(Actual = actual,Predicted = predict)
compare
```

```
##        Actual Predicted
## 1    3866.855  4357.044
## 2    2721.321  3309.793
```

```
## 3   13228.847 13635.638
## 4    1137.011  1137.470
## 5    6203.902  7281.506
## 6   14451.835 14449.854
## 7    6313.759  5976.831
## 8   38709.176 37742.576
## 9    8059.679 15828.822
## 10  34303.167  2801.259
## 11   1743.214  1744.465
## 12  14235.072 13470.860
## 13   6389.378  6123.569
## 14  11741.726 11253.421
## 15   6571.024  7050.021
## 16   7935.291  7448.404
## 17  37165.164 37484.449
## 18  21098.554  8603.823
## 19  43578.939 11840.775
## 20  11073.176 35160.135
## 21  30184.937 46130.526
## 22  21344.847  2203.736
## 23  30942.192 46718.163
## 24   2331.519  1842.519
## 25  47055.532 14474.675
## 26  10825.254 10435.065
## 27   4646.759  4402.233
## 28  11488.317 11289.109
## 29  30259.996 12129.614
## 30   8601.329 26140.360
## 31   6686.431  6198.752
## 32  39556.495 32108.663
## 33  17081.080 16884.924
## 34   6082.405 40103.890
## 35   2457.211  2257.475
## 36  20745.989 18157.876
## 37  40720.551  9620.331
## 38   6334.344  6551.750
## 39  19964.746  6858.480
## 40   7077.189  6123.569
## 41  36950.257 36837.467
## 42  19749.383 11085.587
## 43   6128.797  5327.400
## 44  48824.450 13887.969
## 45   6455.863  6457.843
## 46  43753.337  5124.189
## 47   3981.977  3392.977
## 48   2137.654  1972.950
## 49  12044.342 11454.022
## 50   5649.715  6238.298
## 51   9644.253  9991.038
## 52   8871.152  8765.249
## 53  13012.209 12523.605
## 54   1980.070 23082.955
## 55  25081.768  2457.502
## 56  11987.168 12949.155
```

```
## 57  14001.287 14001.134
## 58   1727.785  1241.565
## 59   1615.767  1252.407
## 60  24476.479 10231.500
## 61   1832.094  1242.260
## 62   4260.744 18804.752
## 63  41097.162  9174.136
## 64  24869.837 11520.100
## 65  36219.405  2136.882
## 66   9282.481 23807.241
## 67   7265.703  7162.012
## 68   9617.662 10106.134
## 69   2523.169 17043.341
## 70   9855.131  9264.797
## 71   4237.127  4441.213
## 72   7742.110  8232.639
## 73   9432.925  8083.920
## 74  47896.791 12741.167
## 75   6746.743  6748.591
## 76   8835.265  9174.136
## 77  24671.663 18804.752
## 78  35491.640  4751.070
## 79   6600.206 39125.332
## 80  47928.030 13393.756
## 81   9144.565  8798.593
## 82  13822.803 13393.756
## 83  12142.579 11743.299
## 84  41919.097  9964.060
## 85  13352.100 13457.961
## 86   8334.458  8334.590
## 87   8932.084  8444.474
## 88  12404.879  4320.411
## 89  14133.038  1631.821
## 90   1607.510  1731.677
## 91  10043.249 10370.913
## 92   8116.269  7731.427
## 93   3481.868  4561.189
## 94   8302.536  8891.139
## 95   3176.816  3594.171
## 96   4618.080  4234.927
## 97   8522.003 18806.145
## 98  19594.810 19798.055
## 99   2134.901  1646.430
## 100  7345.727 18806.145
## 101 46889.261 13393.756
## 102  3167.456  2775.192
## 103  2254.797  2643.269
## 104 28287.898 12347.172
## 105 26109.329 11165.418
## 106 12731.000 12129.614
## 107  4762.329  6196.448
## 108  7512.267 15828.822
## 109  1632.036  1632.564
## 110 13224.693 12146.971
```

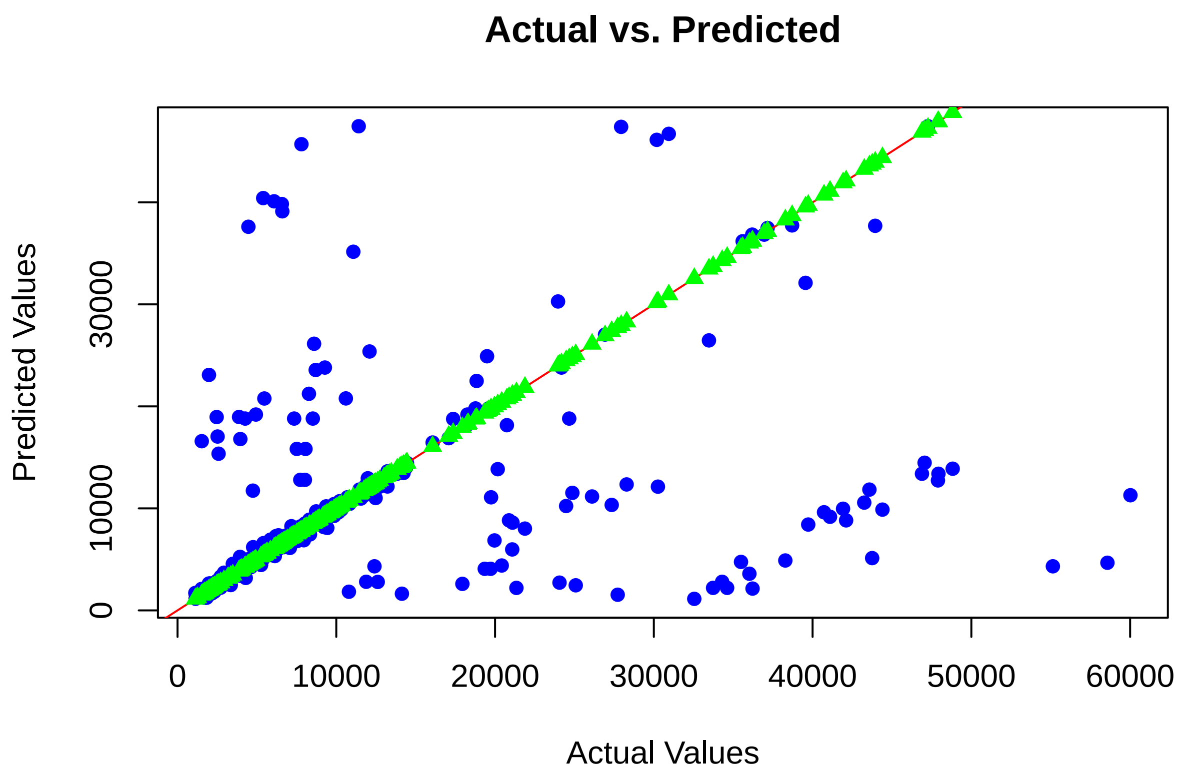
```
## 111  2201.097  1711.027
## 112  2203.472  2203.736
## 113 20878.784  8823.986
## 114 12475.351 11013.712
## 115 17942.106  2597.779
## 116  8027.968  8516.829
## 117 36197.699 36837.467
## 118 32548.340  1131.507
## 119 11455.280 11842.442
## 120 11763.001 11879.104
## 121  2498.414  2497.038
## 122  9361.327 10197.772
## 123 21082.160  5974.385
## 124 27724.289  1532.470
## 125  9866.305  9722.770
## 126  5397.617 40419.019
## 127 24059.680  2719.280
## 128  8342.909  7448.404
## 129 14043.477 13555.005
## 130  6067.127  6555.070
## 131 27346.042 10338.932
## 132  3213.622  3385.399
## 133  3935.180  5245.227
## 134  2494.022  2904.088
## 135 58571.074  4667.608
## 136  9724.530  9377.905
## 137  6356.271  6435.624
## 138  1242.816  1242.260
## 139 43943.876 37701.877
## 140 33471.972 26467.097
## 141  1633.044  1633.962
## 142  6571.544 39836.519
## 143 34617.841  2207.697
## 144  1977.815  2632.992
## 145  7173.360  8252.284
## 146  9391.346  9880.068
## 147 13143.865 13143.337
## 148 10141.136  9620.331
## 149  8280.623  7789.635
## 150  4058.712  4846.920
## 151 14394.398 14001.134
## 152  8703.456 23568.272
## 153  4837.582  5425.023
## 154  6185.321  6186.127
## 155  9863.472 10422.917
## 156  2020.552  2020.177
## 157  5375.038  5012.471
## 158 44400.406  9877.608
## 159  5469.007 20773.628
## 160  9566.991 10156.783
## 161  1263.249  1261.859
## 162  8604.484  8603.823
## 163 43254.418 10564.885
## 164  7985.815  7986.475
```

```
## 165 27941.288 47403.880
## 166 18259.216 19199.944
## 167  7209.492  6986.697
## 168 18310.742 18246.496
## 169 11848.141 12231.614
## 170  7731.858 12797.210
## 171  5584.306  5934.380
## 172 55135.402  4320.411
## 173 16069.085 16455.708
## 174  1526.312 16586.498
## 175 12323.936 11931.125
## 176 36021.011  3591.480
## 177  9872.701  9283.562
## 178 10601.632 20781.489
## 179 42111.665  8823.986
## 180  1875.344  1744.465
## 181  6600.361  6500.236
## 182  1141.445  1629.833
## 183  6849.026  7337.748
## 184  2585.851 15359.104
## 185 19719.695  4074.454
## 186  1682.597  2026.974
## 187 33732.687  2205.981
## 188 13462.520 13393.756
## 189  2927.065  3693.428
## 190 12233.828 12333.828
## 191  1121.874  1711.027
## 192  2217.469  2217.601
## 193  7160.094  7160.330
## 194  6358.776  7358.176
## 195  3875.734 18963.172
## 196 12609.887  2789.057
## 197  4746.344 11737.849
## 198 23967.383 30284.643
## 199  7518.025  7419.478
## 200 10702.642 11085.587
## 201  7804.160 45702.022
## 202  4889.037  5478.037
## 203  4518.826  4320.411
## 204  7144.863  6555.070
## 205  5484.467  5974.385
## 206  5267.818  5266.366
## 207 17361.766 18765.875
## 208  9957.722 10106.134
## 209 18767.738 19798.055
## 210 35595.590 36189.102
## 211 12094.478 25382.297
## 212 39725.518  8413.463
## 213  3161.454  3353.284
## 214 21880.820  8017.061
## 215  7325.048  6837.369
## 216  8023.135 12797.210
## 217  3353.470  2483.736
## 218  8277.523 21232.182
```

```
## 219  4462.722 37607.528
## 220  1981.582  1824.285
## 221 11554.224 10965.446
## 222 13204.286 13415.038
## 223 11884.049  2803.698
## 224  5855.903  6933.242
## 225  1674.632  1826.843
## 226 20420.605  4399.731
## 227 24180.933 23807.241
## 228  9222.403  8162.716
## 229 38282.749  4889.999
## 230 10214.636 10704.470
## 231  1728.897  1241.565
## 232  7623.518  7624.630
## 233  3176.288  3594.171
## 234  7954.517  6875.961
## 235  9630.397 10118.424
## 236  7727.253  7147.473
## 237  7153.554  6664.686
## 238  6112.353  5377.458
## 239  6496.886  6770.193
## 240 19350.369  4074.454
## 241 13844.797 13555.005
## 242 18838.704 22493.660
## 243  4934.705 19199.944
## 244  8733.229  9704.668
## 245  2055.325  2643.269
## 246  3956.071 16796.412
## 247  5415.661  6593.508
## 248  7537.164  6796.863
## 249 60021.399 11289.109
## 250 20167.336 13844.506
## 251 12224.351 12222.898
## 252 47269.854 47462.894
## 253  4296.271  3180.510
## 254  5615.369  5708.867
## 255  4415.159  5002.783
## 256 26926.514 27037.914
## 257  4747.053  5002.783
## 258  1515.345  2103.080
## 259  1708.926  1708.001
## 260  5261.469  4454.403
## 261  2710.829  2221.564
## 262  2464.619 18955.220
## 263  6940.910  7046.722
## 264 19496.719 24915.046
## 265  4239.893  4441.213
## 266 10325.206  9880.068
## 267 10795.937  1826.843
## 268 11411.685 47462.894
```

