IS-804 Project Report - Energy Efficient Data

Introduction

The report intends to present all the statistical learning concepts introduced in the text book. The report is presented for a dataset named ‘Energy Efficient’. This dataset is collected from UCI website. There are two predicting parameter in the dataset i.e. Heating and cooling load while there are 8 predictor variables. This report will present regression technique to predict the output variables. The data can be found in following link. https://archive.ics.uci.edu/ml/datasets/Energy+efficiency (https://archive.ics.uci.edu/ml/datasets/Energy+efficiency)

FIrst load dataset

Chapter 2

Loading Data

```
library(readxl)
```

```
## Warning: package 'readxl' was built under R version 3.3.3
```

```
energy <- read_excel("F:/Spring 2018/quantitative methods/Project/Data/energy.xlsx")
#View(energy)
#fix(energy)
dim(energy)
```

```
## [1] 768  10
```

```
names(energy)
```

```
##  [1] "relative_compactness"      "surface_area"
##  [3] "wall_area"                 "roof_area"
##  [5] "overall_height"            "orientation"
##  [7] "glazing_area"              "glazing_area_distribution"
##  [9] "heating_load"              "cooling_load"
```

```
summary(energy)
```

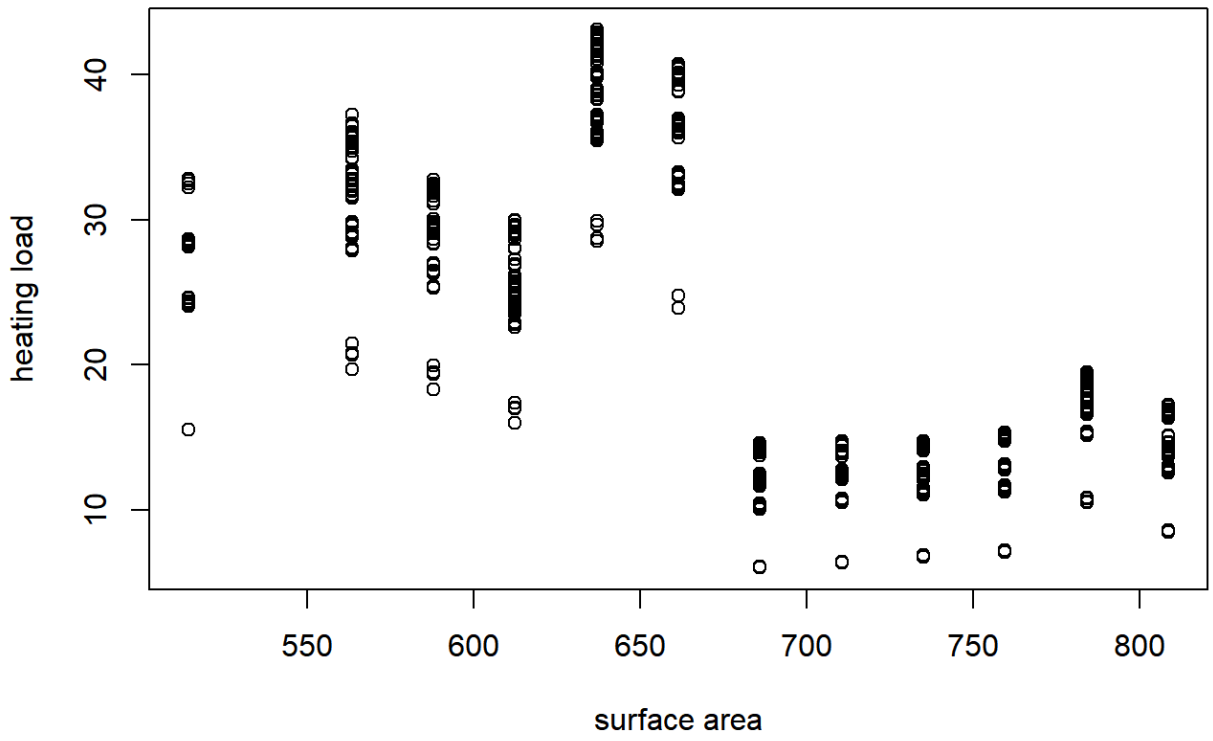
```
##  relative_compactness  surface_area     wall_area       roof_area
##  Min.   :0.6200       Min.   :514.5   Min.   :245.0   Min.   :110.2
##  1st Qu.:0.6825       1st Qu.:606.4   1st Qu.:294.0   1st Qu.:140.9
##  Median :0.7500       Median :673.8   Median :318.5   Median :183.8
##  Mean   :0.7642       Mean   :671.7   Mean   :318.5   Mean   :176.6
##  3rd Qu.:0.8300       3rd Qu.:741.1   3rd Qu.:343.0   3rd Qu.:220.5
##  Max.   :0.9800       Max.   :808.5   Max.   :416.5   Max.   :220.5
##  overall_height  orientation    glazing_area    glazing_area_distribution
##  Min.   :3.50   Min.   :2.00   Min.   :0.0000   Min.   :0.000
##  1st Qu.:3.50   1st Qu.:2.75   1st Qu.:0.1000   1st Qu.:1.750
##  Median :5.25   Median :3.50   Median :0.2500   Median :3.000
##  Mean   :5.25   Mean   :3.50   Mean   :0.2344   Mean   :2.812
##  3rd Qu.:7.00   3rd Qu.:4.25   3rd Qu.:0.4000   3rd Qu.:4.000
##  Max.   :7.00   Max.   :5.00   Max.   :0.4000   Max.   :5.000
##   heating_load    cooling_load
##  Min.   : 6.01   Min.   :10.90
##  1st Qu.:12.99   1st Qu.:15.62
##  Median :18.95   Median :22.08
##  Mean   :22.31   Mean   :24.59
##  3rd Qu.:31.67   3rd Qu.:33.13
##  Max.   :43.10   Max.   :48.03
```

```
attach(energy)
```

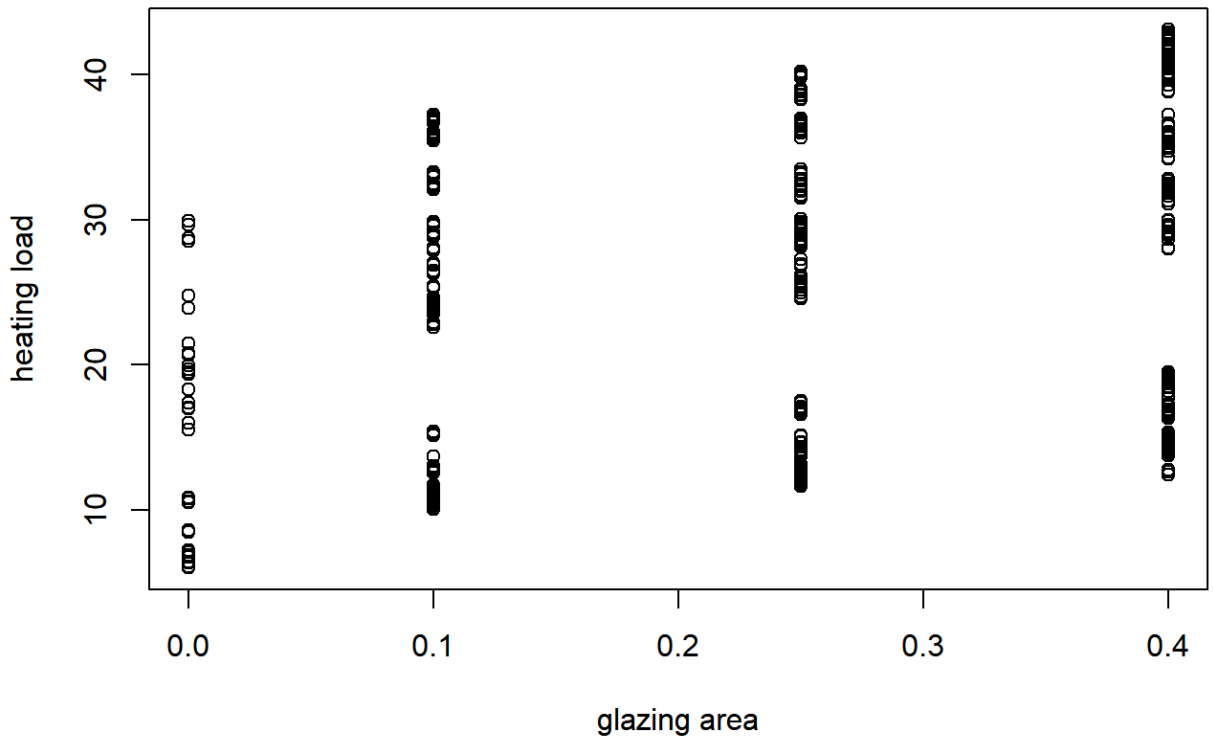
Additional Graphical and Numerical Summaries

Following figures shows how the output variables changes with respect to different input variables.

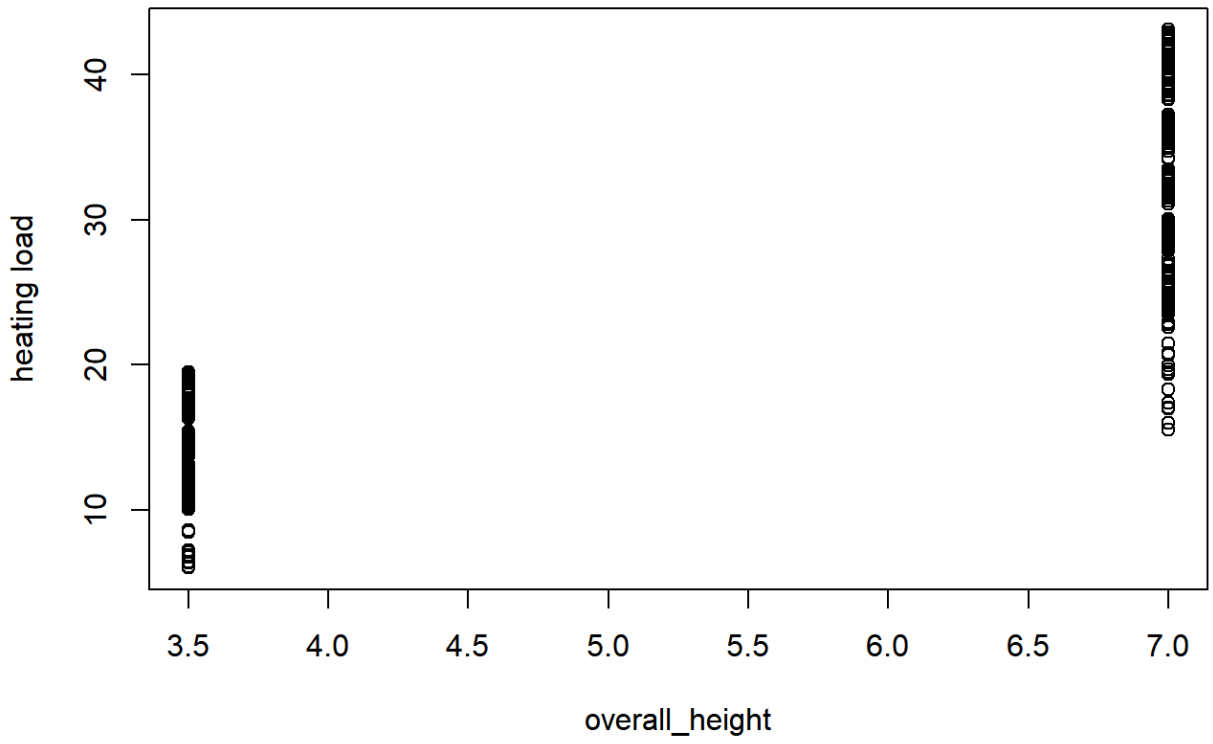
```
plot(surface_area, heating_load,xlab='surface area', ylab='heating load')
```



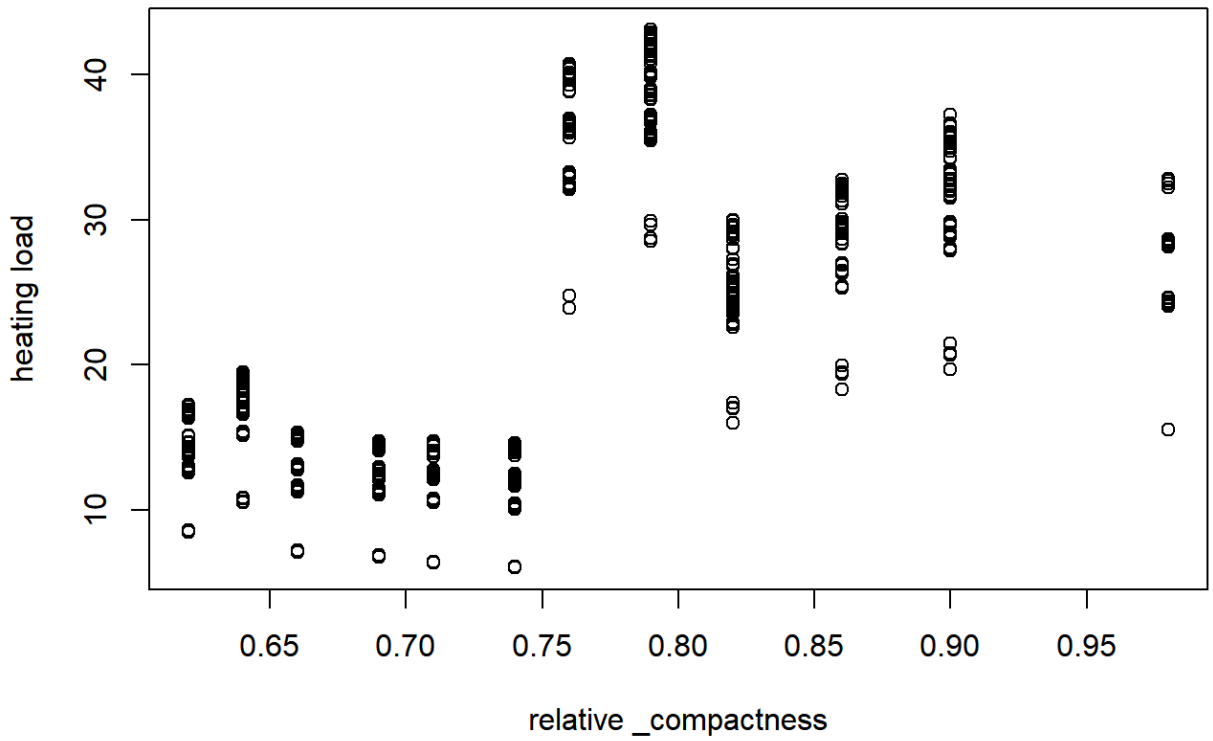

```
plot(glazing_area, heating_load,xlab='glazing area', ylab='heating load')
```



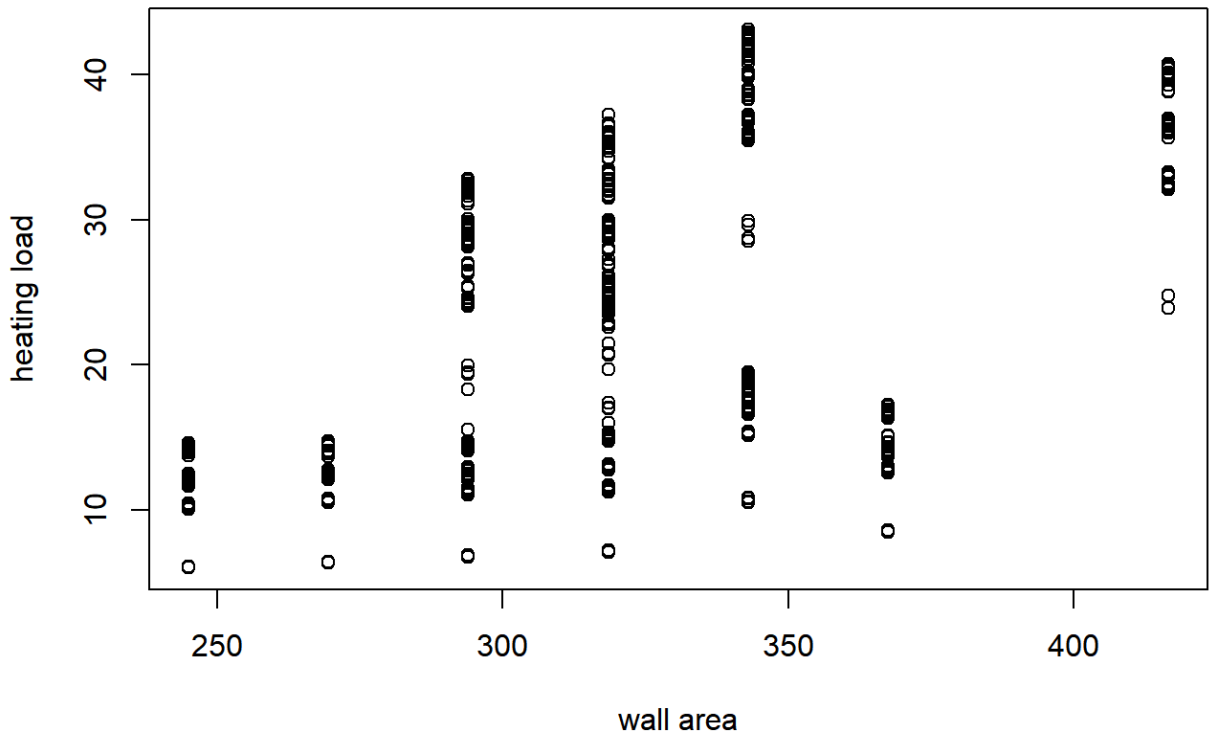

```
plot(overall_height, heating_load,xlab='overall_height', ylab='heating load')
```



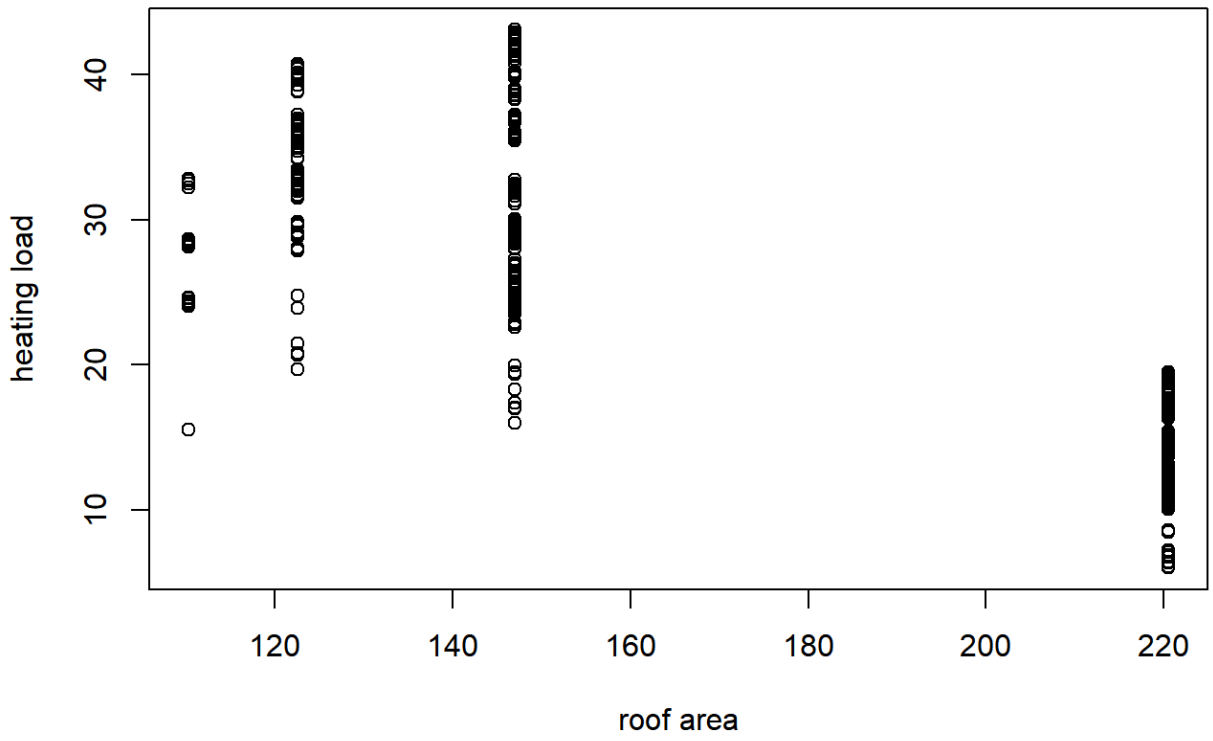

```
plot(relative_compactness, heating_load, xlab='relative _compactness', ylab='heating l
oad')
```



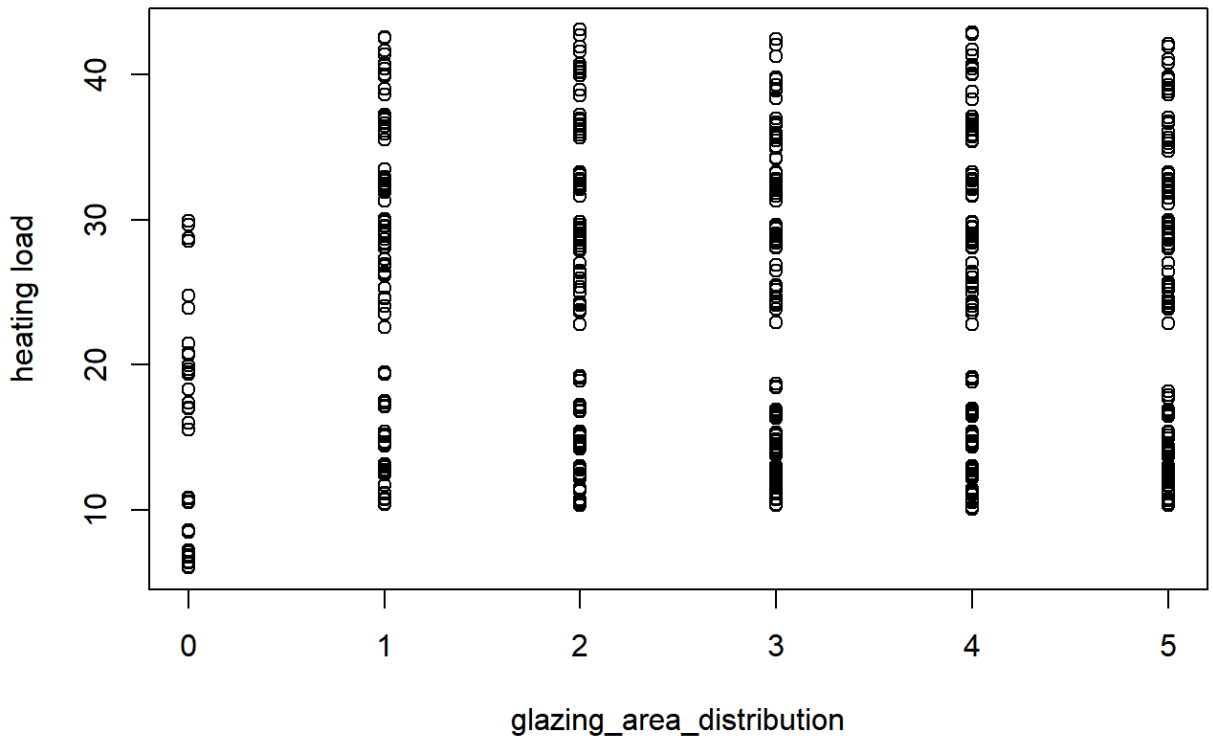

```
plot(wall_area, heating_load,xlab='wall area', ylab='heating load')
```