```r
plot(actual, predict,
     main = "Actual vs. Predicted",
     xlab = "Actual Values",
```

```
     ylab = "Predicted Values",
     col = "blue",   # Set the color of the points to blue
     pch = 16)       # Use solid circles for the points


abline(a = 0, b = 1, col = "red")


points(actual, actual, col = "green", pch = 17)  # Use solid triangles for the actual values
```



The RMSE value is very high in this model. Which is not good. The predictors are not scaled. Since KNN is distance based, its very important that the predictors are scaled.

#scaling with knn and cross validation for which neighbor is the best high rmse

```
#scaling with all predictors

library(FNN)
library(caret)
set.seed(2002)

#split into test and training
quantitative = c("bmi","age")
train_indices2=sample(1:nrow(insurance),0.8*nrow(insurance))
train=insurance[train_indices2,]
test=insurance[-train_indices2,]
```

```r
#these are all the quantitative going to be scaled
train_q=train[,quantitative]
testq=test[,quantitative]

#scaled quantitative
trains=scale(train_q)
tests=scale(testq)

#combine quantitative and qualitative
train_scaled=cbind(train[,setdiff(names(train),quantitative)],trains)
test_scaled=cbind(test[,setdiff(names(test),quantitative)],tests)

#convert predictors to numeric
train_scaled$sex = as.numeric(train_scaled$sex)
test_scaled$sex = as.numeric(test_scaled$sex)
train_scaled$smoker = as.numeric(train_scaled$smoker)
test_scaled$smoker = as.numeric(test_scaled$smoker)
train_scaled$region = as.numeric(train_scaled$region)
test_scaled$region = as.numeric(test_scaled$region)
train_scaled$children=as.numeric(train_scaled$children)
test_scaled$children=as.numeric(test_scaled$children)

#cross validation 5 k folds
ctrl = trainControl(method = "cv", number = 5)

#knn model which is the optimal k
knn_model3 = train(
  x = train_scaled[, -6],
  y = train_scaled$charges,
  method = "knn",
  tuneGrid = expand.grid(k = 1:50),
  trControl = ctrl
)
knn_model3
```

```
## k-Nearest Neighbors
##
## 1070 samples
##    6 predictor
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 857, 858, 856, 855, 854
## Resampling results across tuning parameters:
##
##   k   RMSE      Rsquared   MAE
##    1  272.1389  0.9988684   48.14505
##    2  291.3017  0.9987052   51.42426
##    3  317.9586  0.9985150   56.69777
##    4  344.5751  0.9983462   64.29354
##    5  361.5601  0.9982366   66.72713
##    6  373.2861  0.9981535   71.06845
##    7  388.0232  0.9980399   74.11764
```

```
##    8  396.7011  0.9979565   78.10870
##    9  405.5081  0.9978889   78.17822
##   10  415.6834  0.9978303   80.90294
##   11  426.0571  0.9977611   84.06959
##   12  437.6649  0.9976904   87.45964
##   13  447.2909  0.9976307   89.40025
##   14  458.5797  0.9975676   94.50522
##   15  466.5176  0.9975084   97.30723
##   16  473.0410  0.9974520   99.63316
##   17  481.8277  0.9973928  102.19715
##   18  491.6038  0.9973274  105.36273
##   19  496.9070  0.9972816  107.71343
##   20  505.1055  0.9972288  110.26463
##   21  519.8403  0.9971522  115.51792
##   22  530.7939  0.9970889  118.81918
##   23  541.4335  0.9970257  121.50847
##   24  551.3991  0.9969651  127.41901
##   25  562.7606  0.9968801  129.42060
##   26  575.2186  0.9968064  135.56504
##   27  588.9998  0.9967269  141.14502
##   28  599.0194  0.9966577  145.00154
##   29  612.0462  0.9965543  148.33477
##   30  623.5974  0.9964808  153.17171
##   31  635.5258  0.9963836  157.12433
##   32  648.0737  0.9962974  162.04010
##   33  655.6887  0.9962515  166.77330
##   34  667.5219  0.9961617  171.53977
##   35  682.4350  0.9960628  178.18801
##   36  695.5511  0.9959824  183.57869
##   37  708.3336  0.9958695  189.40982
##   38  720.0964  0.9957842  193.64968
##   39  732.7510  0.9956904  199.16061
##   40  740.6209  0.9956192  203.69686
##   41  752.2085  0.9955232  208.16253
##   42  765.5902  0.9953961  212.92541
##   43  775.7603  0.9953262  216.79344
##   44  790.8594  0.9952298  222.24914
##   45  805.2205  0.9951193  229.97671
##   46  820.6559  0.9949857  235.63563
##   47  835.2078  0.9948655  241.82631
##   48  849.4231  0.9947457  246.54854
##   49  860.1706  0.9946363  251.22481
##   50  875.6980  0.9945270  257.42816
## 
## RMSE was used to select the optimal model using the smallest value.
## The final value used for the model was k = 1.
```

```r
optimal_k = knn_model3$bestTune$k
cat('Optimal k:', optimal_k, '\n')
```

```
## Optimal k: 1
```

```r
#training with optimal k
final_knn_model = knn.reg(train_scaled[, -6], test_scaled[, -6], train_scaled$charges, k = optimal_k)


#predicting
predict2=final_knn_model$pred

actual2 = test_scaled$charges

#comparing actual and predictions
compare2= data.frame(Actual = actual2,Predicted = predict2)
compare2
```

```
##       Actual Predicted
## 1   3866.855  3861.210
## 2   2721.321  2719.280
## 3  13228.847 13224.057
## 4   1137.011  1137.470
## 5   6203.902  6198.752
## 6  14451.835 14449.854
## 7   6313.759  6311.952
## 8  38709.176 38711.000
## 9   8059.679  8062.764
## 10 34303.167 34254.053
## 11  1743.214  1744.465
## 12 14235.072 14254.608
## 13  6389.378  6393.603
## 14 11741.726 11743.299
## 15  6571.024  6555.070
## 16  7935.291  7986.475
## 17 37165.164 37133.898
## 18 21098.554 21195.818
## 19 43578.939 43813.866
## 20 11073.176 11070.535
## 21 30184.937 30166.618
## 22 21344.847 21348.706
## 23 30942.192 30284.643
## 24  2331.519  2322.622
## 25 47055.532 47291.055
## 26 10825.254 10807.486
## 27  4646.759  4661.286
## 28 11488.317 11482.635
## 29 30259.996 30284.643
## 30  8601.329  8603.823
## 31  6686.431  6666.243
## 32 39556.495 39597.407
## 33 17081.080 17085.268
## 34  6082.405  6079.672
## 35  2457.211  2459.720
## 36 20745.989 20773.628
## 37 40720.551 40904.200
## 38  6334.344  6338.076
## 39 19964.746 19933.458
```

```
## 40   7077.189  7050.642
## 41  36950.257 36910.608
## 42  19749.383 19798.055
## 43   6128.797  6123.569
## 44  48824.450 48885.136
## 45   6455.863  6457.843
## 46  43753.337 43813.866
## 47   3981.977  3987.926
## 48   2137.654  2138.071
## 49  12044.342 12032.326
## 50   5649.715  5662.225
## 51   9644.253  9634.538
## 52   8871.152  8891.139
## 53  13012.209 13019.161
## 54   1980.070  1984.453
## 55  25081.768 24915.221
## 56  11987.168 11946.626
## 57  14001.287 14001.134
## 58   1727.785  1727.540
## 59   1615.767  1621.340
## 60  24476.479 24513.091
## 61   1832.094  1837.237
## 62   4260.744  4266.166
## 63  41097.162 41034.221
## 64  24869.837 24873.385
## 65  36219.405 36189.102
## 66   9282.481  9283.562
## 67   7265.703  7261.741
## 68   9617.662  9620.331
## 69   2523.169  2527.819
## 70   9855.131  9850.432
## 71   4237.127  4234.927
## 72   7742.110  7740.337
## 73   9432.925  9447.250
## 74  47896.791 48173.361
## 75   6746.743  6748.591
## 76   8835.265  8827.210
## 77  24671.663 24667.419
## 78  35491.640 35585.576
## 79   6600.206  6593.508
## 80  47928.030 48173.361
## 81   9144.565  9140.951
## 82  13822.803 13831.115
## 83  12142.579 12146.971
## 84  41919.097 41949.244
## 85  13352.100 13390.559
## 86   8334.458  8334.590
## 87   8932.084  8930.935
## 88  12404.879 12430.953
## 89  14133.038 14119.620
## 90   1607.510  1621.340
## 91  10043.249 10065.413
## 92   8116.269  8116.680
## 93   3481.868  3484.331
```

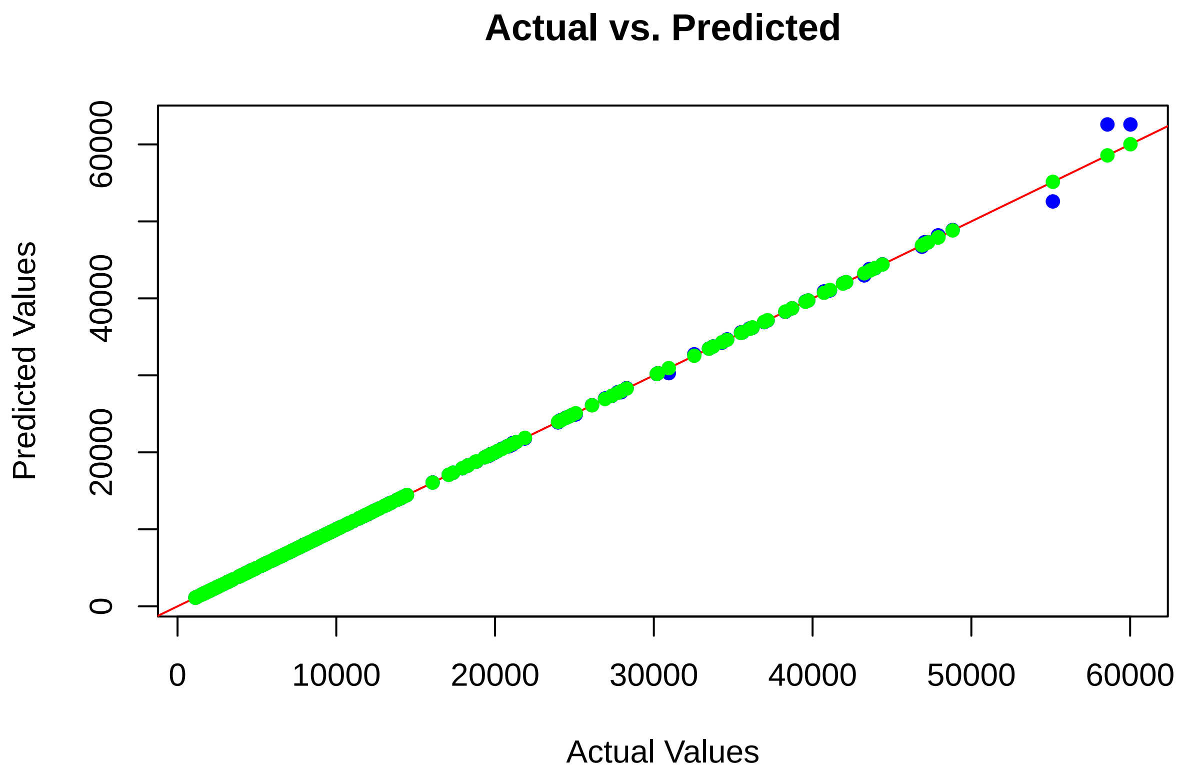
```
## 94   8302.536  8310.839
## 95   3176.816  3180.510
## 96   4618.080  4661.286
## 97   8522.003  8520.026
## 98  19594.810 19539.243
## 99   2134.901  2136.882
## 100  7345.727  7345.084
## 101 46889.261 46718.163
## 102  3167.456  3171.615
## 103  2254.797  2257.475
## 104 28287.898 28340.189
## 105 26109.329 26125.675
## 106 12731.000 12741.167
## 107  4762.329  4766.022
## 108  7512.267  7526.706
## 109  1632.036  1632.564
## 110 13224.693 13224.057
## 111  2201.097  2200.831
## 112  2203.472  2203.736
## 113 20878.784 20781.489
## 114 12475.351 12479.709
## 115 17942.106 17929.303
## 116  8027.968  8026.667
## 117 36197.699 36189.102
## 118 32548.340 32734.186
## 119 11455.280 11454.022
## 120 11763.001 11743.934
## 121  2498.414  2497.038
## 122  9361.327  9377.905
## 123 21082.160 20984.094
## 124 27724.289 27808.725
## 125  9866.305  9869.810
## 126  5397.617  5400.980
## 127 24059.680 24106.913
## 128  8342.909  8347.164
## 129 14043.477 14007.222
## 130  6067.127  6059.173
## 131 27346.042 27322.734
## 132  3213.622  3208.787
## 133  3935.180  3943.595
## 134  2494.022  2497.038
## 135 58571.074 62592.873
## 136  9724.530  9722.770
## 137  6356.271  6360.994
## 138  1242.816  1242.260
## 139 43943.876 43921.184
## 140 33471.972 33475.817
## 141  1633.044  1633.962
## 142  6571.544  6555.070
## 143 34617.841 34672.147
## 144  1977.815  1972.950
## 145  7173.360  7162.012
## 146  9391.346  9386.161
## 147 13143.865 13143.337
```

```
## 148 10141.136 10156.783
## 149  8280.623  8283.681
## 150  4058.712  4058.116
## 151 14394.398 14394.558
## 152  8703.456  8688.859
## 153  4837.582  4830.630
## 154  6185.321  6186.127
## 155  9863.472  9861.025
## 156  2020.552  2020.177
## 157  5375.038  5373.364
## 158 44400.406 44423.803
## 159  5469.007  5472.449
## 160  9566.991  9563.029
## 161  1263.249  1261.859
## 162  8604.484  8603.823
## 163 43254.418 42983.459
## 164  7985.815  7986.475
## 165 27941.288 27808.725
## 166 18259.216 18246.496
## 167  7209.492  7201.701
## 168 18310.742 18328.238
## 169 11848.141 11842.442
## 170  7731.858  7731.427
## 171  5584.306  5594.846
## 172 55135.402 52590.829
## 173 16069.085 16085.128
## 174  1526.312  1532.470
## 175 12323.936 12333.828
## 176 36021.011 36085.219
## 177  9872.701  9869.810
## 178 10601.632 10601.412
## 179 42111.665 42112.236
## 180  1875.344  1877.929
## 181  6600.361  6593.508
## 182  1141.445  1137.470
## 183  6849.026  6858.480
## 184  2585.851  2585.269
## 185 19719.695 19673.336
## 186  1682.597  1694.796
## 187 33732.687 33750.292
## 188 13462.520 13457.961
## 189  2927.065  2913.569
## 190 12233.828 12235.839
## 191  1121.874  1131.507
## 192  2217.469  2217.601
## 193  7160.094  7160.330
## 194  6358.776  6360.994
## 195  3875.734  3877.304
## 196 12609.887 12622.180
## 197  4746.344  4751.070
## 198 23967.383 23887.663
## 199  7518.025  7526.706
## 200 10702.642 10704.470
## 201  7804.160  7789.635
```

```
## 202  4889.037  4889.999
## 203  4518.826  4527.183
## 204  7144.863  7147.473
## 205  5484.467  5488.262
## 206  5267.818  5266.366
## 207 17361.766 17352.680
## 208  9957.722  9964.060
## 209 18767.738 18765.875
## 210 35595.590 35585.576
## 211 12094.478 12096.651
## 212 39725.518 39727.614
## 213  3161.454  3171.615
## 214 21880.820 21797.000
## 215  7325.048  7323.735
## 216  8023.135  8026.667
## 217  3353.470  3353.284
## 218  8277.523  8283.681
## 219  4462.722  4463.205
## 220  1981.582  1984.453
## 221 11554.224 11552.904
## 222 13204.286 13217.094
## 223 11884.049 11881.970
## 224  5855.903  5846.918
## 225  1674.632  1665.000
## 226 20420.605 20462.998
## 227 24180.933 24227.337
## 228  9222.403  9225.256
## 229 38282.749 38245.593
## 230 10214.636 10226.284
## 231  1728.897  1727.540
## 232  7623.518  7624.630
## 233  3176.288  3172.018
## 234  7954.517  7986.475
## 235  9630.397  9634.538
## 236  7727.253  7726.854
## 237  7153.554  7152.671
## 238  6112.353  6113.231
## 239  6496.886  6500.236
## 240 19350.369 19361.999
## 241 13844.797 13844.506
## 242 18838.704 18806.145
## 243  4934.705  4931.647
## 244  8733.229  8765.249
## 245  2055.325  2045.685
## 246  3956.071  3947.413
## 247  5415.661  5425.023
## 248  7537.164  7526.706
## 249 60021.399 62592.873
## 250 20167.336 20177.671
## 251 12224.351 12222.898
## 252 47269.854 47291.055
## 253  4296.271  4320.411
## 254  5615.369  5630.458
## 255  4415.159  4402.233
```

```
## 256 26926.514 27000.985
## 257  4747.053  4751.070
## 258  1515.345  1532.470
## 259  1708.926  1708.001
## 260  5261.469  5257.508
## 261  2710.829  2709.244
## 262  2464.619  2459.720
## 263  6940.910  6933.242
## 264 19496.719 19515.542
## 265  4239.893  4243.590
## 266 10325.206 10338.932
## 267 10795.937 10796.350
## 268 11411.685 11396.900
```

```r
#plotting
plot(actual2, predict2,
     main = "Actual vs. Predicted",
     xlab = "Actual Values",
     ylab = "Predicted Values",
     col = "blue",  # Set the color of the points to blue
     pch = 16)      # Use solid circles for the points


abline(a = 0, b = 1, col = "red")


points(actual2, actual2, col = "green", pch = 16)  # Use solid triangles for the actual values
```



scaling does not improve the model. outliers could be a problem. this is interesting to see that the values are close together but the rmse is very high. higher than when not scaled.

#linear no scaling no cross validation high rmse

```
linear_model = lm(charges ~ ., data = training)
linear_predictions = predict(linear_model, newdata = testing)
actual3=testing$charges
compare_linear = data.frame(Actual = actual3, Predicted = linear_predictions)
compare_linear
```

```
##         Actual  Predicted
## 5     3866.855  5477.5435
## 11    2721.321  3018.1923
## 21   13228.847 15129.8337
## 23    1137.011  3183.2199
## 25    6203.902  7667.6698
## 27   14451.835 11853.0265
## 44    6313.759  8070.9269
## 50   38709.176 32369.4937
## 55    8059.679  9000.4888
## 58   34303.167 27062.1465
## 66    1743.214  1338.3019
## 67   14235.072 16808.6920
## 68    6389.378  7355.3519
## 73   11741.726 11792.1495
## 80    6571.024  8908.8617
```

```
## 82    7935.291 11876.5985
## 83   37165.164 29425.1512
## 86   21098.554 31675.2407
## 87   43578.939 36082.3779
## 88   11073.176 10599.4163
## 93   30184.937 38367.4763
## 103  21344.847  2153.1906
## 104  30942.192 38221.9639
## 107   2331.519  1748.5747
## 110  47055.532 38916.0171
## 111  10825.254 13367.2304
## 113   4646.759  6941.8898
## 115  11488.317 13756.5724
## 116  30259.996 13033.7985
## 119   8601.329  8968.7881
## 120   6686.431  5961.9772
## 124  39556.495 33739.1127
## 127  17081.080 24655.0682
## 130   6082.405  9571.2223
## 135   2457.211  2263.6588
## 145  20745.989 30097.4202
## 147  40720.551 34902.6327
## 153   6334.344  9778.7385
## 154  19964.746 29888.3189
## 155   7077.189  7079.3258
## 159  36950.257 30316.9845
## 160  19749.383 11145.0089
## 168   6128.797  8289.2024
## 176  48824.450 39267.2821
## 177   6455.863  7872.5782
## 186  43753.337 36079.3071
## 187   3981.977  4881.0761
## 193   2137.654  2402.3987
## 194  12044.342 11442.7224
## 197   5649.715  7862.7695
## 199   9644.253  6856.8692
## 202   8871.152 10866.2601
## 203  13012.209 11125.7689
## 211   1980.070  3701.7147
## 220  25081.768  1101.5236
## 226  11987.168 14614.1620
## 232  14001.287 13528.0121
## 233   1727.785 -2159.3507
## 237   1615.767   622.0204
## 246  24476.479 11661.8808
## 249   1832.094  -375.8199
## 254   4260.744  5840.0307
## 255  41097.162 34909.6311
## 263  24869.837 34777.8072
## 264  36219.405 28085.7199
## 270   9282.481  9820.9527
## 273   7265.703 11570.5945
## 274   9617.662 10594.6235
## 275   2523.169  3205.0085
```

```
## 280   9855.131  8298.4365
## 283   4237.127  5358.9306
## 286   7742.110  8807.8555
## 287   9432.925 16360.3171
## 289  47896.791 39178.1878
## 301   6746.743  8053.3140
## 317   8835.265 11291.2702
## 322  24671.663  6393.0402
## 323  35491.640 29653.3840
## 326   6600.206 10042.2827
## 329  47928.030 38870.9799
## 330   9144.565 12772.6744
## 336  13822.803 15256.7836
## 337  12142.579 11669.6333
## 339  41919.097 35627.1302
## 342  13352.100 13780.1059
## 348   8334.458 11391.4726
## 352   8932.084  8506.5813
## 354  12404.879  7980.2295
## 355  14133.038  4261.4697
## 360   1607.510 -1249.6965
## 361  10043.249 11920.6858
## 369   8116.269 10778.3811
## 370   3481.868  4185.3535
## 385   8302.536  7897.4270
## 389   3176.816  1674.2895
## 390   4618.080  5242.9965
## 397   8522.003 11129.5223
## 412  19594.810 29997.8493
## 415   2134.901  3772.2447
## 416   7345.727 10616.8736
## 421  46889.261 38799.5196
## 429   3167.456  -675.5314
## 437   2254.797  3960.0800
## 444  28287.898 15130.6150
## 445  26109.329 35217.3701
## 447  12731.000 13363.0820
## 450   4762.329  9437.9682
## 451   7512.267  9364.6203
## 465   1632.036   867.9681
## 467  13224.693 12699.0093
## 470   2201.097   357.9709
## 472   2203.472  2153.1906
## 474  20878.784 10849.5418
## 487  12475.351 10432.3329
## 495  17942.106 26875.5375
## 498   8027.968  9534.0462
## 501  36197.699 29463.8671
## 504  32548.340 25740.6768
## 510  11455.280 11336.8502
## 511  11763.001 13650.1007
## 512   2498.414  5427.5774
## 513   9361.327  8705.0291
## 515  21082.160 30757.3397
```