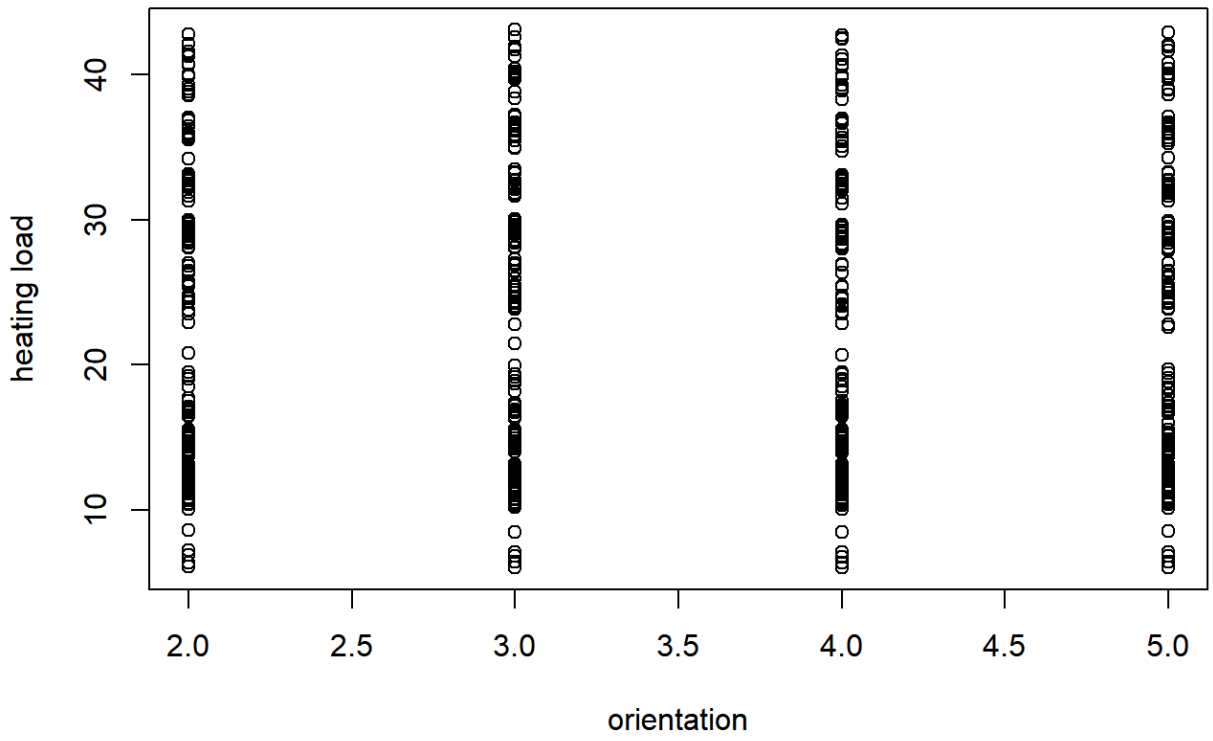

```
plot(roof_area, heating_load,xlab='roof area', ylab='heating load')
```



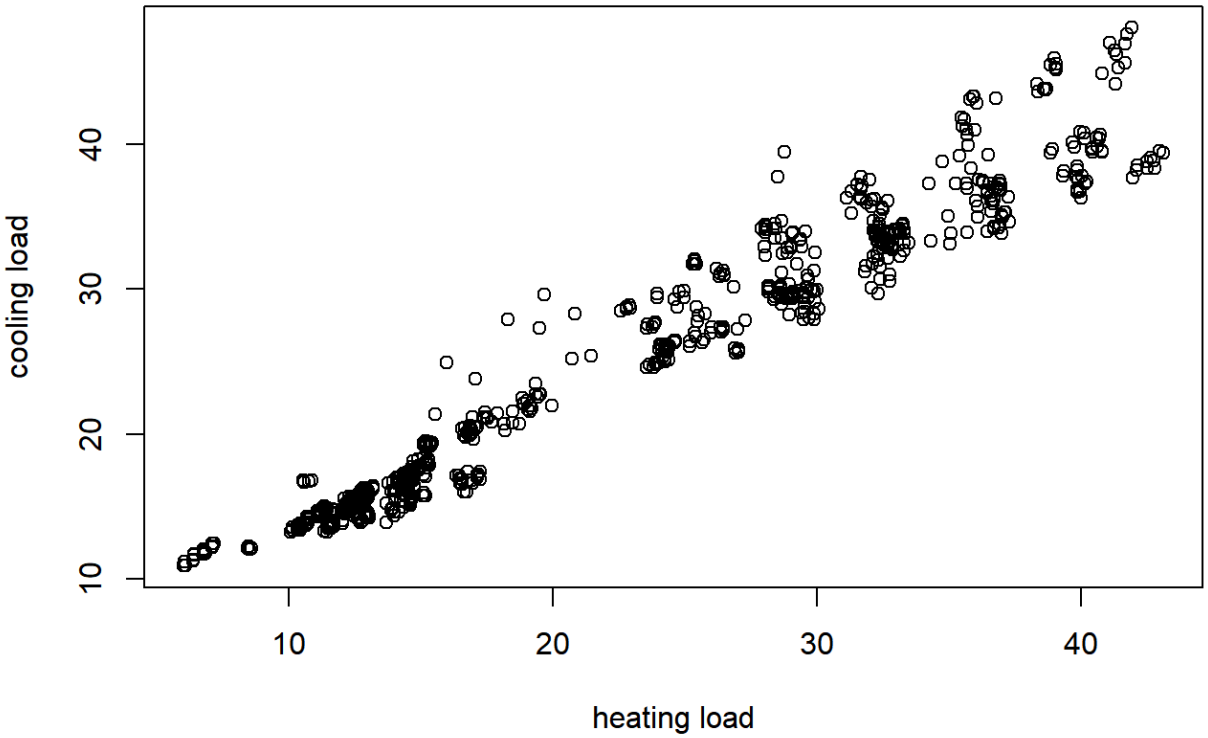

```
plot(glazing_area_distribution, heating_load,xlab='glazing_area_distribution', ylab='h
eating load')
```




```
plot(orientation, heating_load,xlab='orientation', ylab='heating load')
```




```
plot(heating_load,cooling_load,xlab='heating load', ylab='cooling load')
```



From the above figures following observations can be made:

. surface area with less than 0 results in higher heating load.

2. with increasing gla ing area and overall height heating load increases.

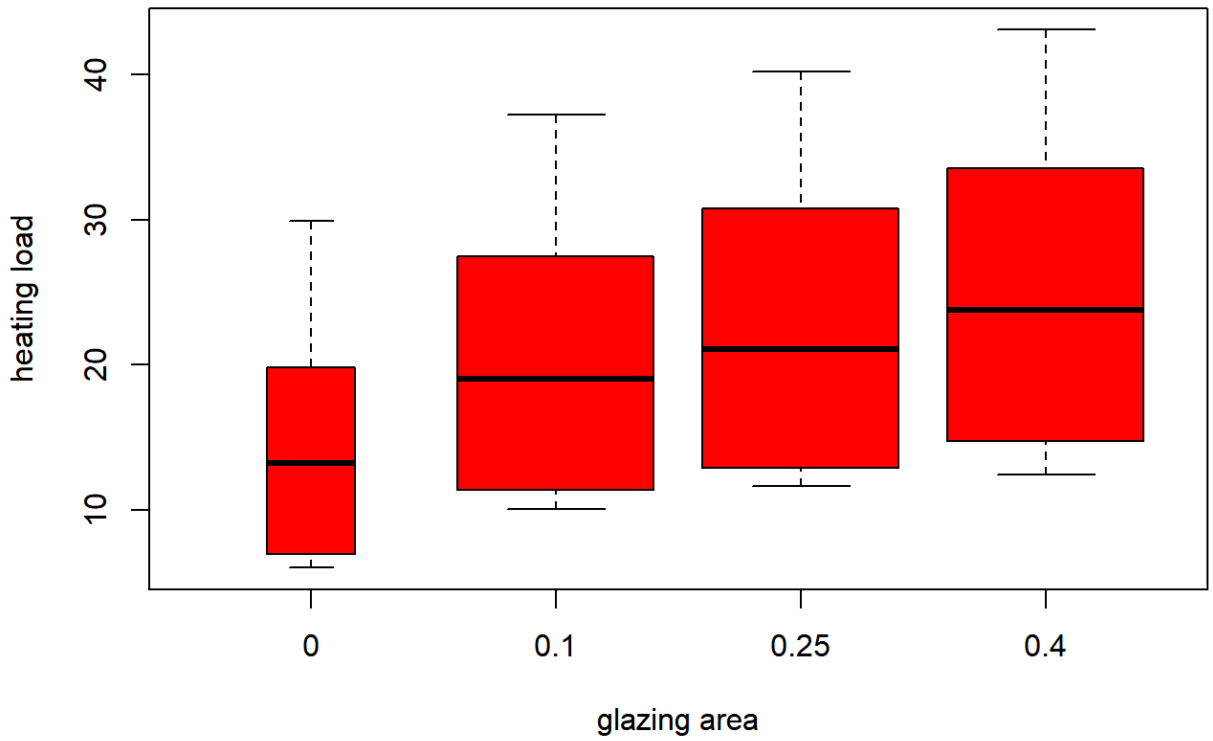
. The relaship of heating load with relative compactness wall area roof area is changing randomly.

4. For all orientations and gla ing area distribution heating load is almost same.

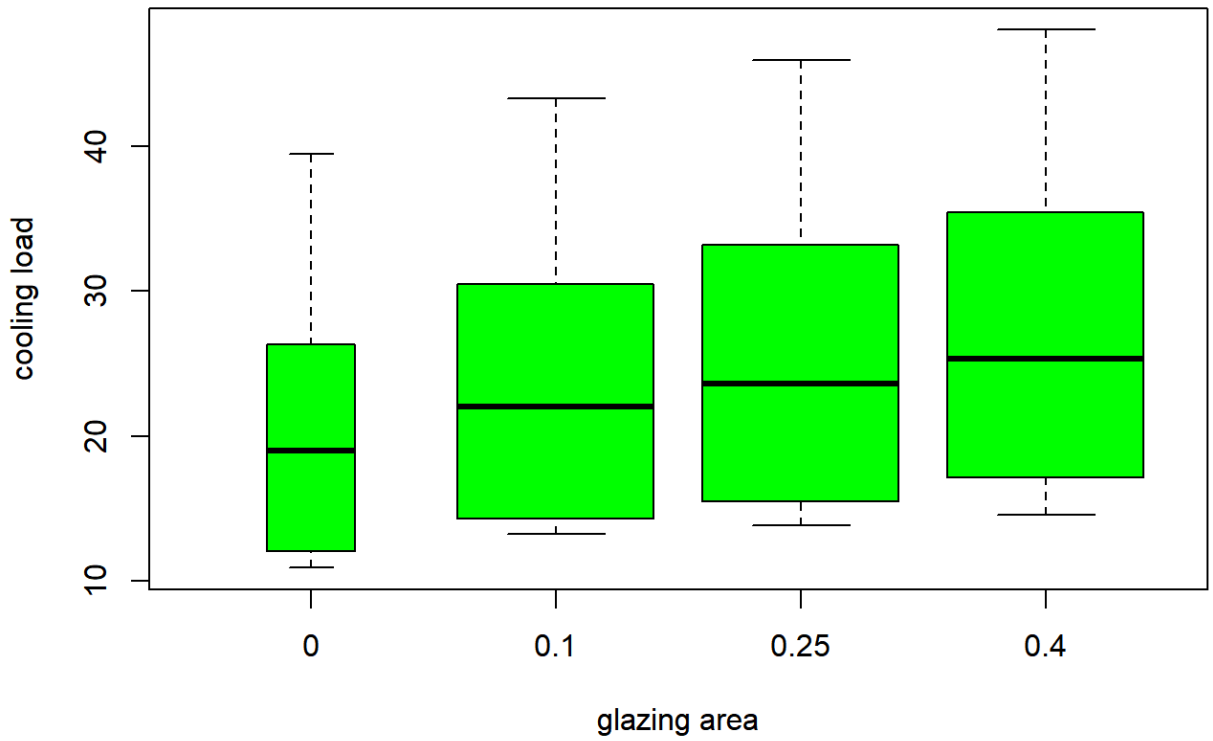
. Two output variables i.e. hetaing and cooling load are linearly correlated with each other.

As heating and cooling load are almost similar therefore most of the codes are used just to predict heating load. Representing variables with qualitative values Eight quantitative input variables are represented as qualititative variables.

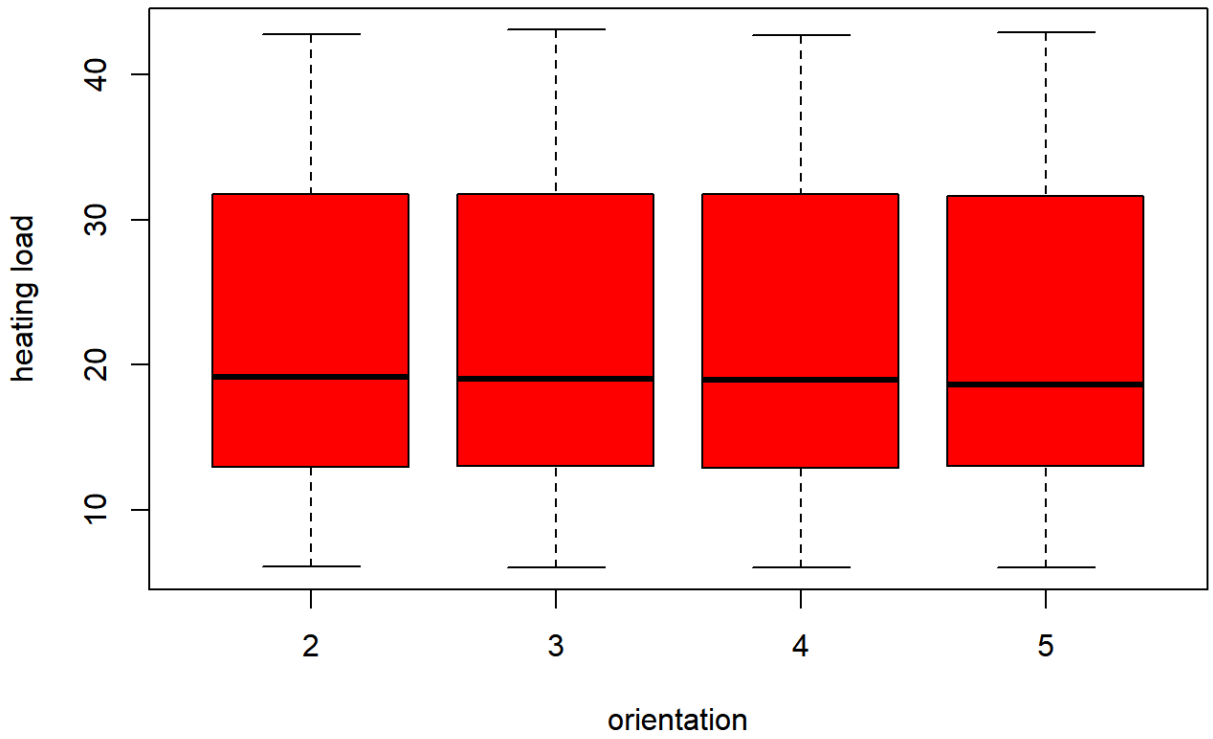
```
qual_glazing_area=as.factor(glazing_area)
qual_orientation=as.factor(orientation)
qual_wall_area= as.factor(wall_area)
qual_surface_area=as.factor(surface_area)
qual_roof_area=as.factor(roof_area)
qual_overall_height=as.factor(overall_height)
qual_relative_compactness=as.factor(relative_compactness)

plot(qual_glazing_area, heating_load, col="red", varwidth= , xlab='glazing area', ylab
='heating load')
```



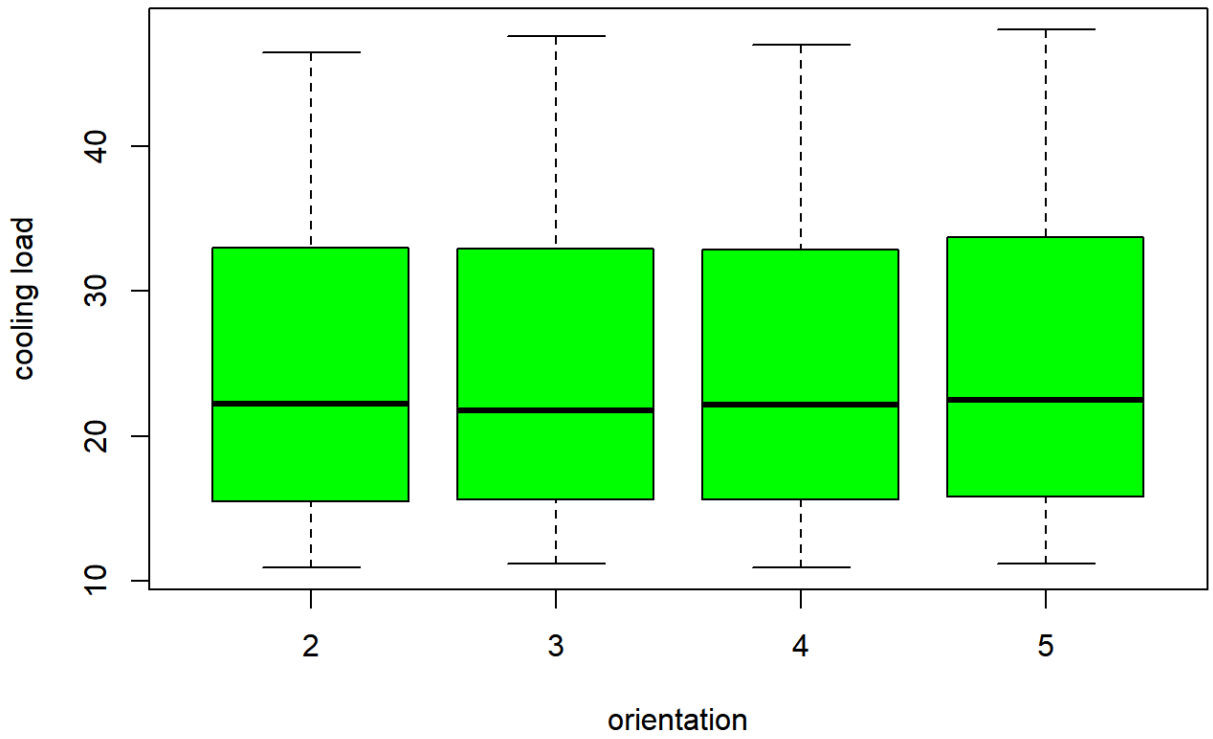

```
plot(qual_glazing_area, cooling_load, col="green", varwidth= , xlab='glazing area', yl
ab='cooling load')
```



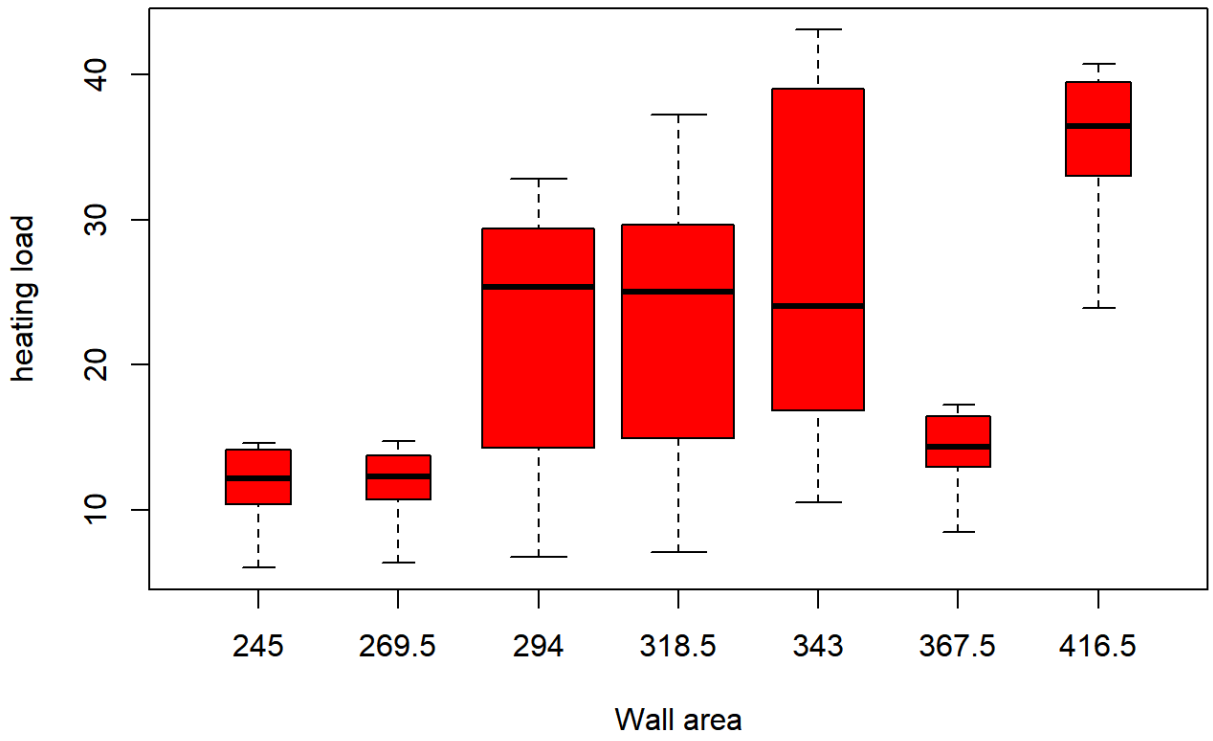

```
plot(qual_orientation, heating_load, col="red", varwidth= , xlab='orientation', ylab
='heating load')
```