```
## 517  27724.289  4661.8765
## 523   9866.305 12088.2635
## 524   5397.617  9383.7252
## 527  24059.680  3471.0206
## 529   8342.909 13307.3039
## 532  14043.477 14312.3211
## 536   6067.127  7596.8951
## 540  27346.042 11583.9189
## 549   3213.622  3527.6797
## 558   3935.180  7454.3314
## 563   2494.022  4199.5773
## 578  58571.074 32183.4882
## 579   9724.530 11292.3242
## 583   6356.271 13449.3365
## 585   1242.816 -1006.6653
## 588  43943.876 30260.9569
## 600  33471.972 14393.9441
## 601   1633.044  4538.7611
## 612   6571.544  9363.8485
## 624  34617.841 26975.5585
## 632   1977.815  3027.1183
## 634   7173.360  7017.9237
## 635   9391.346 14021.0325
## 643  13143.865 14974.9290
## 646  10141.136 12248.4369
## 653   8280.623  9951.8212
## 658   4058.712  6636.3712
## 660  14394.398 14331.7484
## 667   8703.456 10025.3340
## 671   4837.582  7266.8188
## 674   6185.321  8063.7128
## 684   9863.472  9601.0108
## 701   2020.552  3981.3000
## 706   5375.038  7441.2606
## 707  44400.406 36435.6537
## 710   5469.007  6170.8253
## 717   9566.991  8332.0112
## 724   1263.249  3625.3611
## 741   8604.484  8760.9024
## 743  43254.418 36422.4023
## 757   7985.815  7112.0209
## 771  27941.288 16670.0894
## 781  18259.216 28281.0810
## 795   7209.492  9185.6713
## 796  18310.742 27539.7420
## 799  11848.141 12988.0852
## 806   7731.858 10866.0202
## 815   5584.306  9188.8599
## 820  55135.402 31104.7065
## 826  16069.085 16007.4616
## 841   1526.312  2799.9709
## 842  12323.936 11541.6909
## 843  36021.011 28493.7672
## 847   9872.701 12049.0772
```

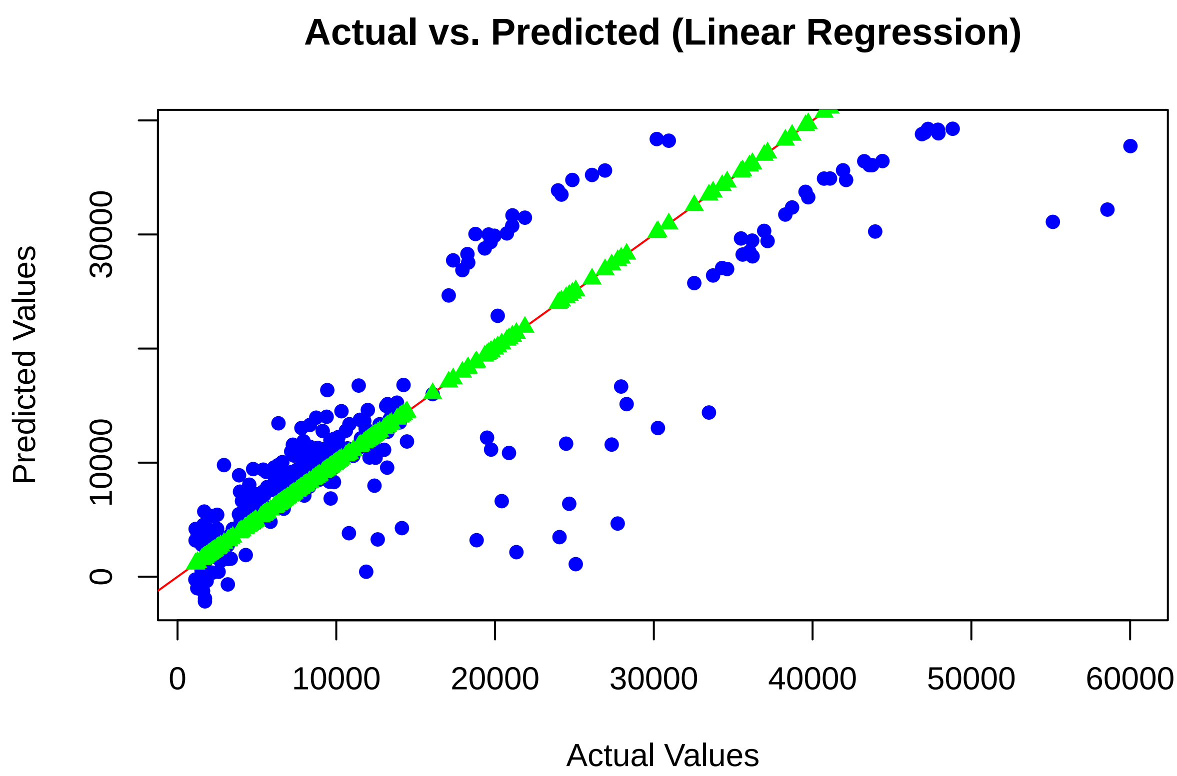
```
## 850   10601.632 12794.7700
## 853   42111.665 34779.0941
## 856    1875.344  1823.6528
## 866    6600.361  8588.2802
## 867    1141.445  4188.4011
## 874    6849.026  8877.8106
## 883    2585.851   432.9964
## 886   19719.695 29334.6801
## 889    1682.597  5711.6212
## 912   33732.687 26406.7961
## 922   13462.520 14078.7081
## 931    2927.065  9786.0103
## 936   12233.828 11488.2818
## 941    1121.874  -248.2609
## 943    2217.469  5326.2863
## 947    7160.094 10976.9491
## 949    6358.776  8873.6965
## 956    3875.734  8891.5700
## 958   12609.887  3268.5762
## 966    4746.344  5814.2741
## 967   23967.383 33862.6068
## 968    7518.025  8115.1158
## 971   10702.642 11248.9932
## 977    7804.160 13032.9420
## 980    4889.037  6393.2080
## 989    4518.826  8065.3984
## 992    7144.863  7865.9659
## 994    5484.467  7209.5518
## 1000   5267.818  5669.1397
## 1001 17361.766 27733.5036
## 1010  9957.722 10949.2062
## 1012 18767.738 30042.1505
## 1022 35595.590 28242.2310
## 1036 12094.478 10449.8980
## 1038 39725.518 33263.3210
## 1044  3161.454  2744.4832
## 1046 21880.820 31475.5627
## 1047  7325.048  7186.0961
## 1051  8023.135 11528.2851
## 1055  3353.470  1579.8492
## 1057  8277.523  9016.8677
## 1060  4462.722  7601.9812
## 1061  1981.582  4113.3216
## 1062 11554.224 12136.3520
## 1075 13204.286  9561.9489
## 1081 11884.049   436.7900
## 1083  5855.903  4820.1597
## 1098  1674.632  4138.3481
## 1105 20420.605  6626.7860
## 1113 24180.933 33497.3756
## 1117  9222.403 11171.4210
## 1119 38282.749 31748.4664
## 1127 10214.636 11424.3027
## 1130  1728.897 -1907.2676
```

```
## 1137  7623.518  7296.6750
## 1138  3176.288  1554.5500
## 1142  7954.517 10032.7792
## 1145  9630.397 11992.3107
## 1161  7727.253 10515.1354
## 1165  7153.554  8009.8780
## 1170  6112.353  8776.7922
## 1178  6496.886  6993.8116
## 1180 19350.369 28779.3686
## 1188 13844.797 14379.3978
## 1196 18838.704  3200.6691
## 1200  4934.705  4674.4670
## 1202  8733.229 13942.2054
## 1203  2055.325  3847.5476
## 1215  3956.071  5230.9650
## 1217  5415.661  6166.1021
## 1220  7537.164  8949.8903
## 1231 60021.399 37748.5659
## 1232 20167.336 22871.6714
## 1238 12224.351 11913.3467
## 1241 47269.854 39253.4539
## 1243  4296.271  1899.5630
## 1246  5615.369  5349.8352
## 1255  4415.159  5170.4233
## 1266 26926.514 35610.6688
## 1274  4747.053  6207.2489
## 1293  1515.345   546.0734
## 1297  1708.926  1134.8105
## 1299  5261.469  6428.9481
## 1300  2710.829  1376.5178
## 1306  2464.619  2522.6419
## 1311  6940.910  7884.9082
## 1319 19496.719 12188.0232
## 1325  4239.893  4852.6093
## 1330 10325.206 14507.0211
## 1332 10795.937  3815.3817
## 1333 11411.685 16758.0951
```

```r
mse_linear = mean((linear_predictions - actual3)^2)
cat('MSE for Linear Regression:', mse_linear, '\n')
```

```
## MSE for Linear Regression: 41385673
```

```r
plot(actual3, linear_predictions,
     main = "Actual vs. Predicted (Linear Regression)",
     xlab = "Actual Values",
     ylab = "Predicted Values",
     col = "blue",  # Set the color of the points to blue
     pch = 16)      # Use solid circles for the points

abline(a = 0, b = 1, col = "red")

points(actual3, actual3, col = "green", pch = 17)
```



linear regression does not require scaling.

#linear no scaling cross validation better rmse

```r
library(boot)
```

```
##
## Attaching package: 'boot'

## The following object is masked from 'package:lattice':
##
##     melanoma
```

```r
library(caret)

# Cross-validation to find the optimal linear model
ctrl <- trainControl(method = "cv", number = 5)
linear_model_cv = train(
  x = training[, -6],  # Exclude the target variable
  y = training$charges,
  method = "lm",
  trControl = ctrl
)


optimal_linear_model <- linear_model_cv$finalModel
```

```r
predictions = predict(optimal_linear_model, newdata = testing)

mse_test <- mean((predictions - testing$charges)^2)
cat('MSE on Test Set:', mse_test, '\n')
```

```
## MSE on Test Set: 3.506902e-22
```

```r
actual4=testing$charges
compare_linear = data.frame(Actual = actual4, Predicted = predictions)
compare_linear
```

```
##         Actual Predicted
## 5     3866.855  3866.855
## 11    2721.321  2721.321
## 21   13228.847 13228.847
## 23    1137.011  1137.011
## 25    6203.902  6203.902
## 27   14451.835 14451.835
## 44    6313.759  6313.759
## 50   38709.176 38709.176
## 55    8059.679  8059.679
## 58   34303.167 34303.167
## 66    1743.214  1743.214
## 67   14235.072 14235.072
## 68    6389.378  6389.378
## 73   11741.726 11741.726
## 80    6571.024  6571.024
## 82    7935.291  7935.291
## 83   37165.164 37165.164
## 86   21098.554 21098.554
## 87   43578.939 43578.939
## 88   11073.176 11073.176
## 93   30184.937 30184.937
## 103  21344.847 21344.847
## 104  30942.192 30942.192
## 107   2331.519  2331.519
## 110  47055.532 47055.532
## 111  10825.254 10825.254
## 113   4646.759  4646.759
## 115  11488.317 11488.317
## 116  30259.996 30259.996
## 119   8601.329  8601.329
## 120   6686.431  6686.431
## 124  39556.495 39556.495
## 127  17081.080 17081.080
## 130   6082.405  6082.405
## 135   2457.211  2457.211
## 145  20745.989 20745.989
## 147  40720.551 40720.551
## 153   6334.344  6334.344
## 154  19964.746 19964.746
## 155   7077.189  7077.189
```