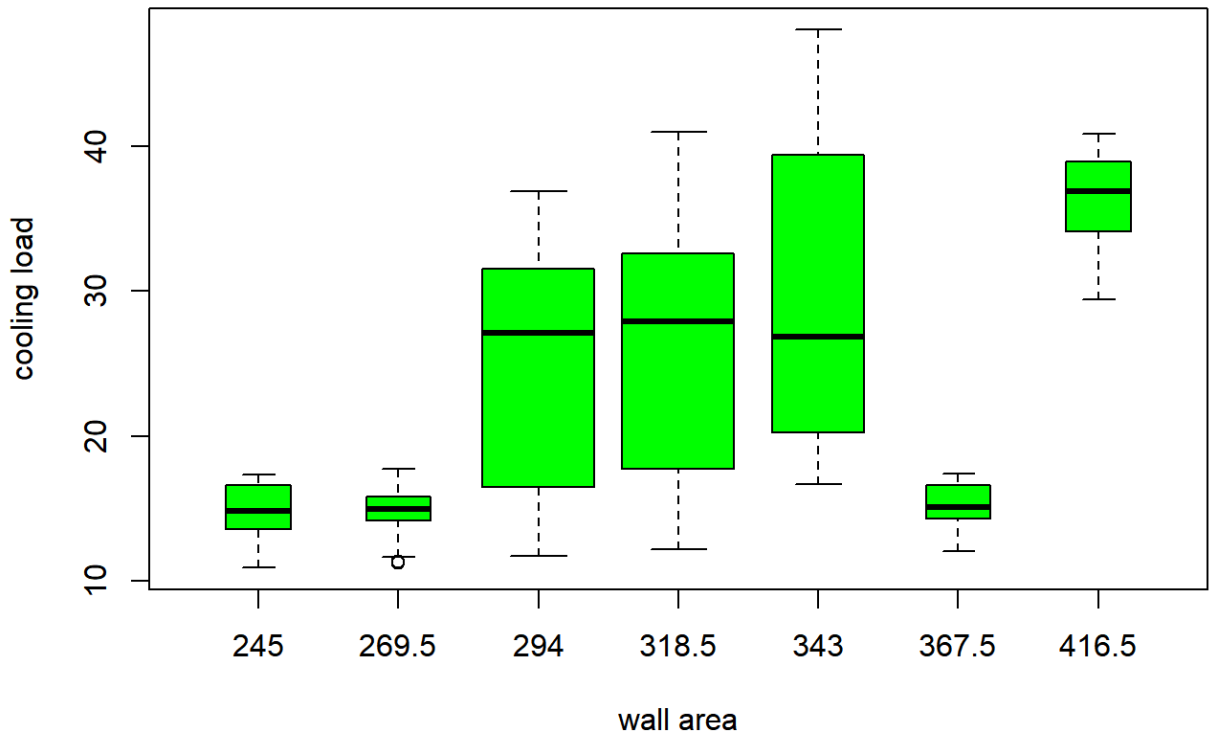

```
plot(qual_orientation, cooling_load, col="green", varwidth= , xlab='orientation', ylab
='cooling load')
```



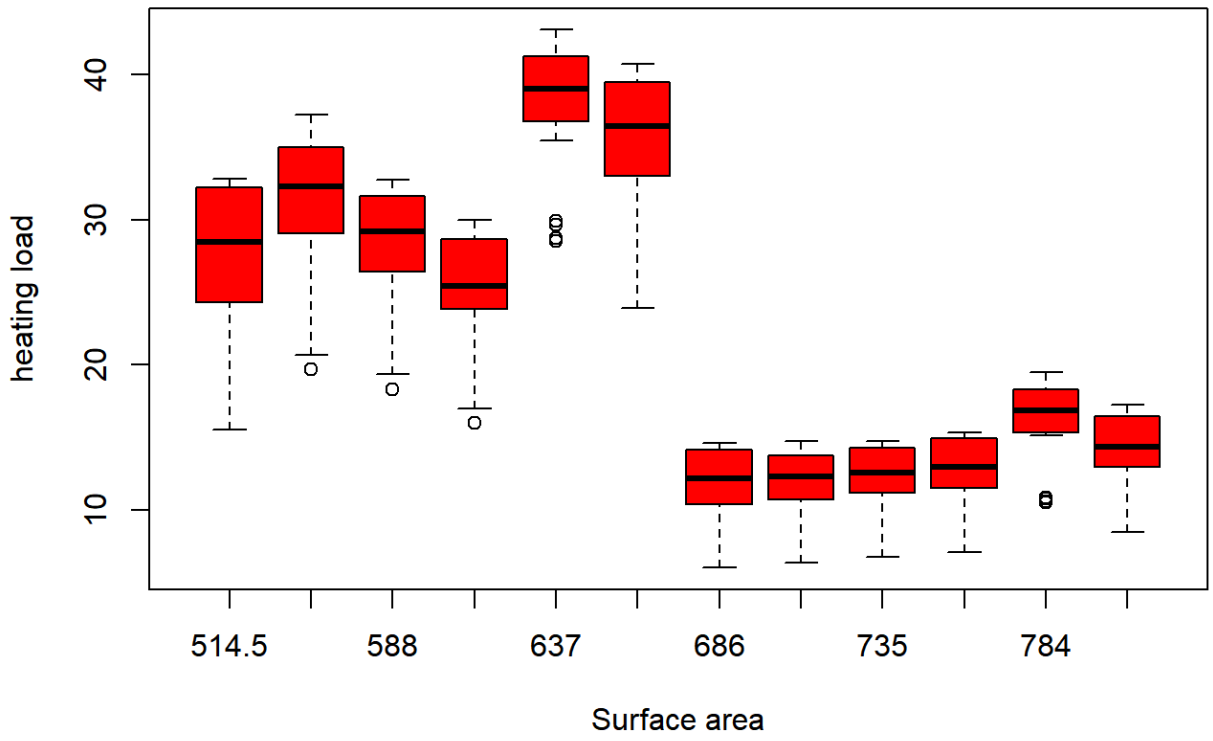

```
plot(qual_wall_area, heating_load, col="red", varwidth= , xlab='Wall area', ylab='heat
ing load')
```



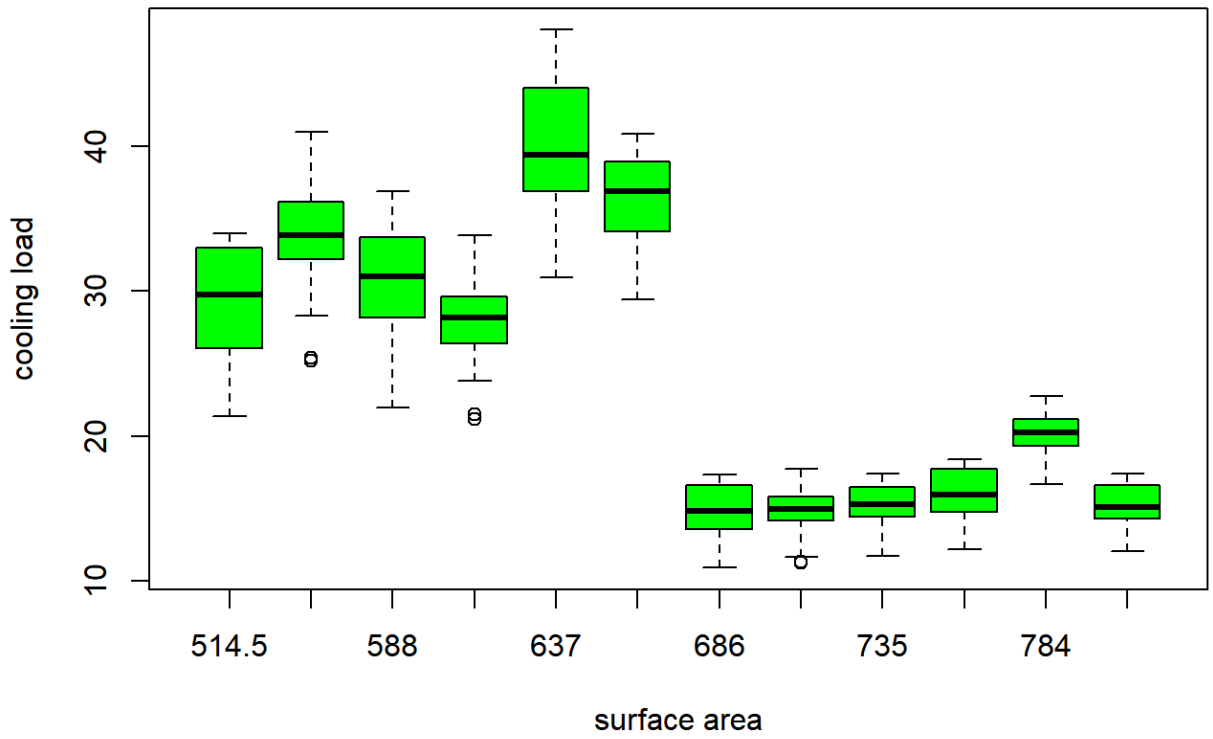

```
plot(qual_wall_area, cooling_load, col="green", varwidth= , xlab='wall area', ylab='co
oling load')
```




```
plot(qual_surface_area, heating_load, col="red", varwidth= , xlab='Surface area', ylab
='heating load')
```




```
plot(qual_surface_area, cooling_load, col="green", varwidth= , xlab='surface area', yl
ab='cooling load')
```




```
plot(relative_compactness, heating_load, col="red", varwidth= , xlab='relative compact
ness', ylab='heating load')
```

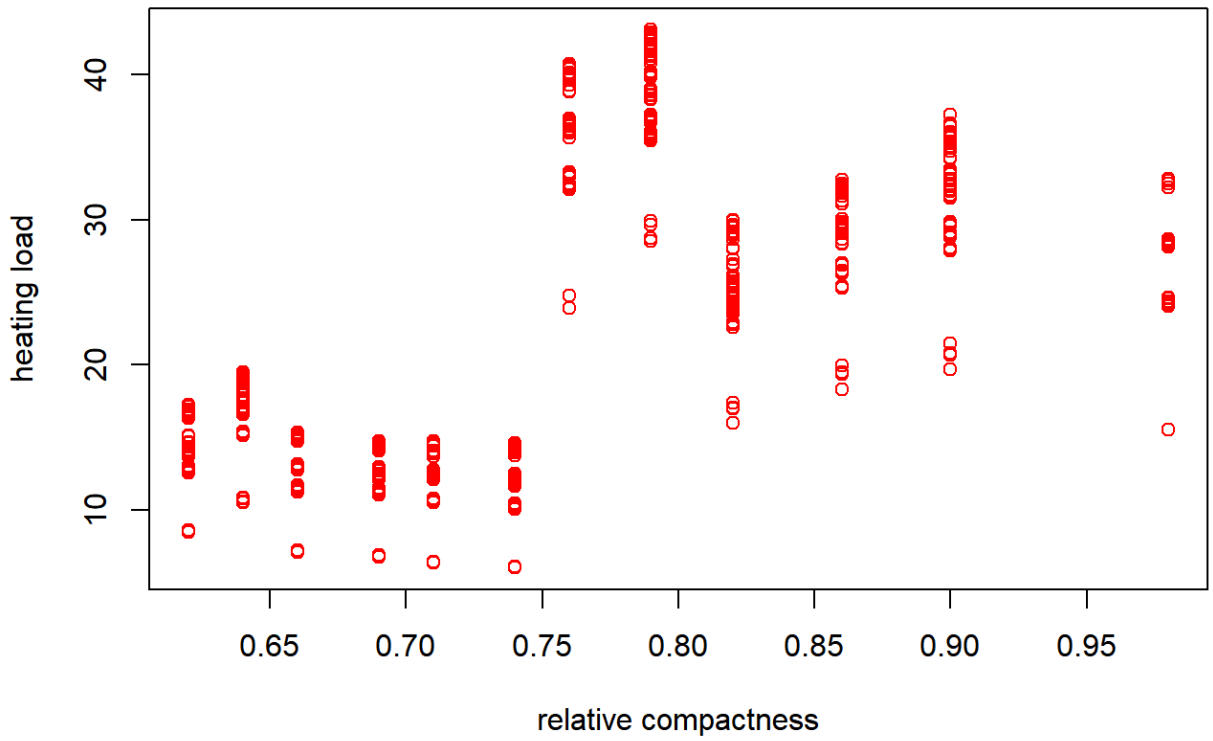
```
## Warning in plot.window(...): "varwidth" is not a graphical parameter
```

```
## Warning in plot.xy(xy, type, ...): "varwidth" is not a graphical parameter
```

```
## Warning in axis(side = side, at = at, labels = labels, ...): "varwidth" is
## not a graphical parameter

## Warning in axis(side = side, at = at, labels = labels, ...): "varwidth" is
## not a graphical parameter
```

```
## Warning in box(...): "varwidth" is not a graphical parameter
```

```
## Warning in title(...): "varwidth" is not a graphical parameter
```




```
plot(relative_compactness, cooling_load, col="green", varwidth= , xlab='relative compa
ctness', ylab='cooling load')
```

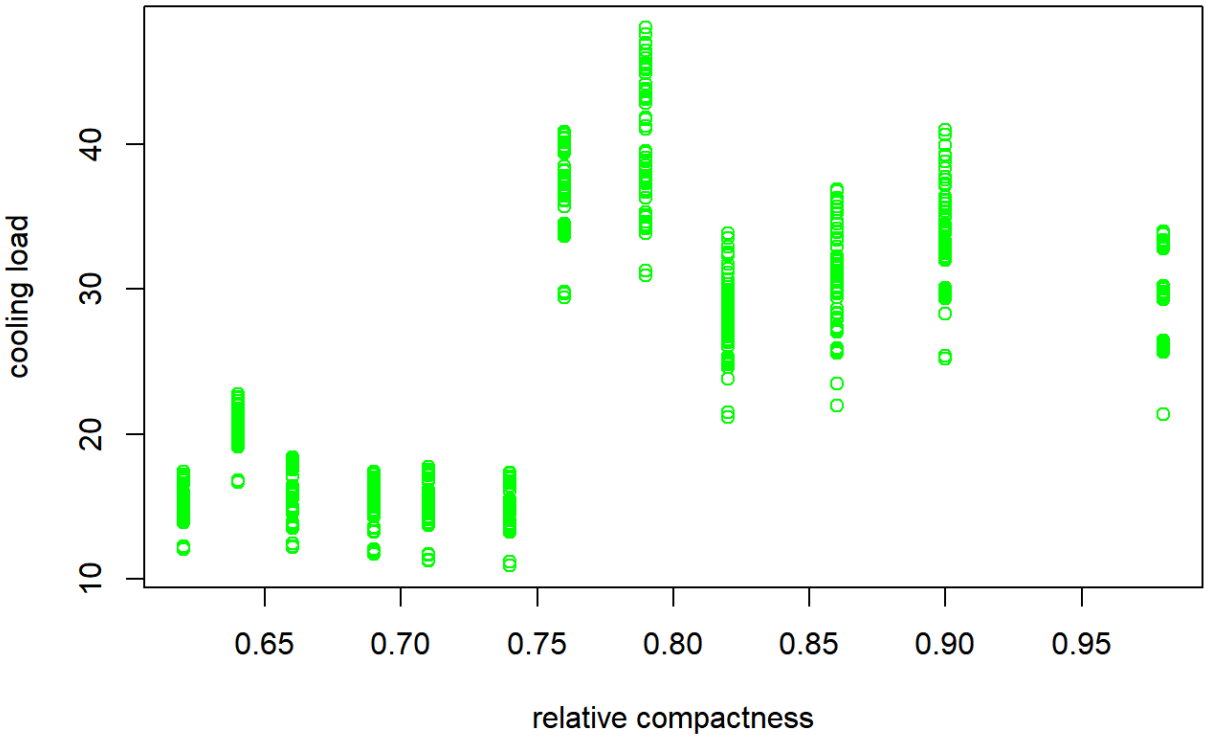
```
## Warning in plot.window(...): "varwidth" is not a graphical parameter
```

```
## Warning in plot.xy(xy, type, ...): "varwidth" is not a graphical parameter
```

```
## Warning in axis(side = side, at = at, labels = labels, ...): "varwidth" is
## not a graphical parameter

## Warning in axis(side = side, at = at, labels = labels, ...): "varwidth" is
## not a graphical parameter
```

```
## Warning in box(...): "varwidth" is not a graphical parameter
```

```
## Warning in title(...): "varwidth" is not a graphical parameter
```

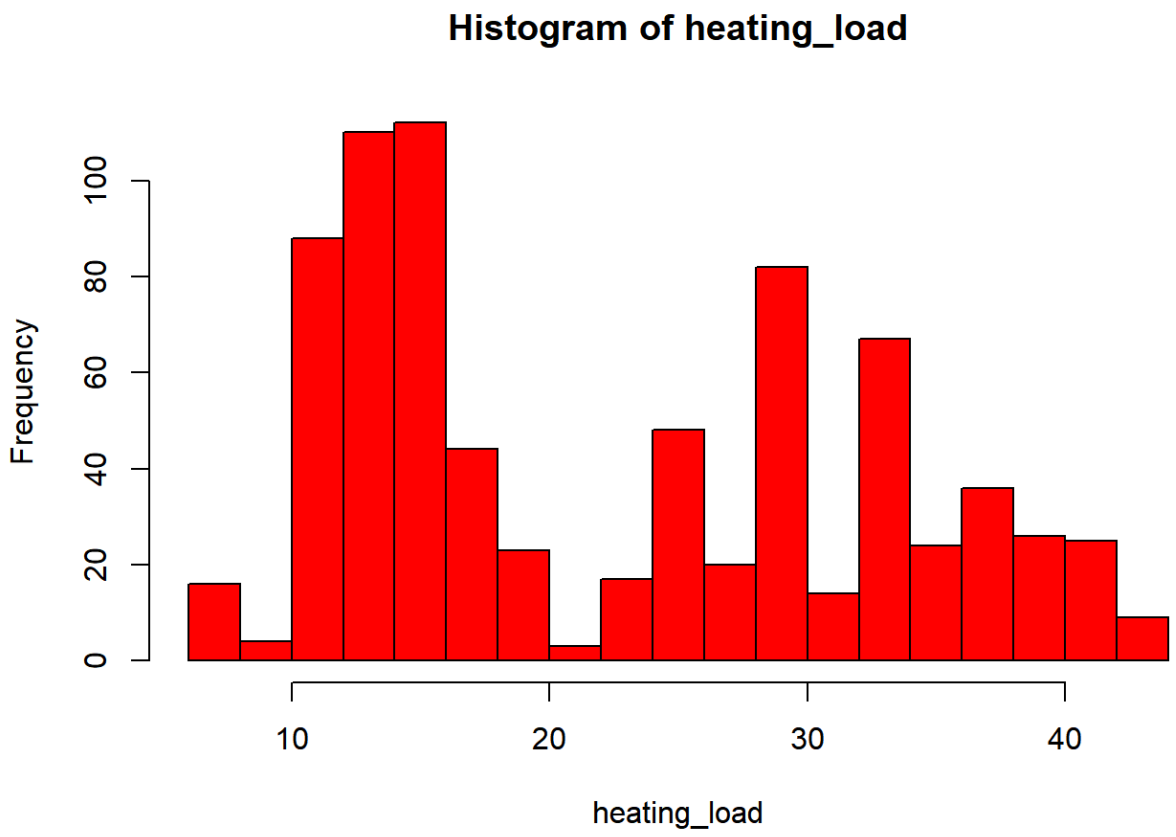


Above observations are obvious from the figures.

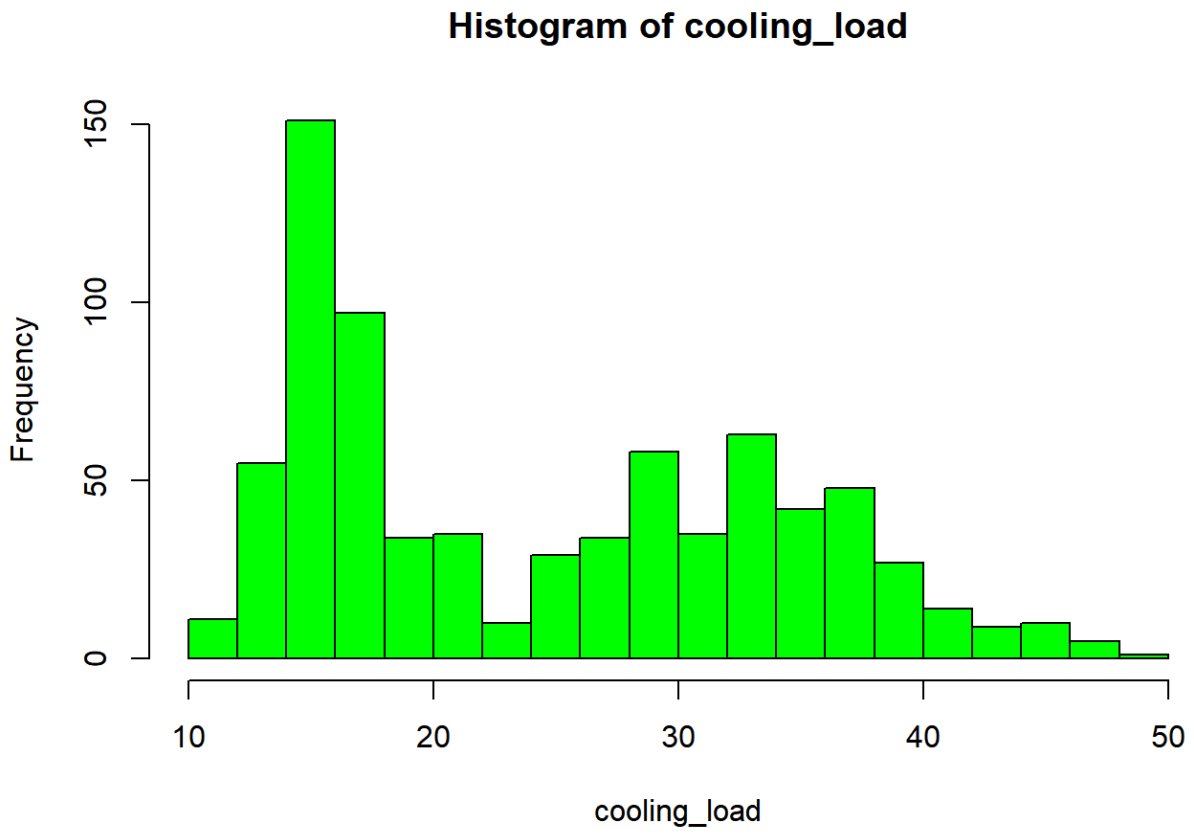
Histogram

Histograms shows the heating and cooling load distribution in the dataset.

```
hist(heating_load,col="red",breaks=15)
```

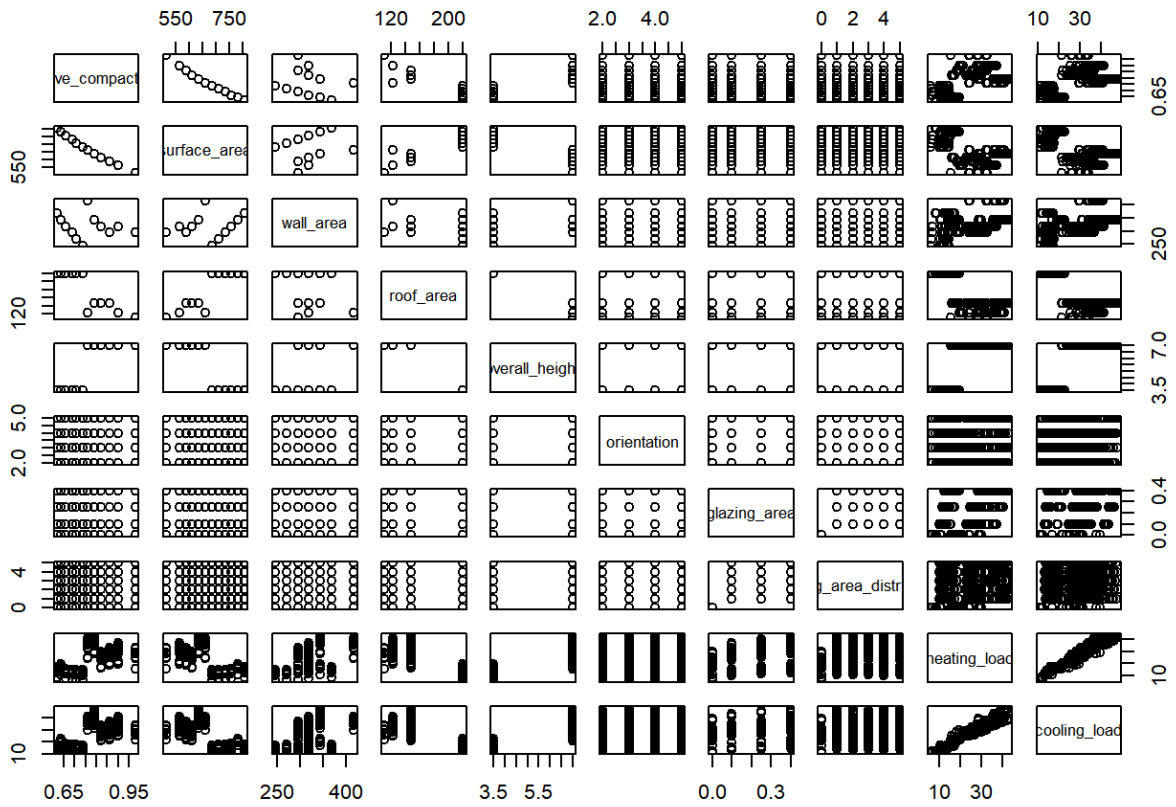



```
hist(cooling_load,col="green",breaks=15)
```

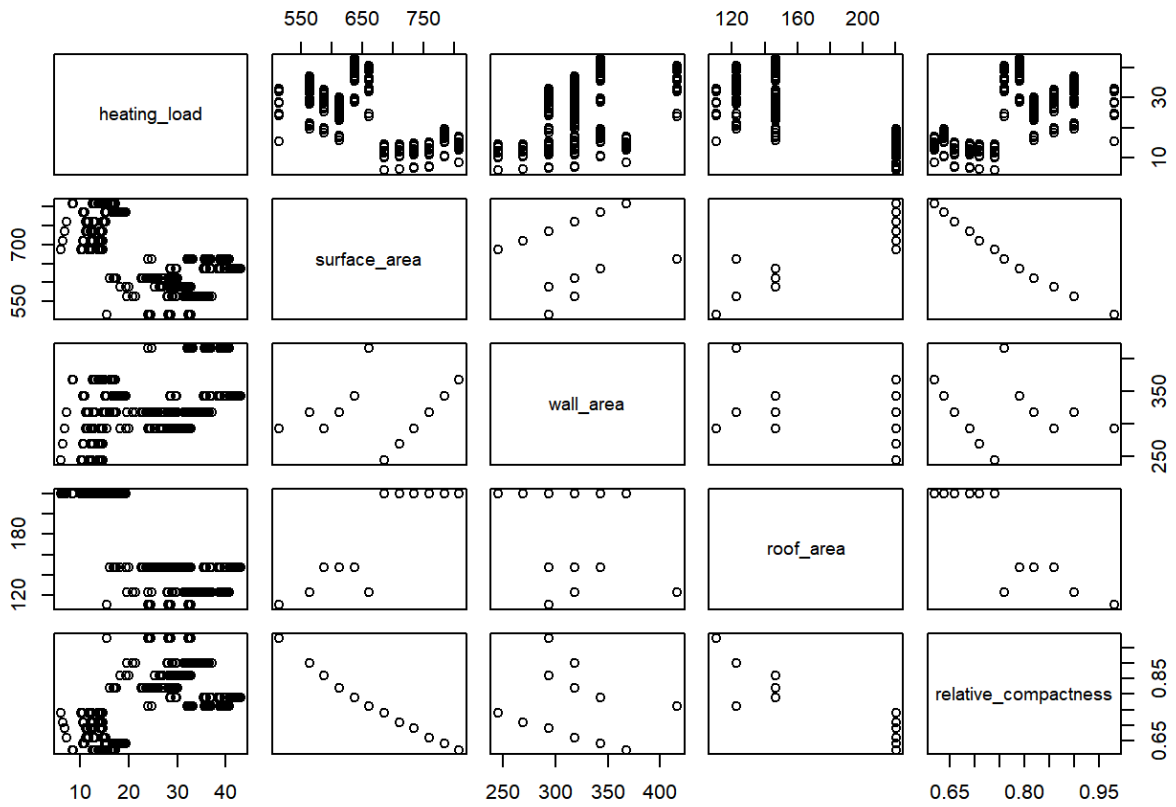


Scatter plot matrix

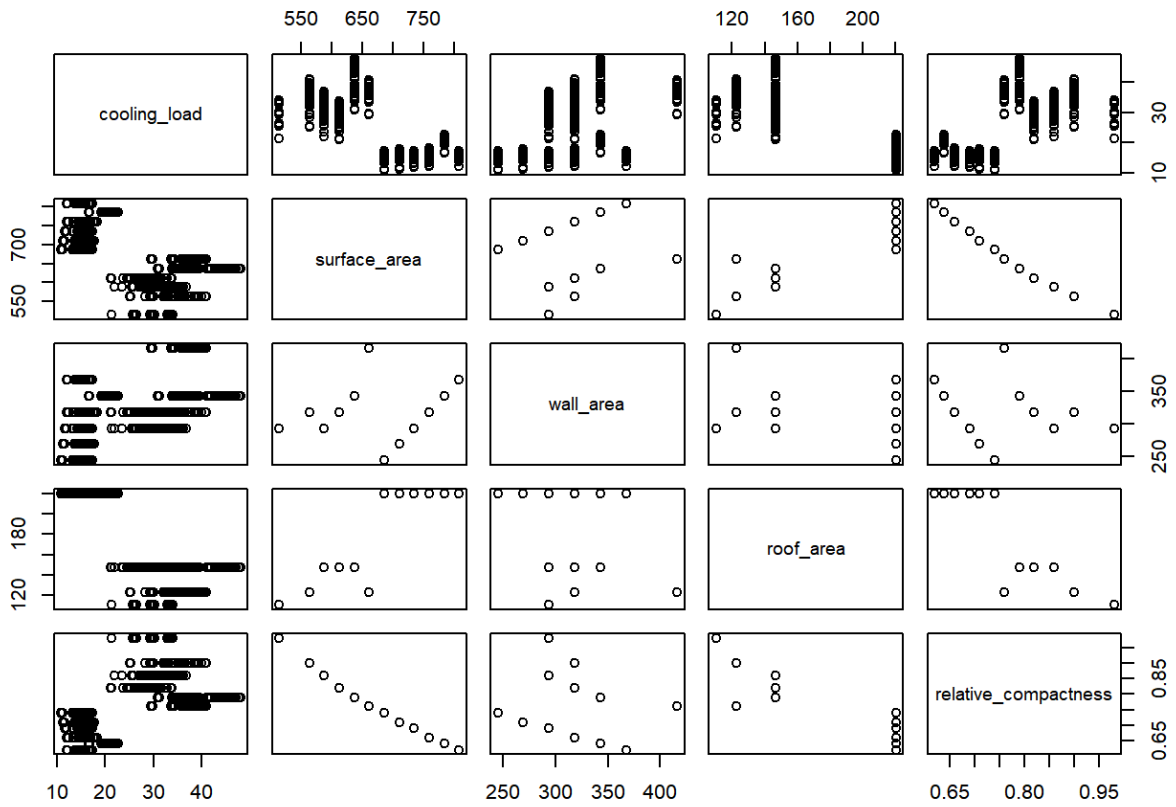
```
pairs(energy) #pairs() function creates a scatterplot matrix
```



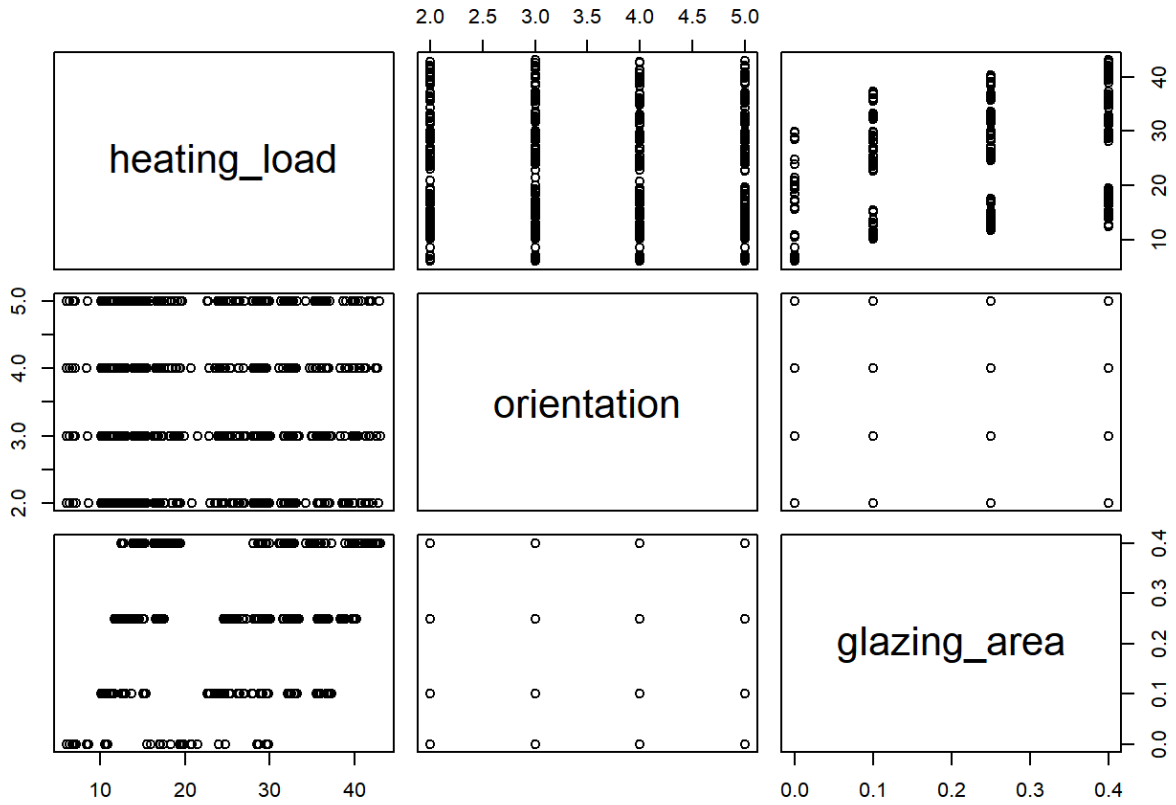

```
pairs( heating_load surface_area wall_area roof_area relative_compactness, en
ergy)
```



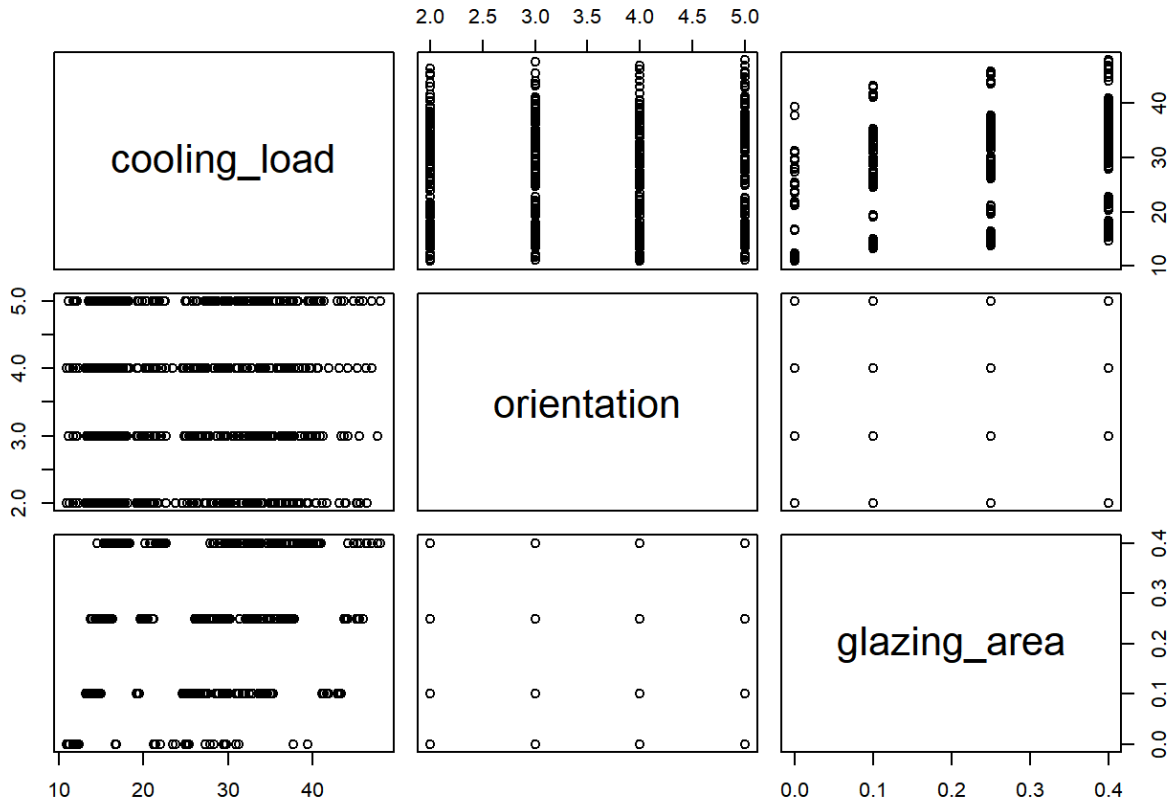

```
pairs( cooling_load surface_area wall_area roof_area relative_compactness, en
ergy)
```




```
pairs(  heating_load   orientation   glazing_area, energy)
```




```
pairs(  cooling_load   orientation   glazing_area, energy)
```



Co-Relation

```
cor(energy)
```

```
##                           relative_compactness  surface_area  wall_area
## relative_compactness              1.000000e 00 -9.919015e-01 -0.2037817
## surface_area                     -9.919015e-01  1.000000e 00  0.1955016
## wall_area                        -2.037817e-01  1.955016e-01  1.0000000
## roof_area                        -8.688234e-01  8.807195e-01 -0.2923165
## overall_height                    8.277473e-01 -8.581477e-01  0.2809757
## orientation                       0.000000e 00  0.000000e 00  0.0000000
## glazing_area                      7.617400e-20  4.664140e-20  0.0000000
## glazing_area_distribution         0.000000e 00  0.000000e 00  0.0000000
## heating_load                      6.222722e-01 -6.581202e-01  0.4556712
## cooling_load                      6.343391e-01 -6.729989e-01  0.4271170
##                              roof_area overall_height  orientation
## relative_compactness      -8.688234e-01      0.8277473  0.000000000
## surface_area               8.807195e-01     -0.8581477  0.000000000
## wall_area                 -2.923165e-01      0.2809757  0.000000000
## roof_area                  1.000000e 00     -0.9725122  0.000000000
## overall_height            -9.725122e-01      1.0000000  0.000000000
## orientation                0.000000e 00      0.0000000  1.000000000
## glazing_area              -1.197187e-19      0.0000000  0.000000000
## glazing_area_distribution  0.000000e 00      0.0000000  0.000000000
## heating_load              -8.618283e-01      0.8894307 -0.002586534
## cooling_load              -8.625466e-01      0.8957852  0.014289598
##                            glazing_area glazing_area_distribution
## relative_compactness       7.617400e-20                0.00000000
## surface_area               4.664140e-20                0.00000000
## wall_area                  0.000000e 00                0.00000000
## roof_area                 -1.197187e-19                0.00000000
## overall_height             0.000000e 00                0.00000000
## orientation                0.000000e 00                0.00000000
## glazing_area               1.000000e 00                0.21296422
## glazing_area_distribution  2.129642e-01                1.00000000
## heating_load               2.698410e-01                0.08736759
## cooling_load               2.075050e-01                0.05052512
##                           heating_load cooling_load
## relative_compactness       0.622272179   0.63433907
## surface_area              -0.658120227  -0.67299893
## wall_area                  0.455671157   0.42711700
## roof_area                 -0.861828253  -0.86254660
## overall_height             0.889430674   0.89578517
## orientation               -0.002586534   0.01428960
## glazing_area               0.269840996   0.20750499
## glazing_area_distribution  0.087367594   0.05052512
## heating_load               1.000000000   0.97586181
## cooling_load               0.975861813   1.00000000
```

summary of output variables

```
summary (heating_load)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    6.01   12.99   18.95   22.31   31.67   43.10
```

```
summary(cooling_load)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   10.90   15.62   22.08   24.59   33.13   48.03
```

Chapter

Linear regression

linear regression is applied with all the input variables.

```
lm.fit=lm(heating_load relative_compactness  surface_area   wall_area   orientation
roof_area   overall_height   glazing_area  glazing_area_distribution,data=energy)
lm.fit
```

```
##
##  all:
## lm(formula = heating_load   relative_compactness   surface_area
##     wall_area   orientation   roof_area   overall_height   glazing_area
##     glazing_area_distribution, data = energy)
##
##  oefficients:
##               ( ntercept)       relative_compactness
##                  84.01452                  -64.77399
##              surface_area                 wall_area
##                  -0.08729                   0.06081
##               orientation                roof_area
##                  -0.02333
##            overall_height              glazing_area
##                   4.16994                   19.93268
## glazing_area_distribution
##                   0.20377
```

```
summary(lm.fit)
```

```
##
##  all:
## lm(formula = heating_load   relative_compactness   surface_area
##     wall_area   orientation   roof_area   overall_height   glazing_area
##     glazing_area_distribution, data = energy)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -9.8965 -1.3196 -0.0252  1.3532  7.7052
##
##  oefficients: (1 not defined because of singularities)
##                              stimate Std.  rror t value Pr(  t )
## ( ntercept)                84.014521  19.033607   4.414 1.16e-05
## relative_compactness      -64.773991  10.289445  -6.295 5.19e-10
## surface_area               -0.087290   0.017075  -5.112 4.04e-07
## wall_area                   0.060813   0.006648   9.148  < 2e-16
## orientation                -0.023328   0.094705  -0.246  0.80550
## roof_area
## overall_height              4.169939   0.337990  12.337  < 2e-16
## glazing_area               19.932680   0.813986  24.488  < 2e-16
## glazing_area_distribution   0.203772   0.069918   2.914  0.00367
## ---
## Signif. codes:  0 '   ' 0.001 '  ' 0.01 ' ' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.934 on 760 degrees of freedom
## Multiple R-squared:  0.9162,  djusted R-squared:  0.9154
## F-statistic:  1187 on 7 and 760 DF,  p-value: < 2.2e-16
```

```
names(lm.fit)
```

```
##  [1] "coefficients"  "residuals"     "effects"       "rank"
##  [5] "fitted.values" "assign"        "qr"            "df.residual"
##  [9] "xlevels"       "call"          "terms"         "model"
```

```
coef(lm.fit)
```

```
##               ( ntercept)      relative_compactness
##                84.01452119               -64.77399149
##               surface_area                  wall_area
##                -0.08729027                 0.06081334
##                orientation                  roof_area
##                -0.02332813
##             overall_height              glazing_area
##                 4.16993881                19.93268018
## glazing_area_distribution
##                 0.20377177
```

```
confint(lm.fit) # confidence interval for the coefficient estimates
```

```
##                                 2.5         97.5
## ( ntercept)                46.64983259 121.37920978
## relative_compactness      -84.97310070 -44.57488228
## surface_area               -0.12081099  -0.05376956
## wall_area                   0.04776285   0.07386383
## orientation                -0.20924198   0.16258573
## roof_area
## overall_height              3.50643397   4.83344366
## glazing_area               18.33475226  21.53060810
## glazing_area_distribution   0.06651684   0.34102670
```