```
## 159  36950.257 36950.257
## 160  19749.383 19749.383
## 168   6128.797  6128.797
## 176  48824.450 48824.450
## 177   6455.863  6455.863
## 186  43753.337 43753.337
## 187   3981.977  3981.977
## 193   2137.654  2137.654
## 194  12044.342 12044.342
## 197   5649.715  5649.715
## 199   9644.253  9644.253
## 202   8871.152  8871.152
## 203  13012.209 13012.209
## 211   1980.070  1980.070
## 220  25081.768 25081.768
## 226  11987.168 11987.168
## 232  14001.287 14001.287
## 233   1727.785  1727.785
## 237   1615.767  1615.767
## 246  24476.479 24476.479
## 249   1832.094  1832.094
## 254   4260.744  4260.744
## 255  41097.162 41097.162
## 263  24869.837 24869.837
## 264  36219.405 36219.405
## 270   9282.481  9282.481
## 273   7265.703  7265.702
## 274   9617.662  9617.662
## 275   2523.169  2523.169
## 280   9855.131  9855.131
## 283   4237.127  4237.127
## 286   7742.110  7742.110
## 287   9432.925  9432.925
## 289  47896.791 47896.791
## 301   6746.743  6746.742
## 317   8835.265  8835.265
## 322  24671.663 24671.663
## 323  35491.640 35491.640
## 326   6600.206  6600.206
## 329  47928.030 47928.030
## 330   9144.565  9144.565
## 336  13822.803 13822.803
## 337  12142.579 12142.579
## 339  41919.097 41919.097
## 342  13352.100 13352.100
## 348   8334.458  8334.458
## 352   8932.084  8932.084
## 354  12404.879 12404.879
## 355  14133.038 14133.038
## 360   1607.510  1607.510
## 361  10043.249 10043.249
## 369   8116.269  8116.269
## 370   3481.868  3481.868
## 385   8302.536  8302.536
```

```
## 389   3176.816  3176.816
## 390   4618.080  4618.080
## 397   8522.003  8522.003
## 412  19594.810 19594.810
## 415   2134.901  2134.901
## 416   7345.727  7345.727
## 421  46889.261 46889.261
## 429   3167.456  3167.456
## 437   2254.797  2254.797
## 444  28287.898 28287.898
## 445  26109.329 26109.329
## 447  12731.000 12731.000
## 450   4762.329  4762.329
## 451   7512.267  7512.267
## 465   1632.036  1632.036
## 467  13224.693 13224.693
## 470   2201.097  2201.097
## 472   2203.472  2203.472
## 474  20878.784 20878.784
## 487  12475.351 12475.351
## 495  17942.106 17942.106
## 498   8027.968  8027.968
## 501  36197.699 36197.699
## 504  32548.340 32548.341
## 510  11455.280 11455.280
## 511  11763.001 11763.001
## 512   2498.414  2498.414
## 513   9361.327  9361.327
## 515  21082.160 21082.160
## 517  27724.289 27724.289
## 523   9866.305  9866.305
## 524   5397.617  5397.617
## 527  24059.680 24059.680
## 529   8342.909  8342.909
## 532  14043.477 14043.477
## 536   6067.127  6067.127
## 540  27346.042 27346.042
## 549   3213.622  3213.622
## 558   3935.180  3935.180
## 563   2494.022  2494.022
## 578  58571.074 58571.074
## 579   9724.530  9724.530
## 583   6356.271  6356.271
## 585   1242.816  1242.816
## 588  43943.876 43943.876
## 600  33471.972 33471.972
## 601   1633.044  1633.044
## 612   6571.544  6571.544
## 624  34617.841 34617.841
## 632   1977.815  1977.815
## 634   7173.360  7173.360
## 635   9391.346  9391.346
## 643  13143.865 13143.865
## 646  10141.136 10141.136
```

```
## 653   8280.623  8280.623
## 658   4058.712  4058.712
## 660  14394.398 14394.398
## 667   8703.456  8703.456
## 671   4837.582  4837.582
## 674   6185.321  6185.321
## 684   9863.472  9863.472
## 701   2020.552  2020.552
## 706   5375.038  5375.038
## 707  44400.406 44400.406
## 710   5469.007  5469.007
## 717   9566.991  9566.991
## 724   1263.249  1263.249
## 741   8604.484  8604.484
## 743  43254.418 43254.418
## 757   7985.815  7985.815
## 771  27941.288 27941.288
## 781  18259.216 18259.216
## 795   7209.492  7209.492
## 796  18310.742 18310.742
## 799  11848.141 11848.141
## 806   7731.858  7731.858
## 815   5584.306  5584.306
## 820  55135.402 55135.402
## 826  16069.085 16069.085
## 841   1526.312  1526.312
## 842  12323.936 12323.936
## 843  36021.011 36021.011
## 847   9872.701  9872.701
## 850  10601.632 10601.632
## 853  42111.665 42111.665
## 856   1875.344  1875.344
## 866   6600.361  6600.361
## 867   1141.445  1141.445
## 874   6849.026  6849.026
## 883   2585.851  2585.851
## 886  19719.695 19719.695
## 889   1682.597  1682.597
## 912  33732.687 33732.687
## 922  13462.520 13462.520
## 931   2927.065  2927.065
## 936  12233.828 12233.828
## 941   1121.874  1121.874
## 943   2217.469  2217.469
## 947   7160.094  7160.094
## 949   6358.776  6358.776
## 956   3875.734  3875.734
## 958  12609.887 12609.887
## 966   4746.344  4746.344
## 967  23967.383 23967.383
## 968   7518.025  7518.025
## 971  10702.642 10702.642
## 977   7804.160  7804.160
## 980   4889.037  4889.037
```

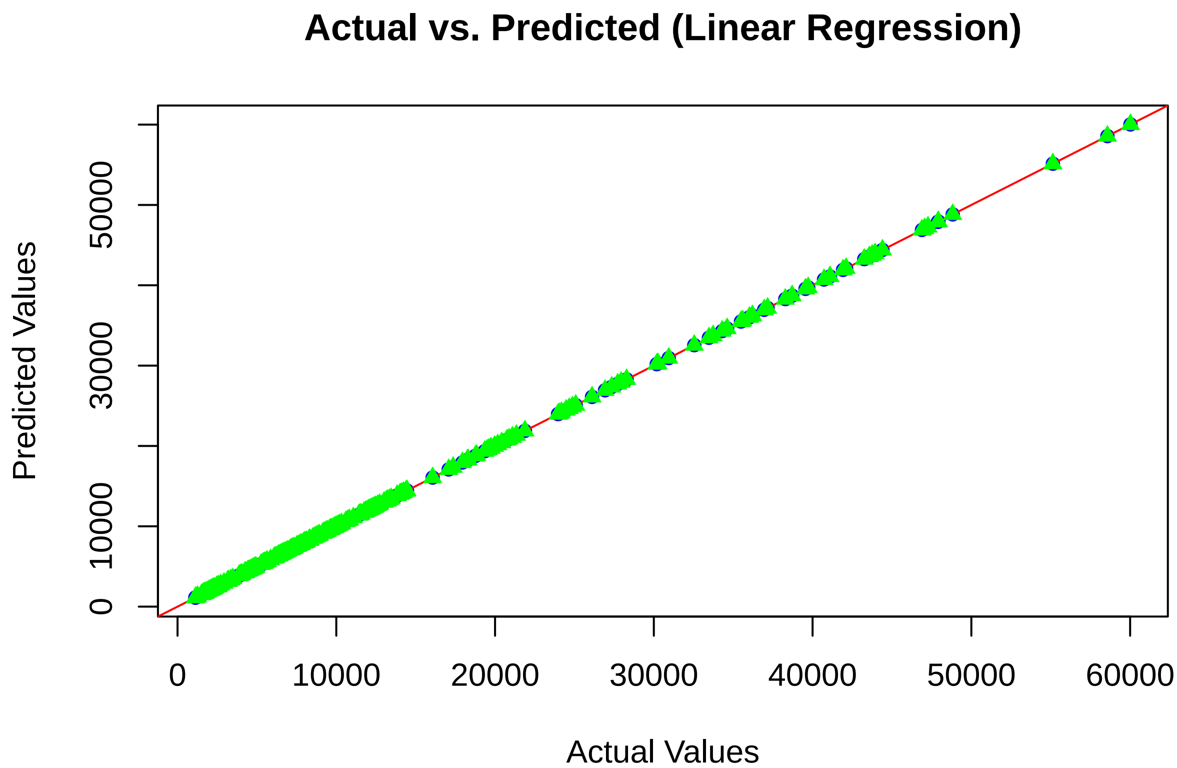
```
## 989   4518.826  4518.826
## 992   7144.863  7144.863
## 994   5484.467  5484.467
## 1000  5267.818  5267.818
## 1001 17361.766 17361.766
## 1010  9957.722  9957.722
## 1012 18767.738 18767.738
## 1022 35595.590 35595.590
## 1036 12094.478 12094.478
## 1038 39725.518 39725.518
## 1044  3161.454  3161.454
## 1046 21880.820 21880.820
## 1047  7325.048  7325.048
## 1051  8023.135  8023.135
## 1055  3353.470  3353.470
## 1057  8277.523  8277.523
## 1060  4462.722  4462.722
## 1061  1981.582  1981.582
## 1062 11554.224 11554.224
## 1075 13204.286 13204.286
## 1081 11884.049 11884.049
## 1083  5855.903  5855.903
## 1098  1674.632  1674.632
## 1105 20420.605 20420.605
## 1113 24180.933 24180.933
## 1117  9222.403  9222.403
## 1119 38282.749 38282.750
## 1127 10214.636 10214.636
## 1130  1728.897  1728.897
## 1137  7623.518  7623.518
## 1138  3176.288  3176.288
## 1142  7954.517  7954.517
## 1145  9630.397  9630.397
## 1161  7727.253  7727.253
## 1165  7153.554  7153.554
## 1170  6112.353  6112.353
## 1178  6496.886  6496.886
## 1180 19350.369 19350.369
## 1188 13844.797 13844.797
## 1196 18838.704 18838.704
## 1200  4934.705  4934.705
## 1202  8733.229  8733.229
## 1203  2055.325  2055.325
## 1215  3956.071  3956.071
## 1217  5415.661  5415.661
## 1220  7537.164  7537.164
## 1231 60021.399 60021.399
## 1232 20167.336 20167.336
## 1238 12224.351 12224.351
## 1241 47269.854 47269.854
## 1243  4296.271  4296.271
## 1246  5615.369  5615.369
## 1255  4415.159  4415.159
## 1266 26926.514 26926.514
```

```
## 1274  4747.053  4747.053
## 1293  1515.345  1515.345
## 1297  1708.926  1708.926
## 1299  5261.469  5261.469
## 1300  2710.829  2710.829
## 1306  2464.619  2464.619
## 1311  6940.910  6940.910
## 1319 19496.719 19496.719
## 1325  4239.893  4239.893
## 1330 10325.206 10325.206
## 1332 10795.937 10795.937
## 1333 11411.685 11411.685
```

```r
mse_linear = mean((predictions - actual4)^2)
cat('MSE for Linear Regression:', mse_linear, '\n')
```

```
## MSE for Linear Regression: 3.506902e-22
```

```r
plot(actual4, predictions,
     main = "Actual vs. Predicted (Linear Regression)",
     xlab = "Actual Values",
     ylab = "Predicted Values",
     col = "blue",  # Set the color of the points to blue
     pch = 16)      # Use solid circles for the points

abline(a = 0, b = 1, col = "red")

points(actual3, actual3, col = "green", pch = 17)
```



#removing charge outliers in the data