Above model provides good R^2 value and p-value is less than 0.1 . If we exclude the surface area and overall height following regression model is obtained.

```
lm.fit2=lm(heating_load relative_compactness   wall_area   orientation   roof_area   g
lazing_area  glazing_area_distribution,data=energy)
lm.fit2
```

```
##
##  all:
## lm(formula = heating_load   relative_compactness   wall_area
##     orientation   roof_area   glazing_area   glazing_area_distribution,
##     data = energy)
##
##  oefficients:
##               ( ntercept)       relative_compactness
##                 261.37820                 -145.94369
##                 wall_area               orientation
##                  -0.12559                   -0.02333
##                 roof_area              glazing_area
##                  -0.52496                   19.93268
## glazing_area_distribution
##                   0.20377
```

```
summary(lm.fit2)
```

```
##
##  all:
## lm(formula = heating_load   relative_compactness   wall_area
##     orientation   roof_area   glazing_area   glazing_area_distribution,
##     data = energy)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -9.8468 -2.0124 -0.2825  1.4104  9.1552
##
##  oefficients:
##                               stimate Std.  rror t value Pr(  t )
## ( ntercept)                261.37820   13.65752  19.138  < 2e-16
## relative_compactness      -145.94369    8.66165 -16.849  < 2e-16
## wall_area                   -0.12559    0.01087 -11.550  < 2e-16
## orientation                 -0.02333    0.10369  -0.225  0.82205
## roof_area                   -0.52496    0.02077 -25.279  < 2e-16
## glazing_area                19.93268    0.89119  22.366  < 2e-16
## glazing_area_distribution    0.20377    0.07655   2.662  0.00793
## ---
## Signif. codes:  0 '   ' 0.001 '  ' 0.01 ' ' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.213 on 761 degrees of freedom
## Multiple R-squared:  0.8994,  djusted R-squared:  0.8986
## F-statistic:  1134 on 6 and 761 DF,  p-value: < 2.2e-16
```

```
names(lm.fit2)
```

```
##  [1] "coefficients"  "residuals"     "effects"       "rank"
##  [5] "fitted.values" "assign"        "qr"            "df.residual"
##  [9] "xlevels"       "call"          "terms"         "model"
```

```
coef(lm.fit2)
```

```
##               ( ntercept)      relative_compactness
##              261.37819945             -145.94368872
##                 wall_area               orientation
##               -0.12558736                -0.02332812
##                 roof_area              glazing_area
##               -0.52495536                19.93268018
## glazing_area_distribution
##                 0.20377177
```

```
confint(lm.fit2)
```

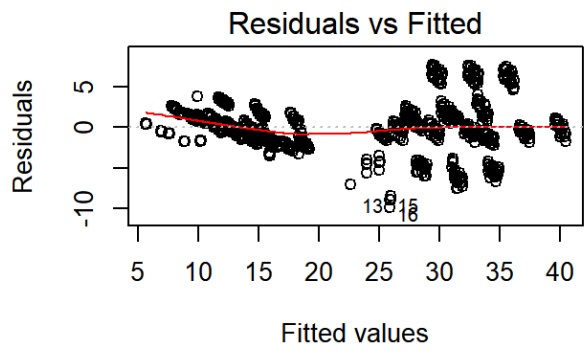
```
##                                  2.5         97.5
## ( ntercept)                234.56730908  288.1890898
## relative_compactness      -162.94725485 -128.9401226
## wall_area                   -0.14693212   -0.1042426
## orientation                 -0.22687603    0.1802198
## roof_area                   -0.56572229   -0.4841884
## glazing_area                18.18318773   21.6821726
## glazing_area_distribution    0.05349812    0.3540454
```

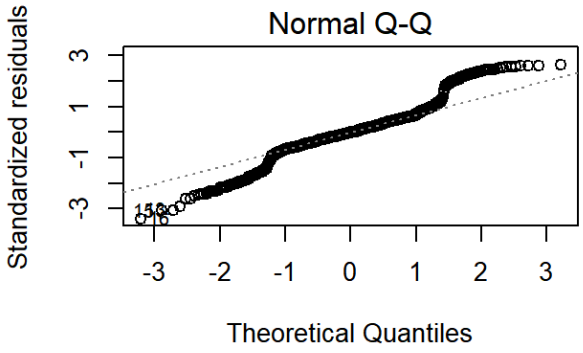
Second model provides lower R^2 value.

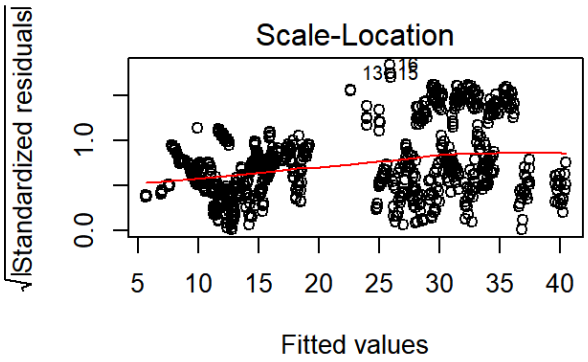
Residual plots

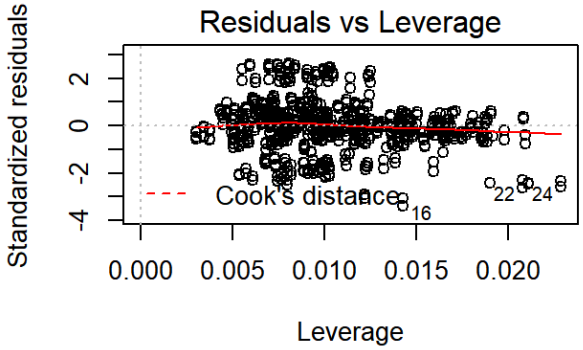
```
par(mfrow=c(2,2))
plot(lm.fit)
abline(lm.fit,lwd=5,col="red" )
```

```
## Warning in abline(lm.fit, lwd = 5, col = "red"): only using the first two
## of 9 regression coefficients
```

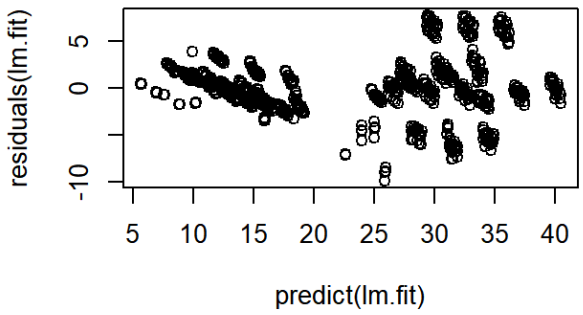


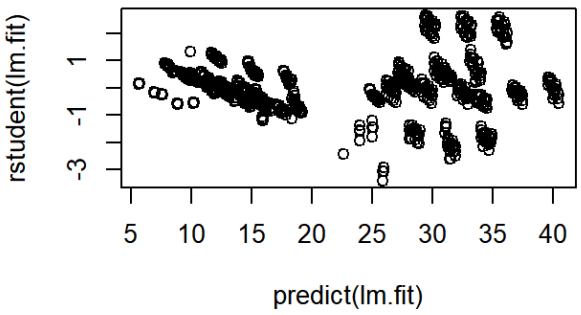







```
plot(predict(lm.fit), residuals(lm.fit))
plot(predict(lm.fit), rstudent(lm.fit))
```





There might be some evidence of heteroscedasticity as it appears a funnel shape in residual vs. fitted value plot. After transforming the output variables with taking logarithm.

```
lm.fit=lm(log(heating_load) relative_compactness  surface_area   wall_area   orientati
on   roof_area   overall_height   glazing_area  glazing_area_distribution,data=energy)
lm.fit
```

```
##
##  all:
## lm(formula = log(heating_load)   relative_compactness   surface_area
##     wall_area   orientation   roof_area   overall_height   glazing_area
##     glazing_area_distribution, data = energy)
##
##  oefficients:
##               ( ntercept)       relative_compactness
##                -0.5787874                  0.5623864
##              surface_area                  wall_area
##                 0.0014251                  0.0015991
##               orientation                  roof_area
##                -0.0008389
##            overall_height               glazing_area
##                 0.2665273                  1.0117131
## glazing_area_distribution
##                 0.0159934
```

```
summary(lm.fit)
```

```
##
##  all:
## lm(formula = log(heating_load)   relative_compactness   surface_area
##     wall_area   orientation   roof_area   overall_height   glazing_area
##     glazing_area_distribution, data = energy)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.35472 -0.05718  0.00176  0.06585  0.31204
##
##  oefficients: (1 not defined because of singularities)
##                              stimate Std.  rror t value Pr(  t )
## ( ntercept)               -0.5787874  0.7806251  -0.741   0.4587
## relative_compactness       0.5623864  0.4220009   1.333   0.1830
## surface_area               0.0014251  0.0007003   2.035   0.0422
## wall_area                  0.0015991  0.0002727   5.865 6.70e-09
## orientation               -0.0008389  0.0038841  -0.216   0.8290
## roof_area
## overall_height             0.2665273  0.0138620  19.227  < 2e-16
## glazing_area               1.0117131  0.0333840  30.305  < 2e-16
## glazing_area_distribution  0.0159934  0.0028675   5.577 3.39e-08
## ---
## Signif. codes:  0 '   ' 0.001 '  ' 0.01 ' ' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1203 on 760 degrees of freedom
## Multiple R-squared:  0.9368,  djusted R-squared:  0.9362
## F-statistic:  1610 on 7 and 760 DF,  p-value: < 2.2e-16
```

```
names(lm.fit)
```

```
##  [1] "coefficients"  "residuals"     "effects"       "rank"
##  [5] "fitted.values" "assign"        "qr"            "df.residual"
##  [9] "xlevels"       "call"          "terms"         "model"
```

```
coef(lm.fit)
```

```
##               ( ntercept)      relative_compactness
##             -0.5787873931              0.5623863990
##              surface_area                 wall_area
##              0.0014250946              0.0015991130
##               orientation                 roof_area
##             -0.0008389492
##            overall_height              glazing_area
##              0.2665273333              1.0117131308
## glazing_area_distribution
##              0.0159934328
```

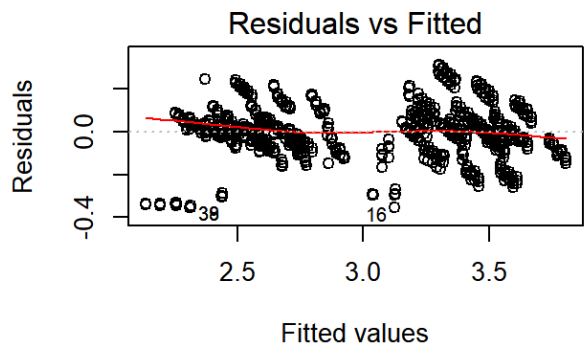
```
confint(lm.fit)
```

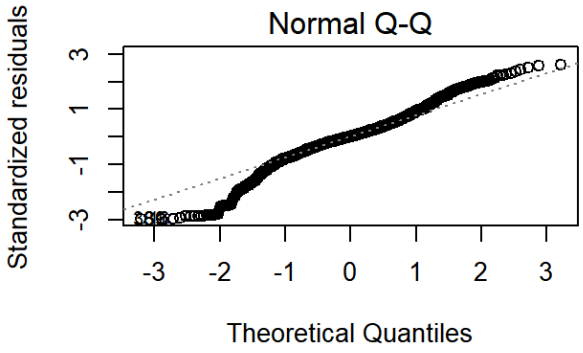
```
##                                   2.5        97.5
## ( ntercept)               -2.111225e 00 0.953650097
## relative_compactness      -2.660394e-01 1.390812204
## surface_area               5.030992e-05 0.002799879
## wall_area                  1.063873e-03 0.002134353
## orientation               -8.463832e-03 0.006785933
## roof_area
## overall_height             2.393150e-01 0.293739649
## glazing_area               9.461773e-01 1.077248928
## glazing_area_distribution  1.036420e-02 0.021622667
```

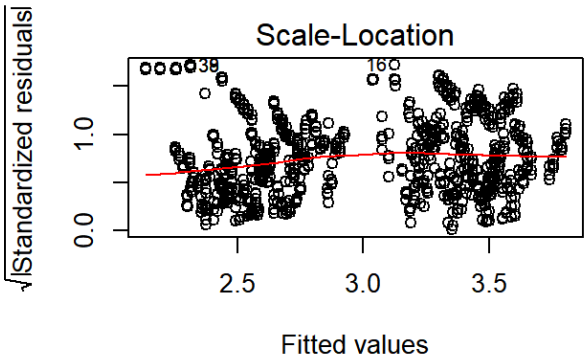
Now the residual plot is as follows:

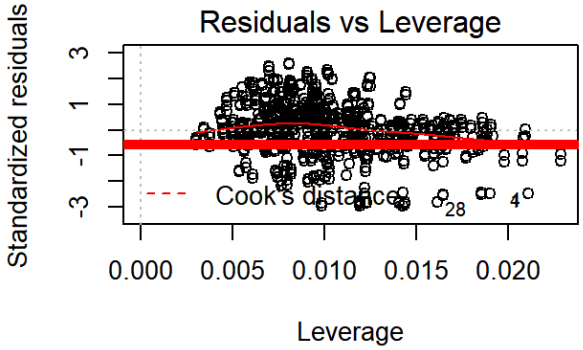
```
par(mfrow=c(2,2))
plot(lm.fit)
abline(lm.fit,lwd=5,col="red" )
```

```
## Warning in abline(lm.fit, lwd = 5, col = "red"): only using the first two
## of 9 regression coefficients
```

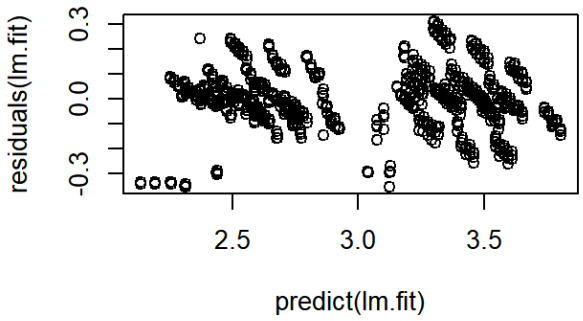


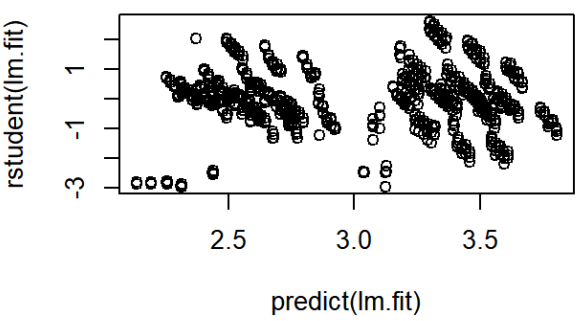







```
plot(predict(lm.fit), residuals(lm.fit))
plot(predict(lm.fit), rstudent(lm.fit))
```



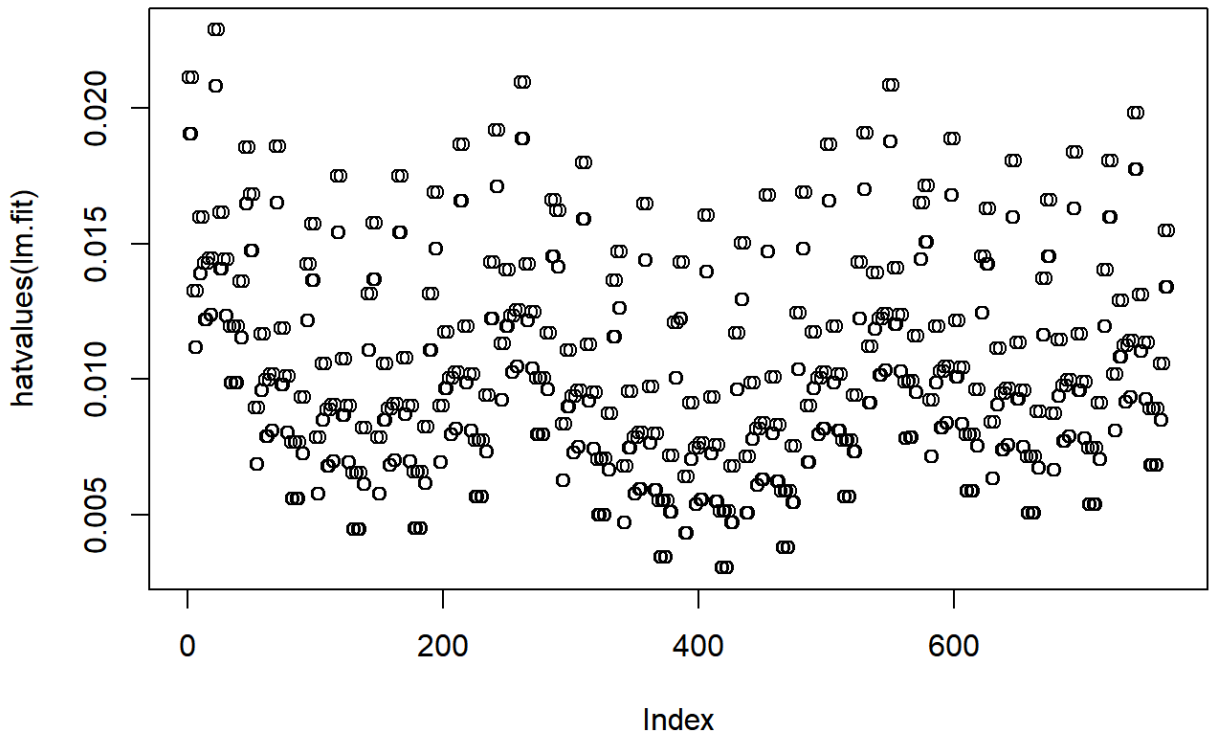


After transforming the funnel shape is removed and no evidence of heteroscedasticity is found.

Leverage statistics

Leverage statistics can be computed for any number of predictors using the hatvalues() function.

```
plot(hatvalues(lm.fit))
```



To get the observation with maximum leverage statistics following code can be used.

```
which.max(hatvalues(lm.fit))
```

```
## 21
## 21
```

Multiple Linear Regression

To see how the heating and cooling load differs for the entire volume of the buildings following model is built.

```
lm.fit3=lm((heating_load)   surface_area   overall_height ,data=energy)
summary(lm.fit3)
```

```
##
##  all:
## lm(formula = (heating_load)   surface_area   overall_height,
##     data = energy)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -16.2286  -2.1288   0.2706   2.2940   9.4924
##
##  oefficients:
##                                stimate Std.  rror t value Pr(  t )
## ( ntercept)                -16.106804   7.896794  -2.040   0.0417
## surface_area                 0.003855   0.010866   0.355   0.7229
## overall_height               1.905594   1.296516   1.470   0.1420
## surface_area:overall_height   0.007607   0.001884   4.037 5.96e-05
## ---
## Signif. codes:  0 '   ' 0.001 '  ' 0.01 ' ' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.088 on 764 degrees of freedom
## Multiple R-squared:  0.8365,  djusted R-squared:  0.8359
## F-statistic:  1303 on 3 and 764 DF,  p-value: < 2.2e-16
```

```
lm.fit4=lm(cooling_load   surface_area   overall_height ,data=energy)
summary(lm.fit4)
```

```
##
##  all:
## lm(formula = cooling_load   surface_area   overall_height, data = energy)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -12.8026  -2.2738  -0.1464   2.1960  12.7796
##
##  oefficients:
##                                stimate Std.  rror t value Pr(  t )
## ( ntercept)                -1.648144   7.306499  -0.226    0.822
## surface_area               -0.009708   0.010054  -0.966    0.335
## overall_height              0.488009   1.199599   0.407    0.684
## surface_area:overall_height  0.008896   0.001744   5.102 4.24e-07
## ---
## Signif. codes:  0 '   ' 0.001 '  ' 0.01 ' ' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.782 on 764 degrees of freedom
## Multiple R-squared:  0.8426,  djusted R-squared:  0.8419
## F-statistic:  1363 on 3 and 764 DF,  p-value: < 2.2e-16
```