```
q = quantile(insurance$charges, c(0.25, 0.75))
iqr = q[2] - q[1]
lower_bound = q[1] - 1.5 * iqr
upper_bound = q[2] + 1.5 * iqr

outliers = insurance$charges < lower_bound | insurance$charges > upper_bound

insurance <- insurance[!outliers, ]
```

#knn with charge (no outliers) better rmse

```
#scaling with all predictors

library(FNN)
library(caret)
set.seed(2002)

#split into test and training
quantitative = c("bmi","age")
train_indices2=sample(1:nrow(insurance),0.8*nrow(insurance))
train=insurance[train_indices2,]
test=insurance[-train_indices2,]

#these are all the quantitative going to be scaled
```

```r
train_q=train[,quantitative]
testq=test[,quantitative]

#scaled quantitative
trains=scale(train_q)
tests=scale(testq)

#combine quantitative and qualitative
train_scaled=cbind(train[,setdiff(names(train),quantitative)],trains)
test_scaled=cbind(test[,setdiff(names(test),quantitative)],tests)

#convert predictors to numeric
train_scaled$sex = as.numeric(train_scaled$sex)
test_scaled$sex = as.numeric(test_scaled$sex)
train_scaled$smoker = as.numeric(train_scaled$smoker)
test_scaled$smoker = as.numeric(test_scaled$smoker)
train_scaled$region = as.numeric(train_scaled$region)
test_scaled$region = as.numeric(test_scaled$region)
train_scaled$children=as.numeric(train_scaled$children)
test_scaled$children=as.numeric(test_scaled$children)

#cross validation 5 k folds
ctrl = trainControl(method = "cv", number = 5)

#knn model which is the optimal k
knn_model3 = train(
  x = train_scaled[, -6],
  y = train_scaled$charges,
  method = "knn",
  tuneGrid = expand.grid(k = 1:50),
  trControl = ctrl
)
knn_model3
```

```
## k-Nearest Neighbors
##
## 959 samples
##   6 predictor
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 767, 767, 767, 767, 768
## Resampling results across tuning parameters:
##
##   k   RMSE       Rsquared   MAE
##    1   46.94489  0.9999571   20.69008
##    2   39.25268  0.9999707   19.48563
##    3   41.71236  0.9999669   21.34637
##    4   50.47833  0.9999515   24.23880
##    5   53.48680  0.9999492   26.78610
##    6   63.00199  0.9999300   30.35523
##    7   73.11336  0.9999010   33.43564
##    8   83.61086  0.9998677   36.70090
```

```
##    9   91.26492  0.9998412   37.99920
##   10  101.79758  0.9998022   41.76178
##   11  114.74027  0.9997603   44.18324
##   12  129.33529  0.9996977   46.66522
##   13  145.99342  0.9996224   50.04997
##   14  156.94474  0.9995622   53.01638
##   15  173.11719  0.9994855   55.73747
##   16  184.59934  0.9994052   58.48021
##   17  200.46629  0.9993070   62.89269
##   18  212.56644  0.9992060   66.76294
##   19  226.93307  0.9991053   69.73989
##   20  239.02908  0.9990099   72.66001
##   21  253.08162  0.9989126   77.42852
##   22  267.45684  0.9987869   80.85718
##   23  281.18090  0.9986633   83.49139
##   24  294.60009  0.9985386   86.24262
##   25  306.40526  0.9984142   88.64808
##   26  320.85364  0.9982847   92.15101
##   27  331.57241  0.9981685   95.70697
##   28  342.22824  0.9980558   98.28276
##   29  354.25079  0.9979355  101.75954
##   30  367.38217  0.9977969  105.04412
##   31  381.04984  0.9976602  109.53180
##   32  393.88855  0.9975201  113.41265
##   33  405.97206  0.9973687  116.81529
##   34  417.24432  0.9972579  121.83426
##   35  429.73039  0.9971069  125.03603
##   36  441.91876  0.9969644  128.60311
##   37  455.64511  0.9967830  132.75403
##   38  468.81677  0.9966210  136.75833
##   39  484.10467  0.9964224  142.02309
##   40  493.32702  0.9963046  144.99509
##   41  507.24590  0.9961292  150.12955
##   42  519.34646  0.9959481  153.86665
##   43  532.14557  0.9957653  157.91070
##   44  543.41370  0.9956326  161.75329
##   45  557.06637  0.9954590  165.74542
##   46  570.33143  0.9952810  171.13276
##   47  581.08002  0.9951041  174.81568
##   48  593.93048  0.9949031  178.57852
##   49  604.65457  0.9947443  182.02741
##   50  617.14003  0.9945560  185.51645
##
## RMSE was used to select the optimal model using the smallest value.
## The final value used for the model was k = 2.
```

```r
optimal_k = knn_model3$bestTune$k
cat('Optimal k:', optimal_k, '\n')
```

```
## Optimal k: 2
```

```r
#training with optimal k
final_knn_model = knn.reg(train_scaled[, -6], test_scaled[, -6], train_scaled$charges, k = optimal_k)
```

```r
#predicting
predict2=final_knn_model$pred

actual2 = test_scaled$charges

#comparing actual and predictions
compare2= data.frame(Actual = actual2,Predicted = predict2)
compare2
```

```
##       Actual Predicted
## 1   4449.462  4447.808
## 2   3866.855  3868.472
## 3  28923.137 28909.567
## 4   2721.321  2731.010
## 5  27808.725 27832.788
## 6   2395.172  2406.525
## 7  13228.847 13224.375
## 8   1137.011  1136.935
## 9   6203.902  6197.600
## 10 14001.134 13991.569
## 11  2198.190  2198.652
## 12  3579.829  3584.739
## 13  8606.217  8604.923
## 14  1743.214  1740.226
## 15 14235.072 14232.572
## 16 11741.726 11740.574
## 17  3947.413  3949.833
## 18  1532.470  1530.308
## 19  2755.021  2758.570
## 20 21098.554 21033.127
## 21 12105.320 12095.565
## 22 10226.284 10223.068
## 23  3645.089  3595.883
## 24  2404.734  2406.525
## 25  8601.329  8604.154
## 26 10115.009 10112.822
## 27 17081.080 17153.554
## 28  9634.538  9628.159
## 29 13616.359 13581.187
## 30 11163.568 11158.099
## 31  1261.442  1262.554
## 32 27375.905 27334.388
## 33  9877.608  9877.874
## 34  5028.147  5021.870
## 35  4830.630  4837.413
## 36  2719.280  2719.610
## 37  1694.796  1704.634
## 38  5246.047  5249.375
## 39  8538.288  8541.182
## 40 11735.879 11733.764
## 41  5325.651  5319.785
## 42  6775.961  6775.773
```

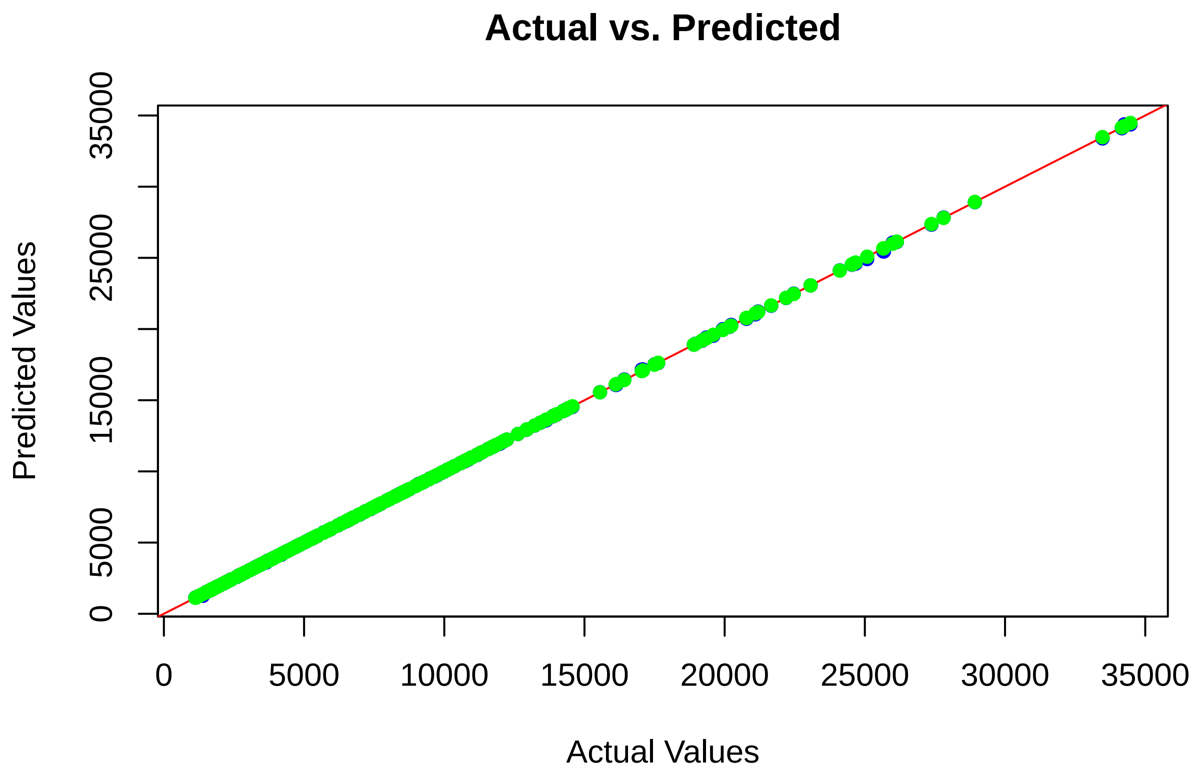
```
## 43   1639.563  1637.648
## 44   8516.829  8517.892
## 45   9644.253  9628.159
## 46   7147.105  7146.168
## 47   4337.735  4343.732
## 48  13880.949 13866.383
## 49   6610.110  6596.935
## 50   7371.772  7353.159
## 51  10355.641 10360.205
## 52  25081.768 24915.134
## 53  10564.885 10570.101
## 54  11987.168 11945.879
## 55   2689.495  2699.613
## 56   6710.192  6716.587
## 57   7196.867  7205.596
## 58   1986.933  1983.018
## 59   4260.744  4254.878
## 60  17085.268 17153.554
## 61  12928.791 12919.939
## 62   4237.127  4237.410
## 63  14256.193 14269.034
## 64  25992.821 26064.140
## 65   2156.752  2153.075
## 66   3906.127  3901.531
## 67   9249.495  9245.027
## 68  20177.671 20158.329
## 69   7749.156  7741.223
## 70   1737.376  1730.287
## 71  24671.663 24593.842
## 72   6600.206  6596.935
## 73   3561.889  3557.771
## 74  18955.220 18967.833
## 75  24603.048 24593.842
## 76   2597.779  2585.560
## 77  13430.265 13422.037
## 78   7639.417  7637.015
## 79   1391.529  1262.554
## 80  21659.930 21636.333
## 81  20781.489 20727.505
## 82   5846.918  5846.211
## 83  10736.871 10708.143
## 84   7526.706  7527.595
## 85   3260.199  3273.004
## 86   4185.098  4150.382
## 87   8539.671  8541.182
## 88  19594.810 19530.606
## 89   2727.395  2731.010
## 90  11840.775 11839.801
## 91   2203.472  2203.406
## 92   1744.465  1740.226
## 93   1824.285  1829.468
## 94  15555.189 15565.187
## 95   1622.188  1621.611
## 96   3044.213  3051.225
```