Now including other variables in multiple regression in following section.

```
lm.fit3=lm((heating_load)   (surface_area   overall_height)   wall_area   orientation
  roof_area   glazing_area  glazing_area_distribution,data=energy)
summary(lm.fit3)
```

```
##
##  all:
## lm(formula = (heating_load)   (surface_area   overall_height)
##     wall_area   orientation   roof_area   glazing_area   glazing_area_distributio
n,
##     data = energy)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -9.6508 -1.3516 -0.0892  1.3344  7.4203
##
##  oefficients: (1 not defined because of singularities)
##                               stimate Std.  rror t value Pr(  t )
## ( ntercept)                 23.229969   7.207439   3.223 0.001322
## surface_area                -0.068893   0.010702  -6.437 2.15e-10
## overall_height              -3.834371   1.085209  -3.533 0.000435
## wall_area                    0.055352   0.005721   9.675  < 2e-16
## orientation                 -0.023328   0.092468  -0.252 0.800891
## roof_area
## glazing_area                19.932680   0.794765  25.080  < 2e-16
## glazing_area_distribution    0.203772   0.068267   2.985 0.002927
## surface_area:overall_height  0.012577   0.001417   8.875  < 2e-16
## ---
## Signif. codes:  0 '   ' 0.001 '  ' 0.01 ' ' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.865 on 760 degrees of freedom
## Multiple R-squared:  0.9201,  djusted R-squared:  0.9194
## F-statistic:  1250 on 7 and 760 DF,  p-value: < 2.2e-16
```

```
lm.fit4=lm((cooling_load)   (surface_area   overall_height)   relative_compactness  wa
ll_area   orientation   roof_area   glazing_area  glazing_area_distribution,data=energ
y)
summary(lm.fit4)
```

```
##
##  all:
## lm(formula = (cooling_load)   (surface_area   overall_height)
##     relative_compactness   wall_area   orientation   roof_area
##     glazing_area   glazing_area_distribution, data = energy)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -7.8481 -1.8945 -0.0708  1.3096 10.3911
##
##  oefficients: (1 not defined because of singularities)
##                                stimate Std.  rror t value Pr(  t )
## ( ntercept)                 -2.125e 02  5.242e 01  -4.053 5.58e-05
## surface_area                 7.433e-02  3.121e-02   2.382   0.0175
## overall_height              -1.913e 01  3.673e 00  -5.208 2.47e-07
## relative_compactness         1.966e 02  4.316e 01   4.555 6.10e-06
## wall_area                    3.216e-03  9.586e-03   0.336   0.7373
## orientation                  1.215e-01  1.007e-01   1.207   0.2279
## roof_area
## glazing_area                 1.472e 01  8.655e-01  17.004  < 2e-16
## glazing_area_distribution    4.070e-02  7.434e-02   0.547   0.5843
## surface_area:overall_height  3.899e-02  6.087e-03   6.405 2.64e-10
## ---
## Signif. codes:  0 '   ' 0.001 '  ' 0.01 ' ' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.12 on 759 degrees of freedom
## Multiple R-squared:  0.8936,  djusted R-squared:  0.8924
## F-statistic: 796.4 on 8 and 759 DF,  p-value: < 2.2e-16
```

Now with the transformed output variable following models are built.

```
lm.fit3=lm((heating_load)   (surface_area   overall_height)   relative_compactness wal
l_area   orientation   roof_area   glazing_area  glazing_area_distribution,data=energ
y)
summary(lm.fit3)
```

```
##
##  all:
## lm(formula = (heating_load)   (surface_area   overall_height)
##     relative_compactness   wall_area   orientation   roof_area
##     glazing_area   glazing_area_distribution, data = energy)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -8.6342 -1.6517 -0.1684  1.7149  6.5317
##
##  oefficients: (1 not defined because of singularities)
##                                stimate Std.  rror t value Pr(  t )
## ( ntercept)                 -3.782e 02  4.586e 01  -8.246 7.18e-16
## surface_area                 1.553e-01  2.730e-02   5.689 1.83e-08
## overall_height              -3.077e 01  3.213e 00  -9.575  < 2e-16
## relative_compactness         3.342e 02  3.775e 01   8.853  < 2e-16
## wall_area                   -1.065e-03  8.386e-03  -0.127   0.8990
## orientation                 -2.333e-02  8.809e-02  -0.265   0.7912
## roof_area
## glazing_area                 1.993e 01  7.572e-01  26.326  < 2e-16
## glazing_area_distribution    2.038e-01  6.504e-02   3.133   0.0018
## surface_area:overall_height  5.818e-02  5.325e-03  10.925  < 2e-16
## ---
## Signif. codes:  0 '   ' 0.001 '  ' 0.01 ' ' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.729 on 759 degrees of freedom
## Multiple R-squared:  0.9276,  djusted R-squared:  0.9268
## F-statistic:  1215 on 8 and 759 DF,  p-value: < 2.2e-16
```

```
lm.fit4=lm(log(cooling_load)   (surface_area   overall_height)   relative_compactness
wall_area   orientation   roof_area   glazing_area  glazing_area_distribution,data=ene
rgy)
summary(lm.fit4)
```

```
##
##  all:
## lm(formula = log(cooling_load)   (surface_area   overall_height)
##     relative_compactness   wall_area   orientation   roof_area
##     glazing_area   glazing_area_distribution, data = energy)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.26777 -0.06733 -0.00730  0.03973  0.31731
##
##  oefficients: (1 not defined because of singularities)
##                                stimate Std.  rror t value Pr(  t )
## ( ntercept)                 -1.4704616  1.8766052  -0.784  0.43353
## surface_area                 0.0012648  0.0011172   1.132  0.25793
## overall_height              -0.1378344  0.1314929  -1.048  0.29487
## relative_compactness         2.7828969  1.5448679   1.801  0.07204 .
## wall_area                    0.0007391  0.0003431   2.154  0.03155
## orientation                  0.0045359  0.0036048   1.258  0.20866
## roof_area
## glazing_area                 0.6341835  0.0309830  20.469  < 2e-16
## glazing_area_distribution    0.0030144  0.0026613   1.133  0.25770
## surface_area:overall_height  0.0005702  0.0002179   2.617  0.00905
## ---
## Signif. codes:  0 '   ' 0.001 '  ' 0.01 ' ' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1117 on 759 degrees of freedom
## Multiple R-squared:   0.92,   djusted R-squared:  0.9191
## F-statistic:  1091 on 8 and 759 DF,  p-value: < 2.2e-16
```

R^2 value has been increased.

Chapter 4

Here K-NN classification is applied on the dataset though the problem is not a classification problem

```
library(class)
train=(glazing_area 0)
energy.train=energy[train,]
energy.test=energy[ train,]
hload.train=heating_load[train]
hload.test=heating_load[ train]
dim(energy.train)
```

```
## [1] 720  10
```

```
dim(energy.test)
```

```
## [1] 48 10
```

```
energy.train=cbind(surface_area, orientation, overall_height, glazing_area)[train,]
energy.test=cbind(surface_area, orientation, overall_height, glazing_area)[ train,]
set.seed(1)
knn.pred=knn(energy.train,energy.test,hload.train,k=1)
mean(knn.pred==hload.test)
```

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

The output doesn't give any satisfactory result here.

Chapter

Different types of resampling technique is applied in this section.

```
library( S R)
```

```
## Warning: package ' S R' was built under R version 3.3.3
```

```
library(boot)
```

Validation Set Approach

```
set.seed(1)
train=sample(576,192)
lm.fit5=lm(heating_load surface_area,data=energy,subset=train)
mean((heating_load-predict(lm.fit5,energy))[-train] 2)
```

```
## [1] 58.72403
```

```
lm.fit6=lm(heating_load  poly(surface_area,overall_height),data=energy,subset=train)
mean((heating_load-predict(lm.fit6,energy))[-train] 2)
```

```
## [1] 19.13367
```

```
lm.fit7=lm(heating_load  poly(surface_area,overall_height, wall_area, relative_compact
ness, glazing_area),data=energy,subset=train)
mean((heating_load-predict(lm.fit7,energy))[-train] 2)
```

```
## [1] 9.108138
```

Leave-One-Out Cross-Validation

```
glm.fit=glm(heating_load surface_area,data=energy)
coef(glm.fit)
```

```
##  ( ntercept) surface_area
##  72.94538243  -0.07538716
```

```
lm.fit=lm(heating_load surface_area,data=energy)
coef(lm.fit)
```

```
##  ( ntercept) surface_area
##  72.94538243  -0.07538716
```

```
cv.err=cv.glm(energy,glm.fit)
cv.err delta
```

```
## [1] 57.86904 57.86889
```

```
cv.error=rep(0,5)
for (i in 1:5)
     glm.fit=glm(heating_load poly(surface_area,i),data=energy)


cv.error
```

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

k-Fold Cross-Validation

```
set.seed(17)
cv.error.10=rep(0,10)
for (i in 1:10)
 glm.fit=glm(heating_load poly(surface_area,i),data=energy)
 cv.error.10[i]=cv.glm(energy,glm.fit, =10) delta[1]


cv.error.10
```

```
##  [1] 57.88547 56.30551 43.62535 41.93680 35.27860 29.64350 21.60262
##  [8] 21.77655 12.57868 11.32085
```

The Bootstrap

```
alpha.fn=function(data,index)
  =data surface_area[index]
  =data heating_load[index]
 return((var( )-cov( , ))/(var( ) var( )-2 cov( , )))

alpha.fn(energy,1:100)
```

```
## [1] 0.06001243
```

```
set.seed(1)
alpha.fn(energy,sample(100,100,replace= ))
```

```
## [1] 0.07135656
```

```
boot(energy,alpha.fn,R=1000)
```

```
##
##  RD   R     P R M  R       S R P
##
##
##  all:
## boot(data = energy, statistic = alpha.fn, R = 1000)
##
##
##  ootstrap Statistics :
##       original       bias    std. error
## t1  0.07604519 -2.64864e-05 0.002145546
```

Estimating the Accuracy of a Linear Regression Model

```
boot.fn=function(data,index)
 return(coef(lm(heating_load surface_area,data=energy,subset=index)))

boot.fn(energy,1:576)
```

```
##  ( ntercept) surface_area
##  69.86780807  -0.07231806
```

```
set.seed(1)
boot.fn(energy,sample(350,350,replace= ))
```

```
##  ( ntercept) surface_area
##  71.06448215  -0.07568437
```

```
boot(energy,boot.fn,1000)
```

```
##
##  RD   R     P R M  R       S R P
##
##
##  all:
## boot(data = energy, statistic = boot.fn, R = 1000)
##
##
##  ootstrap Statistics :
##        original        bias    std. error
## t1  72.94538243 -5.175956e-03 1.640272340
## t2  -0.07538716  4.790684e-06 0.002288614
```

```
summary(lm(heating_load surface_area,data=energy)) coef
```

```
##                  stimate  Std.  rror   t value      Pr(  t )
## ( ntercept)  72.94538243 2.111061837  34.55388 1.972297e-158
## surface_area -0.07538716 0.003116179 -24.19217  1.686907e-96
```

```
boot.fn=function(data,index)
 coefficients(lm(heating_load surface_area  (overall_height  surface_area),data=energ
y,subset=index))

set.seed(1)
boot(energy,boot.fn,1000)
```

```
##
##  RD   R     P R M  R       S R P
##
##
##  all:
## boot(data = energy, statistic = boot.fn, R = 1000)
##
##
##  ootstrap Statistics :
##        original        bias     std. error
## t1  -4.92774605  8.318531e-02 2.1009693541
## t2  -0.01177671 -7.376136e-05 0.0021485688
## t3   0.01035435 -1.059017e-05 0.0002558188
```

```
summary(lm(heating_load surface_area  (overall_height  surface_area),data=energy)) coe
f
```

```
##                                     stimate   Std.  rror   t value
## ( ntercept)                      -4.92774605 2.1250868965 -2.318844
## surface_area                     -0.01177671 0.0022280476 -5.285663
##  (overall_height   surface_area)  0.01035435 0.0002387967 43.360518
##                                      Pr(  t )
## ( ntercept)                       2.066611e-02
## surface_area                      1.635358e-07
##  (overall_height   surface_area) 2.806047e-208
```

Chapter

Necessary libraries for this section are included.

```
library( S R)
library(leaps)
```

```
## Warning: package 'leaps' was built under R version 3.3.3
```

```
library(pls)
```

```
## Warning: package 'pls' was built under R version 3.3.3
```

```
##
##  ttaching package: 'pls'
```

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

```
library(glmnet)
```

```
## Warning: package 'glmnet' was built under R version 3.3.3
```

```
##  oading required package: Matrix
```

```
##  oading required package: foreach
```

```
## Warning: package 'foreach' was built under R version 3.3.3
```

```
##  oaded glmnet 2.0-16
```

Forward and Backward Stepwise Selection

```
regfit.full=regsubsets(heating_load .,data=energy-cooling_load)
```

```
## Warning in leaps.setup(x, y, wt = wt, nbest = nbest, nvmax = nvmax,
## force.in = force.in, : 1 linear dependencies found
```

```
regfit.fwd=regsubsets(heating_load .,data=energy-cooling_load,nvmax=19,method="forwar
d")
```

```
## Warning in leaps.setup(x, y, wt = wt, nbest = nbest, nvmax = nvmax,
## force.in = force.in, : 1 linear dependencies found
```

```
## Warning in leaps.setup(x, y, wt = wt, nbest = nbest, nvmax = nvmax,
## force.in = force.in, : nvmax reduced to 8
```

```
## Warning in rval lopt[] <- rval vorder[rval lopt]: number of items to
## replace is not a multiple of replacement length
```

```
summary(regfit.fwd)
```

```
## Subset selection object
##  all: regsubsets.formula(heating_load   ., data = energy - cooling_load,
##     nvmax = 19, method = "forward")
## 9  ariables  (and intercept)
##                           Forced in Forced out
## relative_compactness          F  S       F  S
## surface_area                  F  S       F  S
## wall_area                     F  S       F  S
## roof_area                     F  S       F  S
## overall_height                F  S       F  S
## orientation                   F  S       F  S
## glazing_area                  F  S       F  S
## glazing_area_distribution     F  S       F  S
## cooling_load                  F  S       F  S
## 1 subsets of each size up to 8
## Selection  lgorithm: forward
##          relative_compactness surface_area wall_area roof_area
## 1  ( 1 ) " "                  " "          " "       " "
## 2  ( 1 ) " "                  " "          " "       " "
## 3  ( 1 ) " "                  " "          " "       " "
## 4  ( 1 ) " "                  " "          " "       " "
## 5  ( 1 ) " "                  " "          " "       " "
## 6  ( 1 ) " "                  " "          " "       " "
## 7  ( 1 ) " "                  " "          " "       " "
## 8  ( 1 ) " "                  " "          " "       " "
##          overall_height orientation glazing_area glazing_area_distribution
## 1  ( 1 ) " "            " "         " "          " "
## 2  ( 1 ) " "            " "         " "          " "
## 3  ( 1 ) " "            " "         " "          " "
## 4  ( 1 ) " "            " "         " "          " "
## 5  ( 1 ) " "            " "         " "          " "
## 6  ( 1 ) " "            " "         " "          " "
## 7  ( 1 ) " "            " "         " "          " "
## 8  ( 1 ) " "            " "         " "          " "
##          cooling_load
## 1  ( 1 ) " "
## 2  ( 1 ) " "
## 3  ( 1 ) " "
## 4  ( 1 ) " "
## 5  ( 1 ) " "
## 6  ( 1 ) " "
## 7  ( 1 ) " "
## 8  ( 1 ) " "
```

```
regfit.bwd=regsubsets(heating_load .,data=energy-cooling_load,nvmax=19,method="backwar
d")
```

```
## Warning in leaps.setup(x, y, wt = wt, nbest = nbest, nvmax = nvmax,
## force.in = force.in, : 1 linear dependencies found
```

```
## Warning in leaps.setup(x, y, wt = wt, nbest = nbest, nvmax = nvmax,
## force.in = force.in, : nvmax reduced to 8
```

```
## Warning in rval lopt[] <- rval vorder[rval lopt]: number of items to
## replace is not a multiple of replacement length
```

```
summary(regfit.bwd)
```

```
## Subset selection object
##  all: regsubsets.formula(heating_load   ., data = energy - cooling_load,
##     nvmax = 19, method = "backward")
## 9  ariables  (and intercept)
##                           Forced in Forced out
## relative_compactness          F  S       F  S
## surface_area                  F  S       F  S
## wall_area                     F  S       F  S
## roof_area                     F  S       F  S
## overall_height                F  S       F  S
## orientation                   F  S       F  S
## glazing_area                  F  S       F  S
## glazing_area_distribution     F  S       F  S
## cooling_load                  F  S       F  S
## 1 subsets of each size up to 8
## Selection  lgorithm: backward
##          relative_compactness surface_area wall_area roof_area
## 1  ( 1 ) " "                  " "          " "       " "
## 2  ( 1 ) " "                  " "          " "       " "
## 3  ( 1 ) " "                  " "          " "       " "
## 4  ( 1 ) " "                  " "          " "       " "
## 5  ( 1 ) " "                  " "          " "       " "
## 6  ( 1 ) " "                  " "          " "       " "
## 7  ( 1 ) " "                  " "          " "       " "
## 8  ( 1 ) " "                  " "          " "       " "
##          overall_height orientation glazing_area glazing_area_distribution
## 1  ( 1 ) " "            " "         " "          " "
## 2  ( 1 ) " "            " "         " "          " "
## 3  ( 1 ) " "            " "         " "          " "
## 4  ( 1 ) " "            " "         " "          " "
## 5  ( 1 ) " "            " "         " "          " "
## 6  ( 1 ) " "            " "         " "          " "
## 7  ( 1 ) " "            " "         " "          " "
## 8  ( 1 ) " "            " "         " "          " "
##          cooling_load
## 1  ( 1 ) " "
## 2  ( 1 ) " "
## 3  ( 1 ) " "
## 4  ( 1 ) " "
## 5  ( 1 ) " "
## 6  ( 1 ) " "
## 7  ( 1 ) " "
## 8  ( 1 ) " "
```

```
coef(regfit.full,7)
```

```
##               ( ntercept)      relative_compactness
##                7.56365691               -10.11968399
##              surface_area                  wall_area
##               -0.01810849                 0.02706675
##             overall_height               orientation
##                1.10842511                -0.11175627
##              glazing_area glazing_area_distribution
##                9.21040173                 0.17387147
```

```
coef(regfit.fwd,7)
```

```
##               ( ntercept)      relative_compactness
##                7.56365691               -10.11968399
##              surface_area                  wall_area
##               -0.01810849                 0.02706675
##             overall_height               orientation
##                1.10842511                -0.11175627
##              glazing_area glazing_area_distribution
##                9.21040173                 0.17387147
```

```
coef(regfit.bwd,7)
```

```
##               ( ntercept)      relative_compactness
##                7.56365691               -10.11968399
##              surface_area                  wall_area
##               -0.01810849                 0.02706675
##             overall_height               orientation
##                1.10842511                -0.11175627
##              glazing_area glazing_area_distribution
##                9.21040173                 0.17387147
```

Ridge Regression

```
x=model.matrix(heating_load .-cooling_load,energy)[,-1]
y=energy heating_load
train=sample(1:nrow(x), nrow(x)/2)
test=(-train)
y.test=y[test]

grid=10 seq(10,-2,length=100)
ridge.mod=glmnet(x,y,alpha=0,lambda=grid)
dim(coef(ridge.mod))
```

```
## [1]   9 100
```

```
ridge.mod lambda[50]
```

```
## [1] 11497.57
```

```
coef(ridge.mod)[,50]
```

```
##               ( ntercept)      relative_compactness
##              2.228309e 01              5.190043e-02
##              surface_area                 wall_area
##             -6.590888e-05              9.232665e-05
##                 roof_area            overall_height
##             -1.683826e-04              4.481734e-03
##               orientation              glazing_area
##             -2.044136e-05              1.790797e-02
## glazing_area_distribution
##              4.977703e-04
```

```
sqrt(sum(coef(ridge.mod)[-1,50] 2))
```

```
## [1] 0.05508834
```

```
ridge.mod lambda[60]
```

```
## [1] 705.4802
```

```
coef(ridge.mod)[,60]
```

```
##               ( ntercept)      relative_compactness
##             21.9176708609              0.7991612269
##              surface_area                 wall_area
##             -0.0010174328              0.0014740659
##                 roof_area            overall_height
##             -0.0026225902              0.0699658480
##               orientation              glazing_area
##             -0.0003287366              0.2877307265
## glazing_area_distribution
##              0.0079355567
```

```
sqrt(sum(coef(ridge.mod)[-1,60] 2))
```

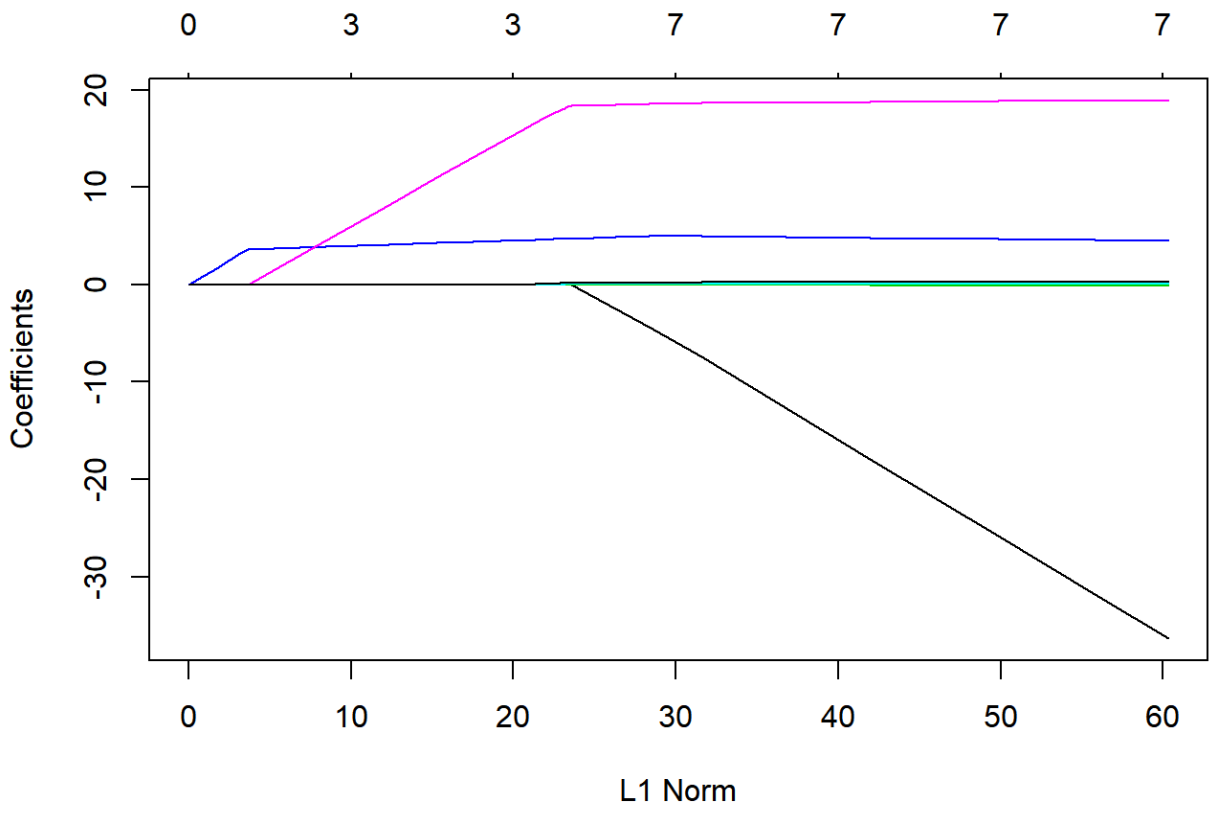
```
## [1] 0.8523004
```

```
predict(ridge.mod,s=50,type="coefficients")[1:7,]
```