```
## 97   8413.463  8419.058
## 98   5240.765  5249.375
## 99  25656.575 25449.705
## 100  5397.617  5393.159
## 101 13887.204 13903.896
## 102 11187.657 11158.099
## 103  1646.430  1637.648
## 104  9058.730  9074.913
## 105  2801.259  2796.378
## 106 11552.904 11560.262
## 107  3761.292  3761.753
## 108  4753.637  4749.061
## 109 12222.898 12227.982
## 110 17626.240 17611.762
## 111 13635.638 13581.187
## 112  5976.831  5977.058
## 113  9283.562  9286.310
## 114 25678.778 25449.705
## 115  6571.544  6563.047
## 116  1880.070  1879.208
## 117  3659.346  3650.975
## 118  9182.170  9183.987
## 119 12129.614 12133.785
## 120 11365.952 11363.019
## 121  8280.623  8280.602
## 122  8527.532  8521.015
## 123 22192.437 22181.073
## 124  8703.456  8711.044
## 125  6500.236  6485.450
## 126  4837.582  4837.413
## 127 10976.246 10979.854
## 128  5375.038  5375.411
## 129  5469.007  5475.243
## 130  8310.839  8318.497
## 131 10848.134 10816.370
## 132 10106.134 10112.822
## 133 14007.222 13991.569
## 134  3757.845  3761.753
## 135  3062.508  3063.598
## 136  1906.358  1913.423
## 137 23065.421 23064.261
## 138  9095.068  9121.374
## 139 11842.624 11839.801
## 140  8062.764  8063.932
## 141  7448.404  7444.781
## 142  5934.380  5923.475
## 143  1252.407  1255.117
## 144 21195.818 21227.929
## 145  4719.524  4729.002
## 146 14313.846 14301.245
## 147  7731.858  7730.536
## 148  2680.949  2699.613
## 149  8219.204  8221.870
## 150 20773.628 20727.505
```

```
## 151 11743.934 11740.574
## 152 11657.719 11658.247
## 153 10601.632 10601.898
## 154 24106.913 24120.307
## 155  5458.046  5455.599
## 156 26140.360 26117.502
## 157  3443.064  3446.096
## 158  4877.981  4886.451
## 159  1682.597  1669.816
## 160 17496.306 17514.682
## 161 14382.709 14394.478
## 162  7626.993  7624.074
## 163  5488.262  5481.252
## 164 10096.970 10096.533
## 165  8965.796  8966.195
## 166  2304.002  2312.461
## 167  9487.644  9502.442
## 168  1121.874  1133.953
## 169 19933.458 19987.190
## 170 16138.762 16077.106
## 171 19199.944 19179.641
## 172 14571.891 14532.654
## 173 16420.495 16453.301
## 174 34472.841 34371.512
## 175  8627.541  8610.331
## 176  4433.388  4434.505
## 177  9957.722  9950.699
## 178  8765.249  8757.849
## 179  2709.244  2699.613
## 180  1711.027  1710.114
## 181  4137.523  4133.862
## 182 12950.071 12953.137
## 183 20234.855 20287.335
## 184 33475.817 33389.761
## 185  9288.027  9286.310
## 186  3353.470  3359.977
## 187 14349.854 14338.698
## 188 10928.849 10933.033
## 189  2102.265  2103.597
## 190  9748.911  9723.650
## 191 10577.087 10570.101
## 192 11299.343 11296.237
## 193  4561.189  4563.517
## 194  3471.410  3483.100
## 195  2904.088  2900.115
## 196  5693.431  5709.016
## 197 34166.273 34105.358
## 198 18903.491 18900.938
## 199  3693.428  3718.490
## 200  3176.288  3174.417
## 201  7954.517  7960.553
## 202  6338.076  6345.307
## 203 11289.109 11296.237
## 204  2203.736  2203.406
```

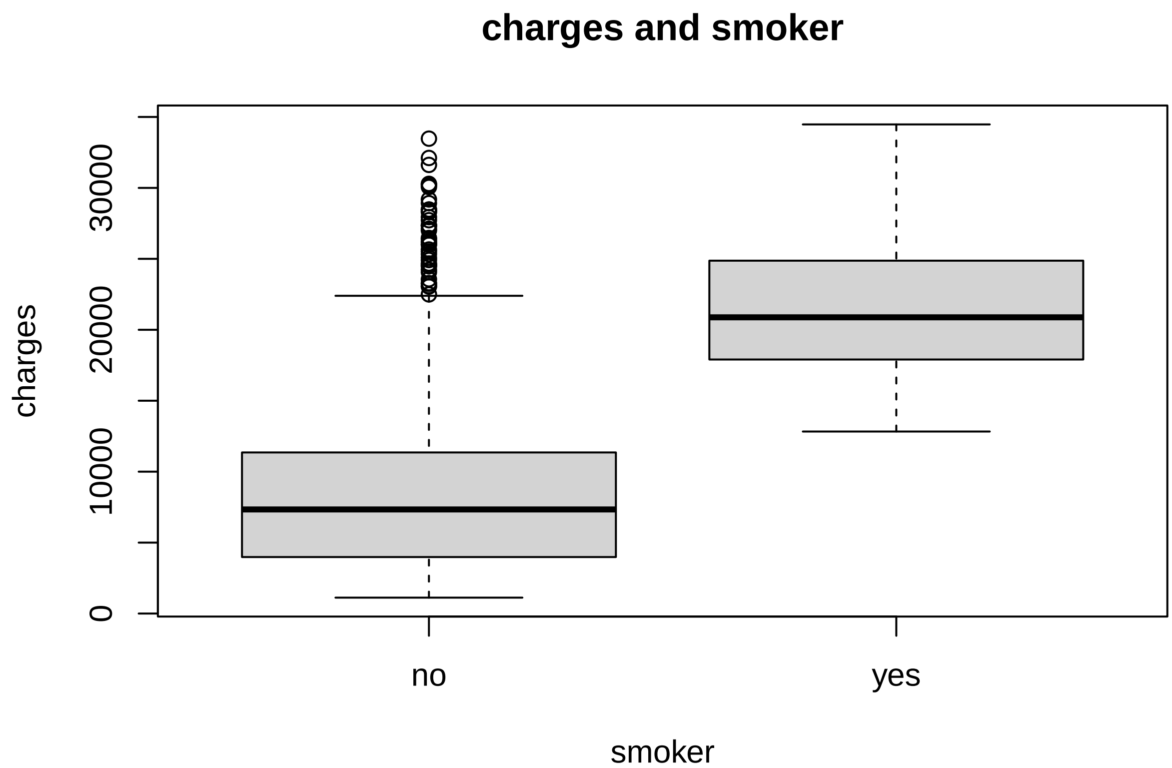
```
## 205 12235.839 12232.721
## 206  4670.640  4670.500
## 207  2154.361  2153.075
## 208  2899.489  2900.115
## 209 19350.369 19402.176
## 210  7650.774  7637.015
## 211  9447.382  9440.088
## 212  5699.837  5709.016
## 213  9964.060  9950.699
## 214  5116.500  5124.702
## 215  1702.455  1704.634
## 216  4058.116  4066.583
## 217  4718.204  4729.002
## 218  7162.012  7160.212
## 219  2699.568  2699.613
## 220 14449.854 14453.740
## 221  6985.507  6967.699
## 222  1135.941  1136.935
## 223 10370.913 10360.205
## 224 10704.470 10708.143
## 225 34254.053 34371.512
## 226 14478.330 14465.160
## 227 17043.341 17153.554
## 228 10959.330 10962.570
## 229 22462.044 22486.130
## 230  4189.113  4150.382
## 231 24535.699 24516.678
## 232  1720.354  1722.494
## 233  1515.345  1530.308
## 234  1708.926  1710.114
## 235  2710.829  2719.610
## 236 16115.305 16077.106
## 237  6940.910  6940.972
## 238 12629.166 12626.038
## 239 10795.937 10796.843
## 240  1629.833  1630.070
```

```r
#plotting
plot(actual2, predict2,
     main = "Actual vs. Predicted",
     xlab = "Actual Values",
     ylab = "Predicted Values",
     col = "blue",  # Set the color of the points to blue
     pch = 16)      # Use solid circles for the points


abline(a = 0, b = 1, col = "red")