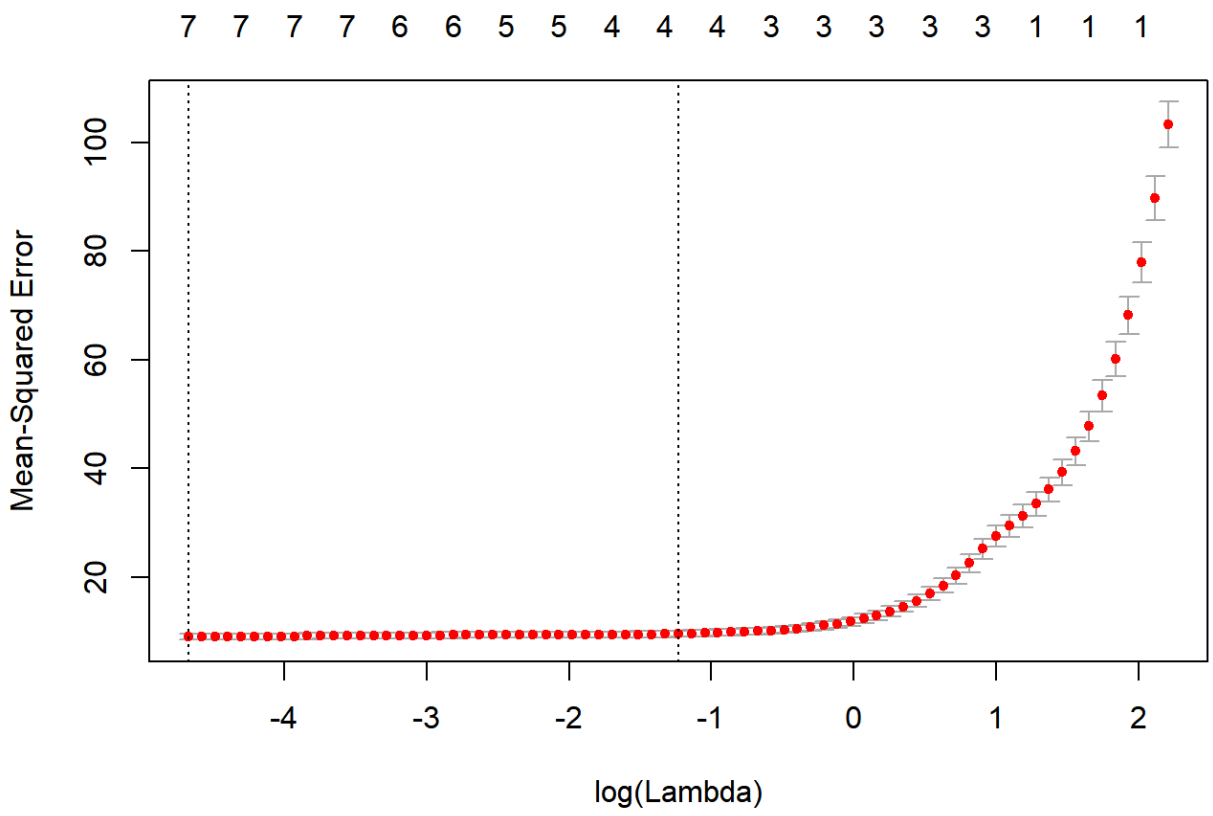
```
##          ( ntercept) relative_compactness         surface_area
##         17.500608897          6.197211495         -0.008160936
##            wall_area            roof_area       overall_height
##          0.016558894         -0.023245027          0.638116407
##          orientation
##         -0.003967923
```

The Lasso

```
lasso.mod=glmnet(x[train,],y[train],alpha=1,lambda=grid)
plot(lasso.mod)
```




```
set.seed(1)
cv.out=cv.glmnet(x[train,],y[train],alpha=1)
plot(cv.out)
```




```
bestlam=cv.out lambda.min
lasso.pred=predict(lasso.mod,s=bestlam,newx=x[test,])
mean((lasso.pred-y.test) 2)
```

```
## [1] 8.686998
```

```
out=glmnet(x,y,alpha=1,lambda=grid)
lasso.coef=predict(out,type="coefficients",s=bestlam)[1:7,]
lasso.coef
```

```
##          ( ntercept) relative_compactness         surface_area
##         4.424064e 01        -4.303555e 01        -1.773279e-05
##            wall_area            roof_area       overall_height
##         0.000000e 00        -1.049950e-01         4.636637e 00
##          orientation
##        -1.438385e-02
```

```
lasso.coef[lasso.coef =0]
```

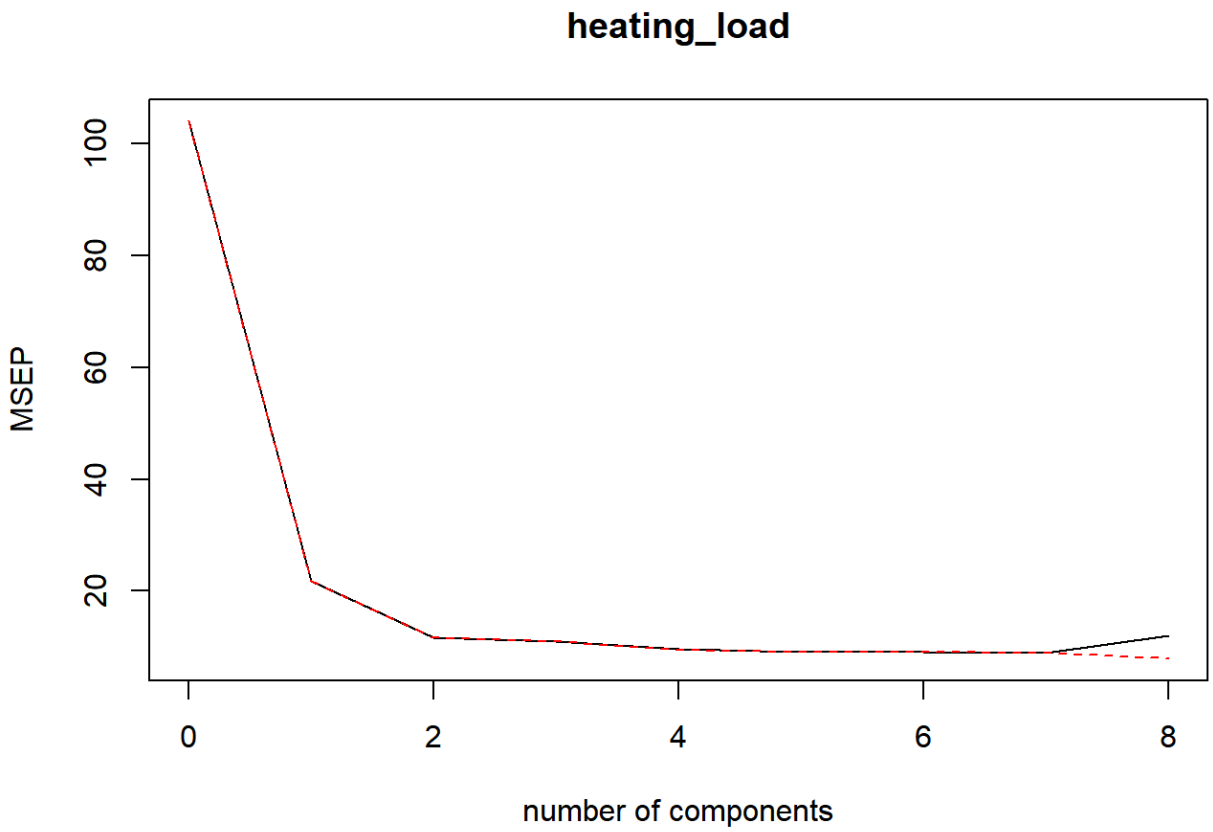
```
##          ( ntercept) relative_compactness         surface_area
##         4.424064e 01        -4.303555e 01        -1.773279e-05
##            roof_area       overall_height          orientation
##        -1.049950e-01         4.636637e 00        -1.438385e-02
```

Partial Least Squares

```
set.seed(1)
pls.fit=plsr(heating_load .-cooling_load, data=energy,subset=train,scale= R  , validat
ion="  ")
summary(pls.fit)
```

```
## Data:      dimension: 384 8
##    dimension: 384 1
## Fit method: kernelpls
##  umber of components considered: 8
##
##     D     : RMS P
##  ross-validated using 10 random segments.
##        ( ntercept)  1 comps  2 comps  3 comps  4 comps  5 comps  6 comps
##               10.2    4.671    3.411    3.313    3.101    3.018    3.013
## adj           10.2    4.669    3.407    3.310    3.074    3.014    3.010
##        7 comps  8 comps
##          3.001    3.460
## adj      2.998    2.805
##
##  R      :   variance explained
##               1 comps  2 comps  3 comps  4 comps  5 comps  6 comps
##                 45.04    61.41    70.64    73.17    87.10    99.92
## heating_load    79.30    89.11    89.69    91.39    91.57    91.57
##               7 comps  8 comps
##                100.00   100.21
## heating_load    91.65    91.64
```

```
validationplot(pls.fit,val.type="MS P")
```




```
pls.pred=predict(pls.fit,energy[ train,],ncomp=2)
mean((pls.pred-heating_load[ train]) 2)
```

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

```
summary(pls.fit)
```

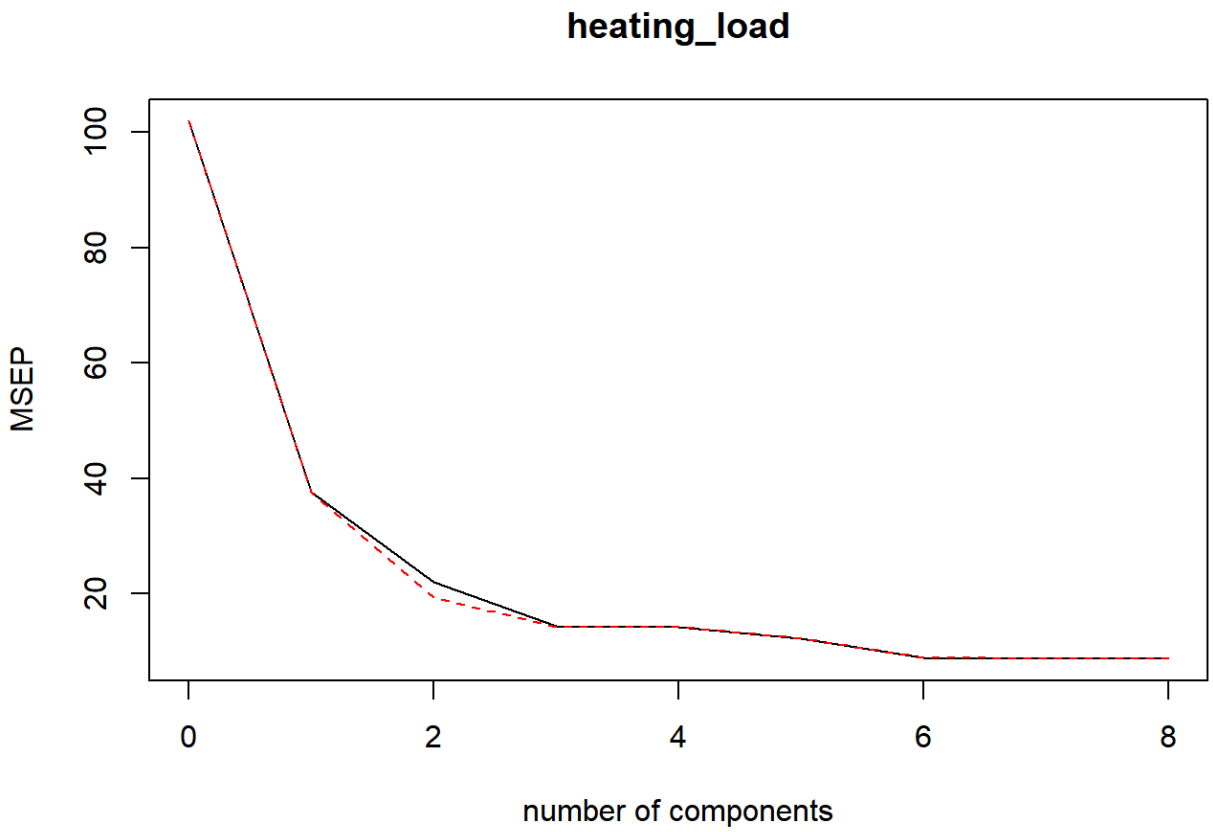
```
## Data:      dimension: 384 8
##    dimension: 384 1
## Fit method: kernelpls
##  umber of components considered: 8
##
##     D     : RMS P
##  ross-validated using 10 random segments.
##        ( ntercept)  1 comps  2 comps  3 comps  4 comps  5 comps  6 comps
##               10.2    4.671    3.411    3.313    3.101    3.018    3.013
## adj           10.2    4.669    3.407    3.310    3.074    3.014    3.010
##        7 comps  8 comps
##          3.001    3.460
## adj      2.998    2.805
##
##  R      :   variance explained
##                1 comps  2 comps  3 comps  4 comps  5 comps  6 comps
##                  45.04    61.41    70.64    73.17    87.10    99.92
## heating_load    79.30    89.11    89.69    91.39    91.57    91.57
##               7 comps  8 comps
##                100.00   100.21
## heating_load    91.65    91.64
```

Principal Components Regression

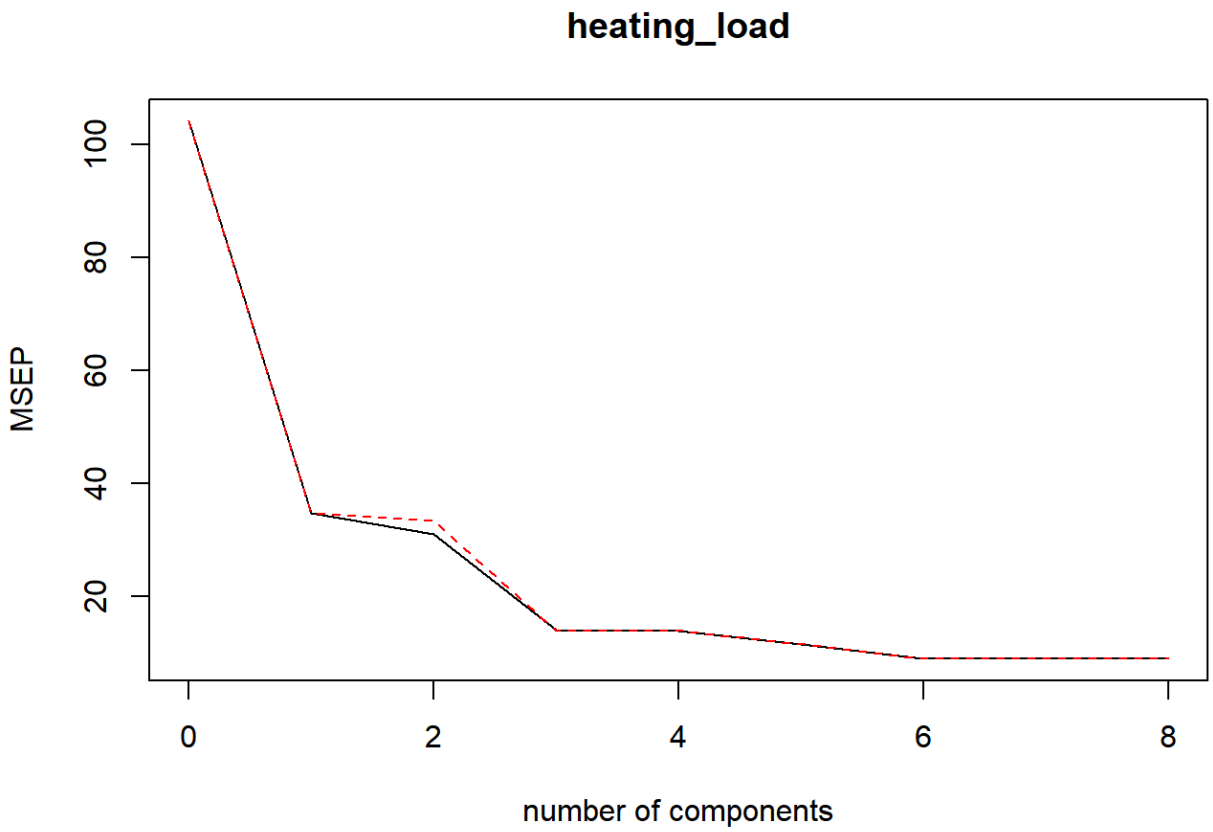
```
set.seed(2)
pcr.fit=pcr(heating_load .-cooling_load, data=energy,scale= R  ,validation="  ")
summary(pcr.fit)
```

```
## Data:      dimension: 768 8
##    dimension: 768 1
## Fit method: svdpc
##  umber of components considered: 8
##
##     D     : RMS P
##  ross-validated using 10 random segments.
##        ( ntercept)  1 comps  2 comps  3 comps  4 comps  5 comps  6 comps
##               10.1    6.133    4.690    3.784    3.778    3.495    2.982
## adj           10.1    6.132    4.407    3.773    3.775    3.493    2.981
##        7 comps  8 comps
##          2.957    2.973
## adj      2.955    2.956
##
##  R      :   variance explained
##                1 comps  2 comps  3 comps  4 comps  5 comps  6 comps
##                  46.29    61.78    76.95    89.45    99.28    99.94
## heating_load    63.23    80.87    86.13    86.13    88.24    91.46
##               7 comps  8 comps
##                100.00   100.00
## heating_load    91.62    91.62
```

```
validationplot(pcr.fit,val.type="MS P")
```




```
set.seed(1)
energy.train=energy[train,]
energy.test=energy[ train,]
hload.train=heating_load[train]
hload.test=heating_load[ train]

pcr.fit=pcr(heating_load .-cooling_load, data=energy, subset=train,scale= R  , validat
ion="  ")
validationplot(pcr.fit,val.type="MS P")
```



Chapter

```
fit=lm(heating_load poly(surface_area,4),data=energy)
coef(summary(fit))
```

```
##                            stimate Std.  rror    t value      Pr(  t )
## ( ntercept)              22.30720   0.232871  95.792113  0.000000e 00
## poly(surface_area, 4)1 -183.90880   6.453509 -28.497488 3.259806e-122
## poly(surface_area, 4)2  -34.77301   6.453509  -5.388233  9.488203e-08
## poly(surface_area, 4)3   99.23903   6.453509  15.377531  1.103886e-46
## poly(surface_area, 4)4   37.84959   6.453509   5.864962  6.689696e-09
```

```
fit2=lm(heating_load poly(surface_area,4,raw= ),data=energy)
coef(summary(fit2))
```

```
##                                       stimate   Std.  rror   t value
## ( ntercept)                        3.952241e 03 9.601885e 02  4.116109
## poly(surface_area, 4, raw =  )1 -2.693894e 01 5.940182e 00 -4.535035
## poly(surface_area, 4, raw =  )2  6.822672e-02 1.367174e-02  4.990348
## poly(surface_area, 4, raw =  )3 -7.547985e-05 1.387715e-05 -5.439148
## poly(surface_area, 4, raw =  )4  3.074757e-08 5.242585e-09  5.864962
##                                      Pr(  t )
## ( ntercept)                       4.273143e-05
## poly(surface_area, 4, raw =  )1 6.686209e-06
## poly(surface_area, 4, raw =  )2 7.470493e-07
## poly(surface_area, 4, raw =  )3 7.214574e-08
## poly(surface_area, 4, raw =  )4 6.689696e-09
```

```
fit2a=lm(heating_load surface_area  (overall_height wall_area)  (roof_area orientatio
n)  (glazing_area orientation),data=energy)
coef(fit2a)
```

```
##                   ( ntercept)                   surface_area
##                   5.889217204                   -0.006452942
##  (overall_height   wall_area)     (roof_area   orientation)
##                   0.011997491                   -0.006099466
##  (glazing_area   orientation)
##                   5.125053486
```

```
fit2b=lm(heating_load cbind(surface_area,overall_height,wall_area,orientation),data=en
ergy)
surfacelims=range(surface_area)
surface.grid=seq(from=surfacelims[1],to=surfacelims[2])
preds=predict(fit,newdata=list(surface=surface.grid),se= R  )
```

```
se.bands=cbind(preds fit 2 preds se.fit,preds fit-2 preds se.fit)
max(se.bands)
```

```
## [1] 32.74122
```

```
anova(fit,fit2,fit2a,fit2b)
```

```
##  nalysis of  ariance  able
##
## Model 1: heating_load   poly(surface_area, 4)
## Model 2: heating_load   poly(surface_area, 4, raw =  )
## Model 3: heating_load   surface_area    (overall_height   wall_area)
##      (roof_area   orientation)    (glazing_area   orientation)
## Model 4: heating_load   cbind(surface_area, overall_height, wall_area,
##     orientation)
##   Res.Df   RSS Df Sum of Sq F Pr( F)
## 1    763 31777
## 2    763 31777  0       0.0
## 3    763  7428  0   24349.0
## 4    763 12644  0   -5215.9
```

```
coef(summary(fit2b))
```

```
##                                                                                stim
ate
## ( ntercept)                                                                -28.74603
429
## cbind(surface_area, overall_height, wall_area, orientation)surface_area     0.01538
361
## cbind(surface_area, overall_height, wall_area, orientation)overall_height   5.53040
787
## cbind(surface_area, overall_height, wall_area, orientation)wall_area        0.03694
487
## cbind(surface_area, overall_height, wall_area, orientation)orientation     -0.02332
813
##                                                                             Std.  rr
or
## ( ntercept)                                                               4.5658690
27
## cbind(surface_area, overall_height, wall_area, orientation)surface_area   0.0070137
54
## cbind(surface_area, overall_height, wall_area, orientation)overall_height 0.3605225
77
## cbind(surface_area, overall_height, wall_area, orientation)wall_area      0.0075757
20
## cbind(surface_area, overall_height, wall_area, orientation)orientation    0.1313854
29
##                                                                               t valu
e
## ( ntercept)                                                               -6.295851
7
## cbind(surface_area, overall_height, wall_area, orientation)surface_area    2.193349
0
## cbind(surface_area, overall_height, wall_area, orientation)overall_height 15.339976
5
## cbind(surface_area, overall_height, wall_area, orientation)wall_area       4.876747
4
## cbind(surface_area, overall_height, wall_area, orientation)orientation    -0.177554
9
##                                                                                  Pr(
t )
## ( ntercept)                                                               5.156048e
-10
## cbind(surface_area, overall_height, wall_area, orientation)surface_area   2.858329e
-02
## cbind(surface_area, overall_height, wall_area, orientation)overall_height 1.718169e
-46
## cbind(surface_area, overall_height, wall_area, orientation)wall_area      1.312890e
-06
## cbind(surface_area, overall_height, wall_area, orientation)orientation    8.591197e
-01
```