points(actual2, actual2, col = "green", pch = 16)  # Use solid triangles for the actual values
```

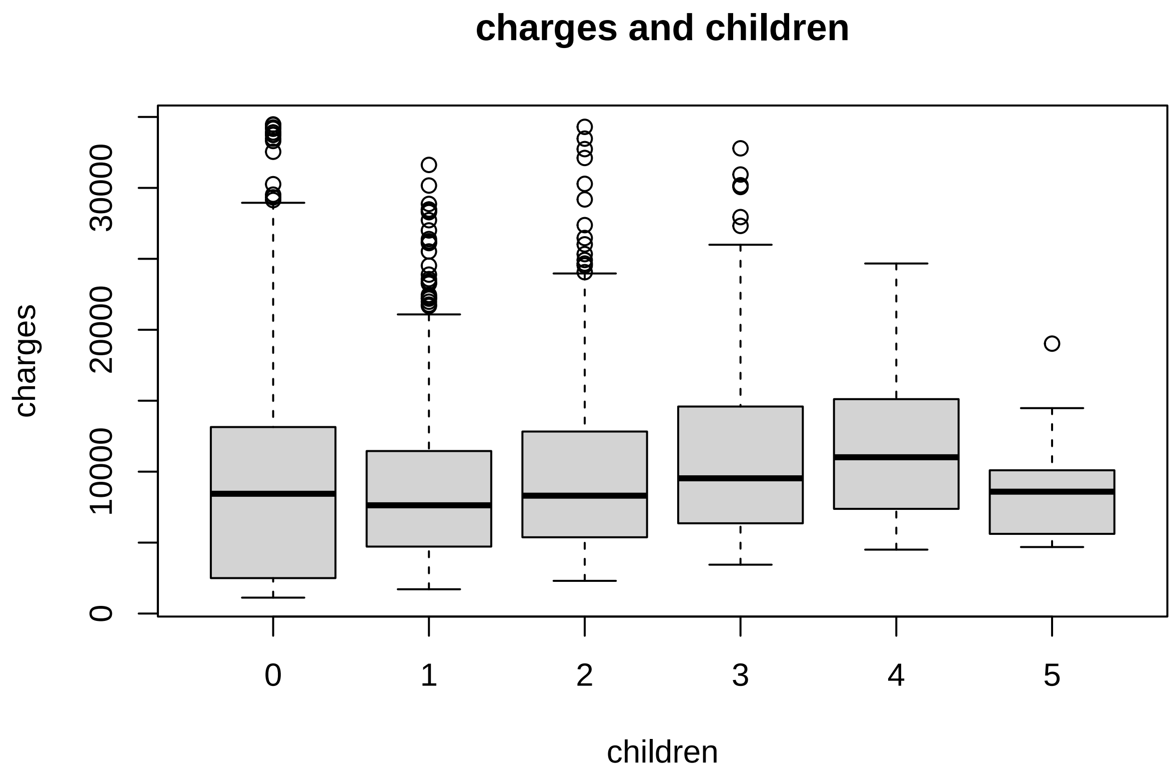


#visuals with no charge outliers

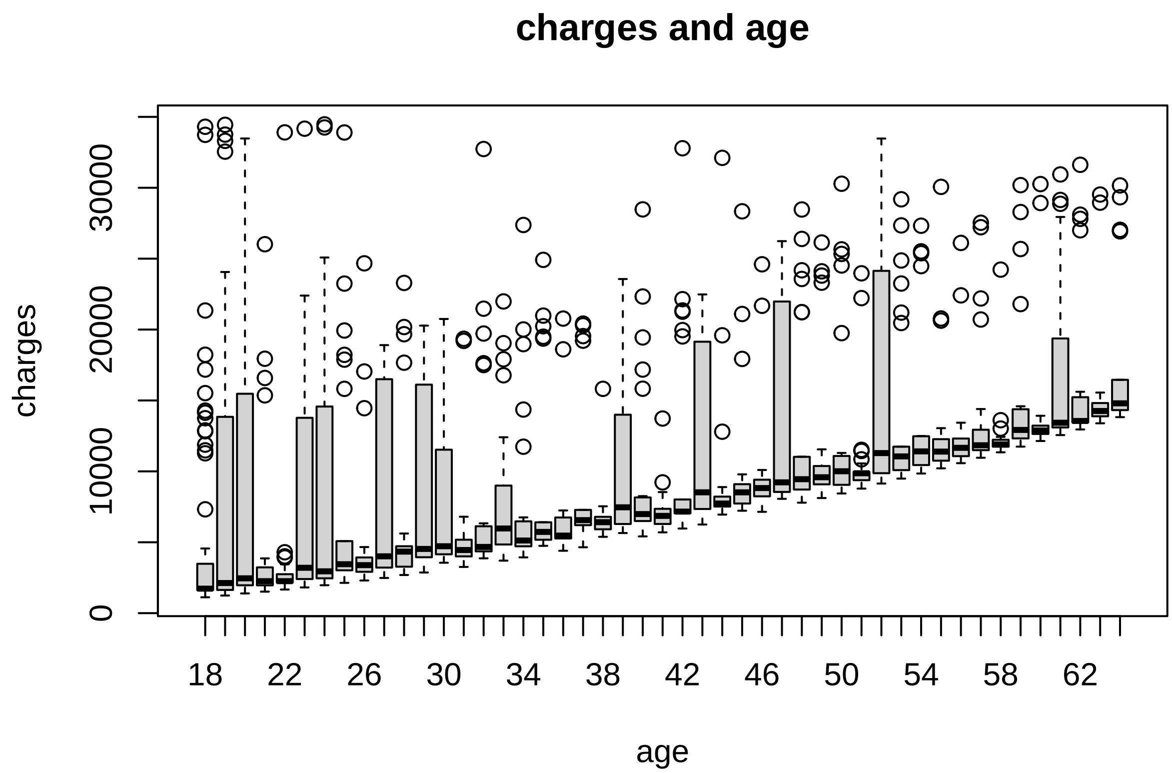
```
#visuals and characteristics
boxplot(charges~smoker, data=insurance,main="charges and smoker")
```



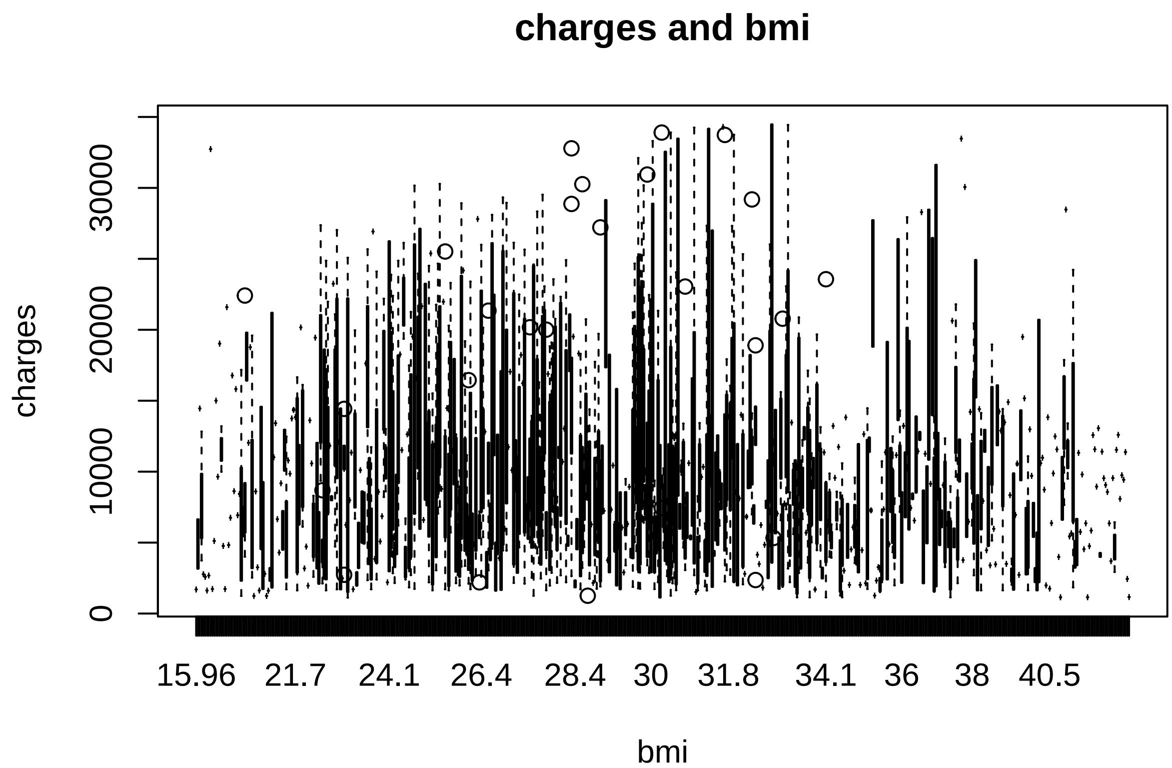

```r
boxplot(charges~children, data=insurance,main="charges and children")
```



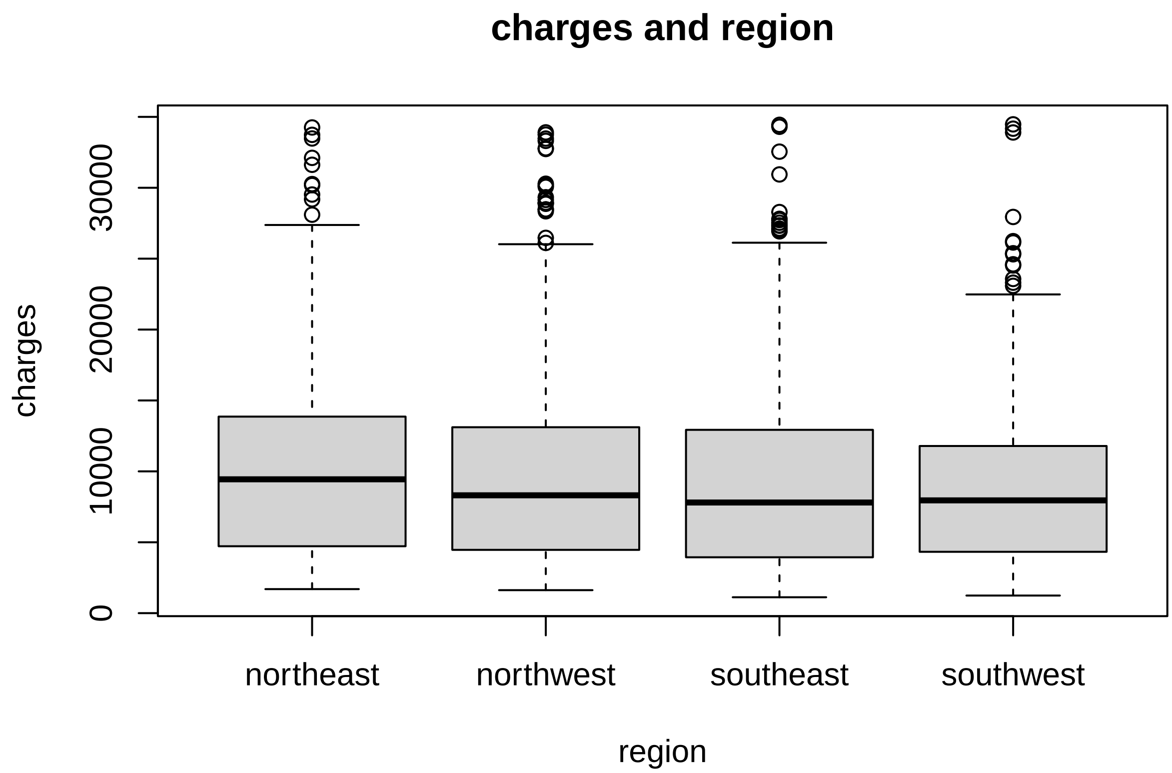

```
boxplot(charges~age, data=insurance,main="charges and age")
```



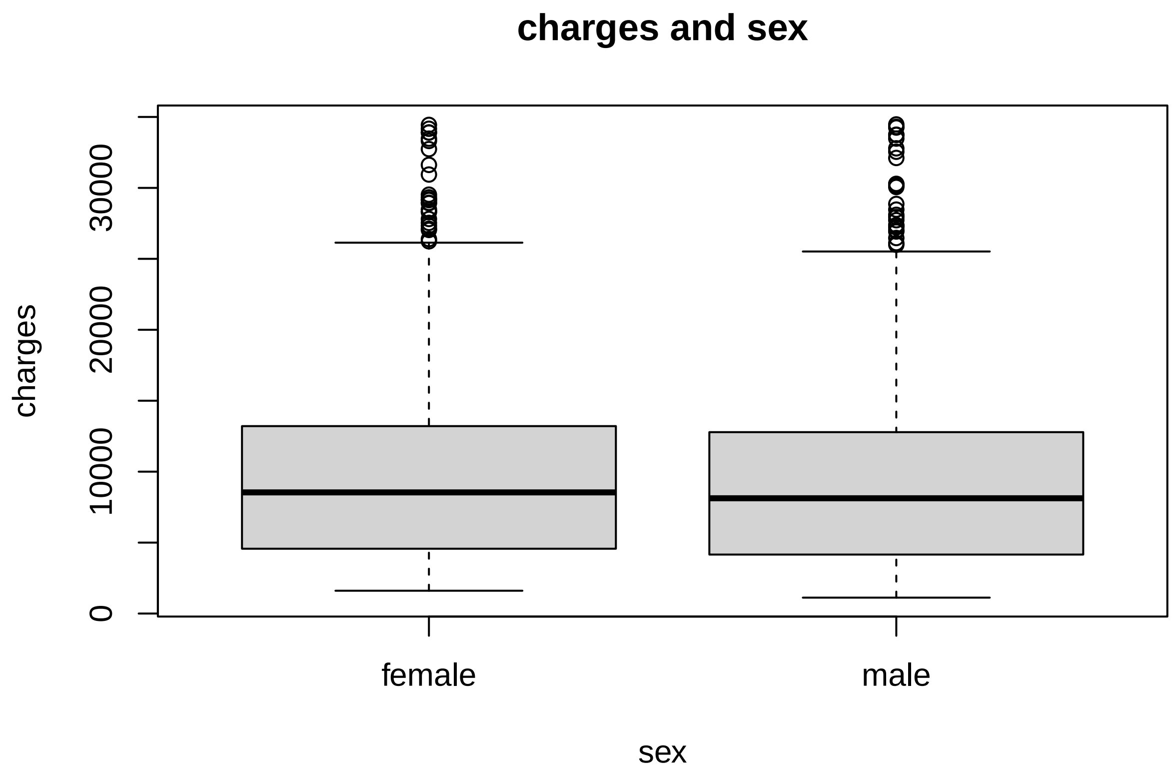

```
boxplot(charges~bmi, data=insurance,main="charges and bmi")
```




```r
boxplot(charges~region,data=insurance,main="charges and region")
```




```
boxplot(charges~sex,data=insurance,main="charges and sex")
```


charges and sex
charges
0
10000
20000
30000
female
male
sex