Chapter 8

Fitting Regression Trees

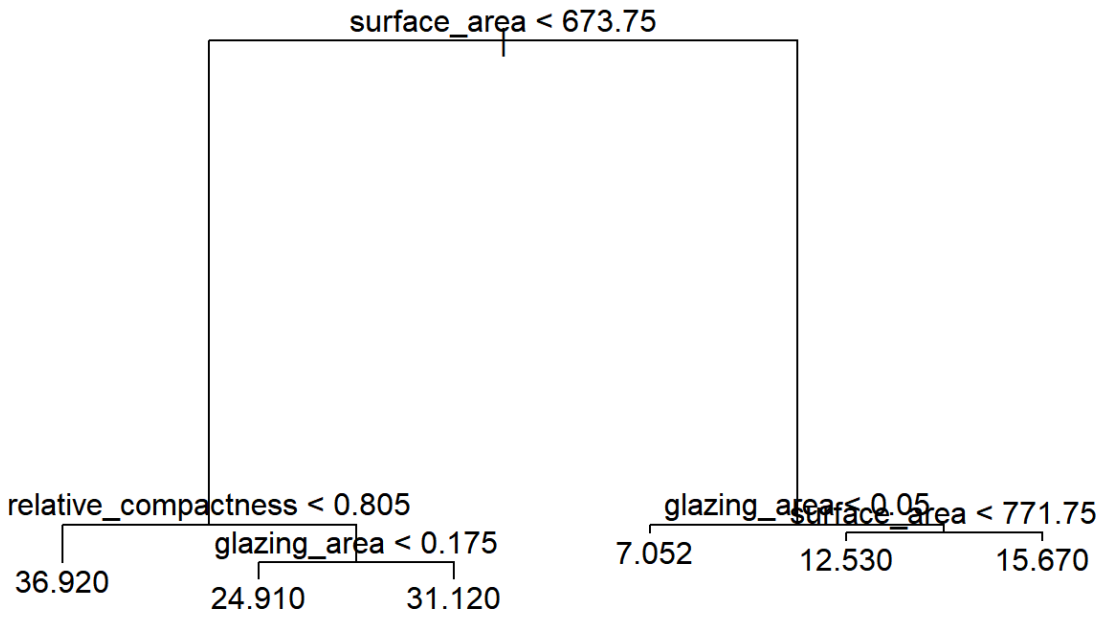
```
library(M SS)
library(tree)
```

```
## Warning: package 'tree' was built under R version 3.3.3
```

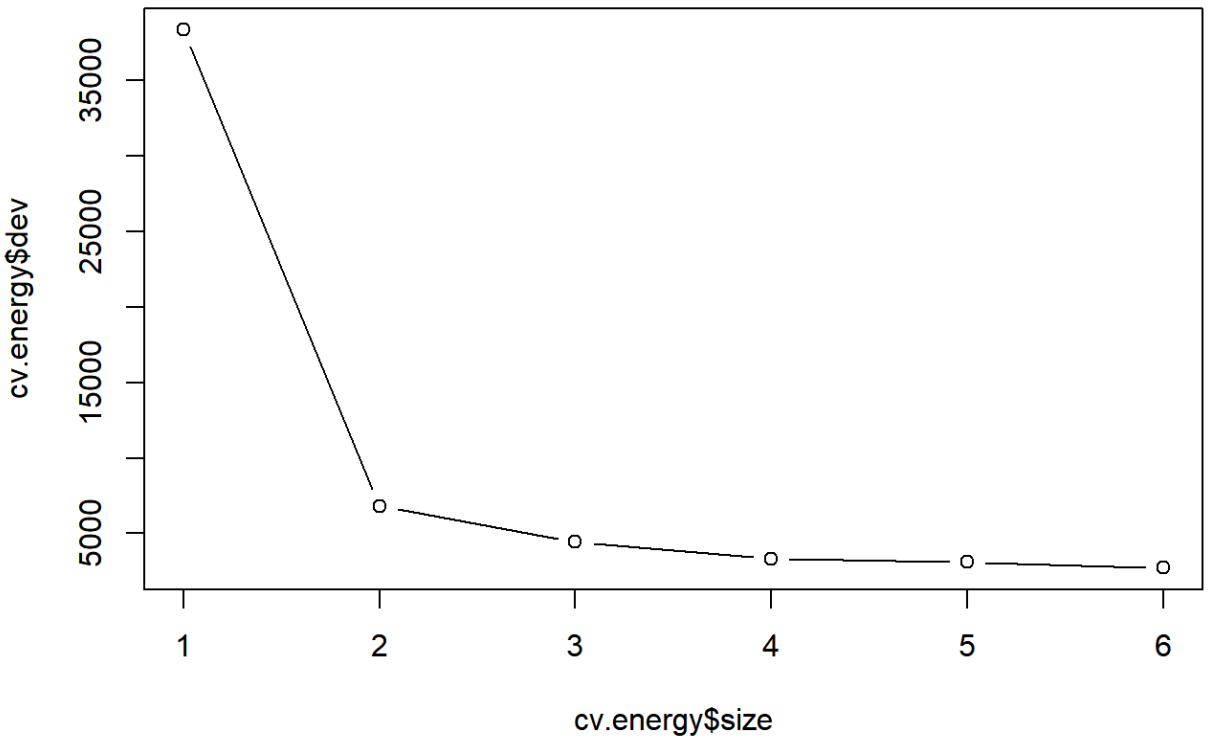
```
set.seed(1)
train = sample(1:nrow(energy), nrow(energy)/2)
tree.energy=tree(heating_load .-cooling_load, data=energy,subset=train)
summary(tree.energy)
```

```
##
## Regression tree:
## tree(formula = heating_load   . - cooling_load, data = energy,
##     subset = train)
##  ariables actually used in tree construction:
## [1] "surface_area"         "relative_compactness" "glazing_area"
##  umber of terminal nodes:  6
## Residual mean deviance:  6.026 = 2278 / 378
## Distribution of residuals:
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## -12.9900  -1.3690  -0.2265   0.0000   1.6140   6.1220
```

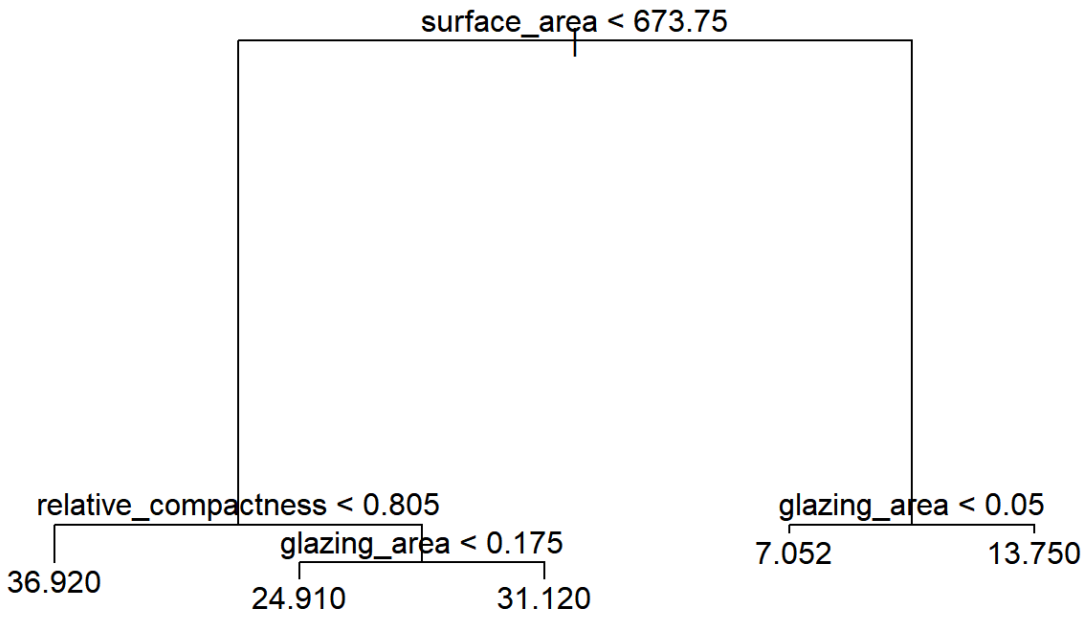
```
plot(tree.energy)
text(tree.energy,pretty=0)
```




```
cv.energy=cv.tree(tree.energy)
plot(cv.energy size,cv.energy dev,type='b')
```




```
#pruning
prune.energy=prune.tree(tree.energy,best=5)
plot(prune.energy)
text(prune.energy,pretty=0)
yhat=predict(tree.energy,newdata=energy[-train,])
energy.test=energy[-train,"heating_load"]
abline(0,1)
```




```
mean((yhat-energy.test) 2)
```

```
## [1] 9.371754
```

Random Forest

```
library(randomForest)
```

```
## Warning: package 'randomForest' was built under R version 3.3.3
```

```
## randomForest 4.6-14
```

```
##  ype rf ews() to see new features/changes/bug fixes.
```

```
set.seed(1)
bag.energy=randomForest(heating_load .-cooling_load, data=energy,subset=train,mtry=8,i
mportance= R  )
bag.energy
```

```
##
##  all:
##  randomForest(formula = heating_load   . - cooling_load, data = energy,      mtry
= 8, importance =  R  , subset = train)
##                 ype of random forest: regression
##                       umber of trees: 500
##  o. of variables tried at each split: 8
##
##           Mean of squared residuals: 0.3186205
##                         ar explained: 99.68
```

```
yhat.bag = predict(bag.energy,newdata=energy[-train,])
mean((yhat.bag-energy.test) 2)
```

```
## [1] 0.5750208
```

```
bag.energy=randomForest(heating_load .-cooling_load, data=energy,subset=train,mtry=8,n
tree=15)
yhat.bag = predict(bag.energy,newdata=energy[-train,])
mean((yhat.bag-energy.test) 2)
```

```
## [1] 0.5781198
```

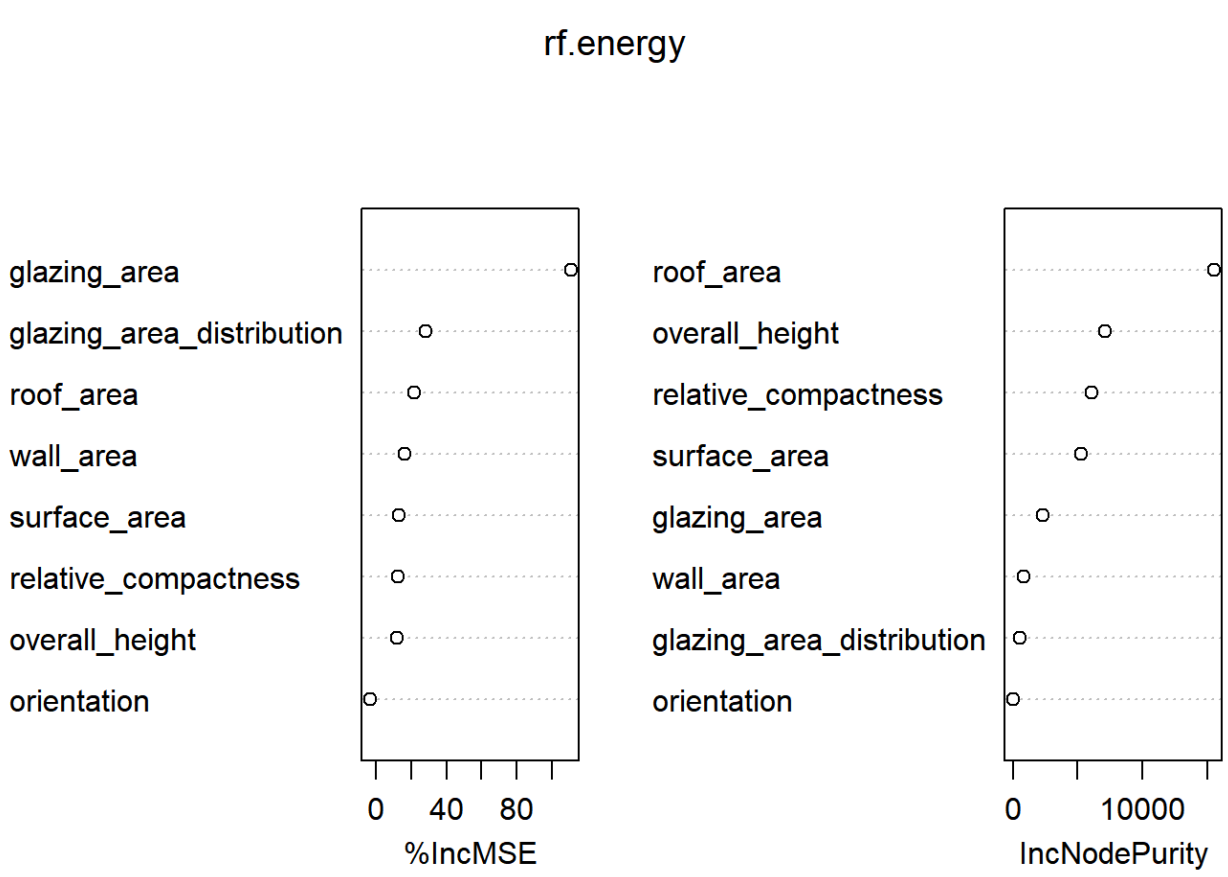
```
set.seed(1)
rf.energy=randomForest(heating_load .-cooling_load, data=energy,subset=train,mtry=5,im
portance= R  )
yhat.rf = predict(rf.energy,newdata=energy[-train,])
mean((yhat.rf-energy.test) 2)
```

```
## [1] 0.7347269
```

```
importance(rf.energy)
```

```
##                              ncMS   nc odePurity
## relative_compactness       12.515133    6047.63750
## surface_area               12.972023    5264.24172
## wall_area                  16.064636     819.27886
## roof_area                  21.762050   15483.65600
## overall_height             11.952900    7119.88632
## orientation                -3.770718      34.19988
## glazing_area              110.876688    2339.40616
## glazing_area_distribution  27.931019     554.74209
```

```
var mpPlot(rf.energy)
```



Boosting

```
library(gbm)
```

```
## Warning: package 'gbm' was built under R version 3.3.3
```

```
##  oading required package: survival
```

```
##
##  ttaching package: 'survival'
```

```
##  he following object is masked from 'package:boot':
##
##     aml
```

```
##  oading required package: lattice
```

```
##
##  ttaching package: 'lattice'
```

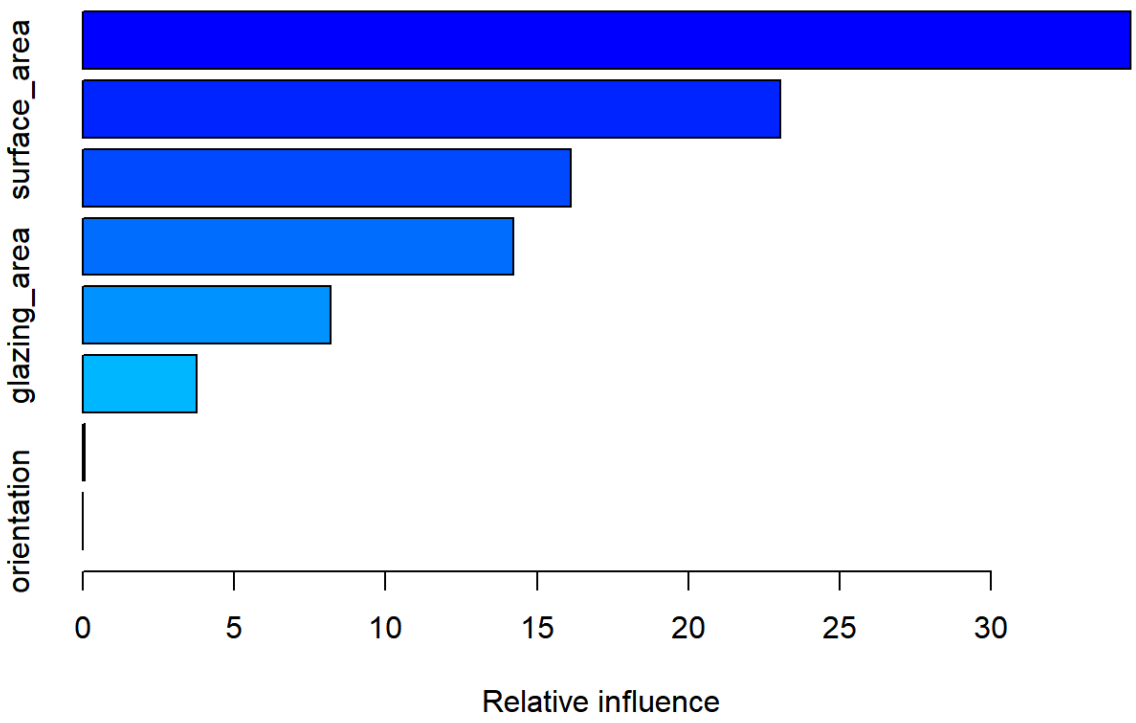
```
##  he following object is masked from 'package:boot':
##
##     melanoma
```

```
##  oading required package: splines
```

```
##  oading required package: parallel
```

```
##  oaded gbm 2.1.3
```

```
set.seed(1)
boost.energy=gbm(heating_load .-cooling_load,data=energy[train,],distribution="gaussia
n",n.trees=5000,interaction.depth=4)
summary(boost.energy)
```



```
##                                             var      rel.inf
## relative_compactness           relative_compactness 3.460257e 01
## surface_area                           surface_area 2.304977e 01
## roof_area                                 roof_area 1.612662e 01
## overall_height                       overall_height 1.422015e 01
## glazing_area                           glazing_area 8.181105e 00
## wall_area                                 wall_area 3.762536e 00
## glazing_area_distribution glazing_area_distribution 5.661478e-02
## orientation                             orientation 6.405433e-04
```

```
yhat.boost=predict(boost.energy,newdata=energy[-train,],n.trees=5000)
mean((yhat.boost-energy.test) 2)
```

```
## [1] 1.205164
```

```
boost.energy=gbm(heating_load .-cooling_load,data=energy[train,],distribution="gaussia
n",n.trees=5000,interaction.depth=4,shrinkage=0.2,verbose=F)
yhat.boost=predict(boost.energy,newdata=energy[-train,],n.trees=5000)
mean((yhat.boost-energy.test) 2)
```

```
## [1] 0.3368904
```

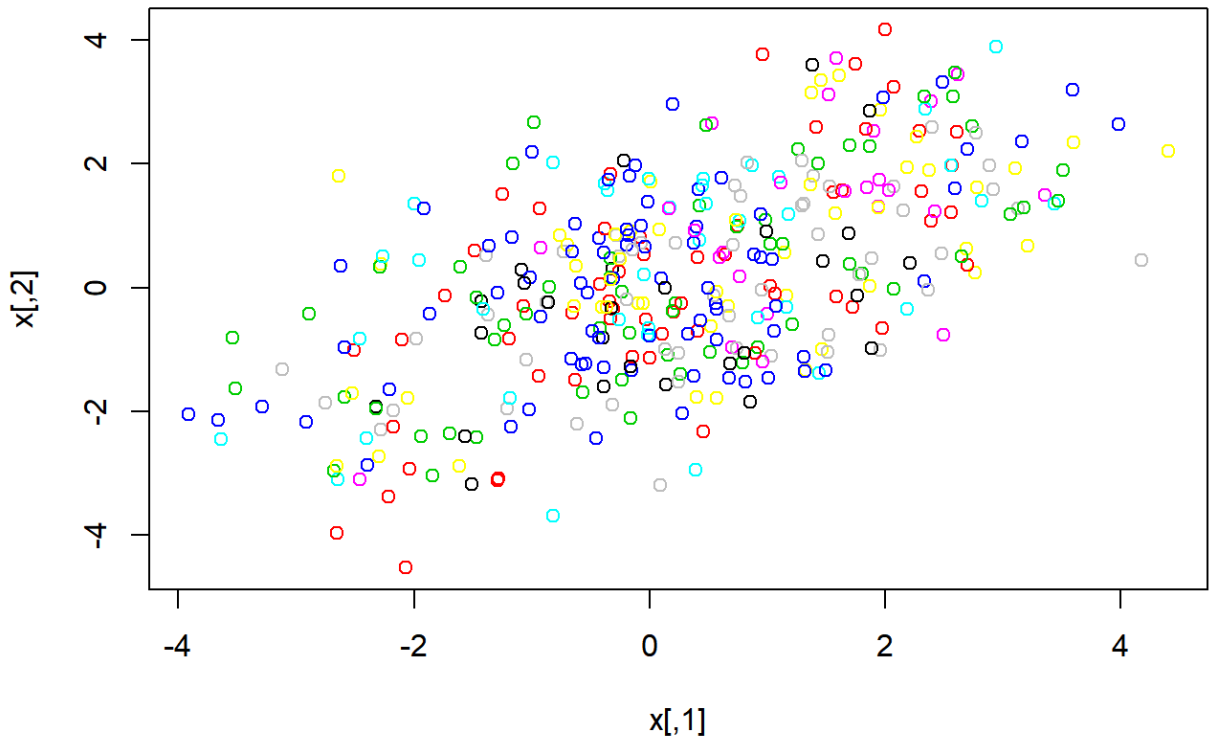
Chapter

Support Vector Machine

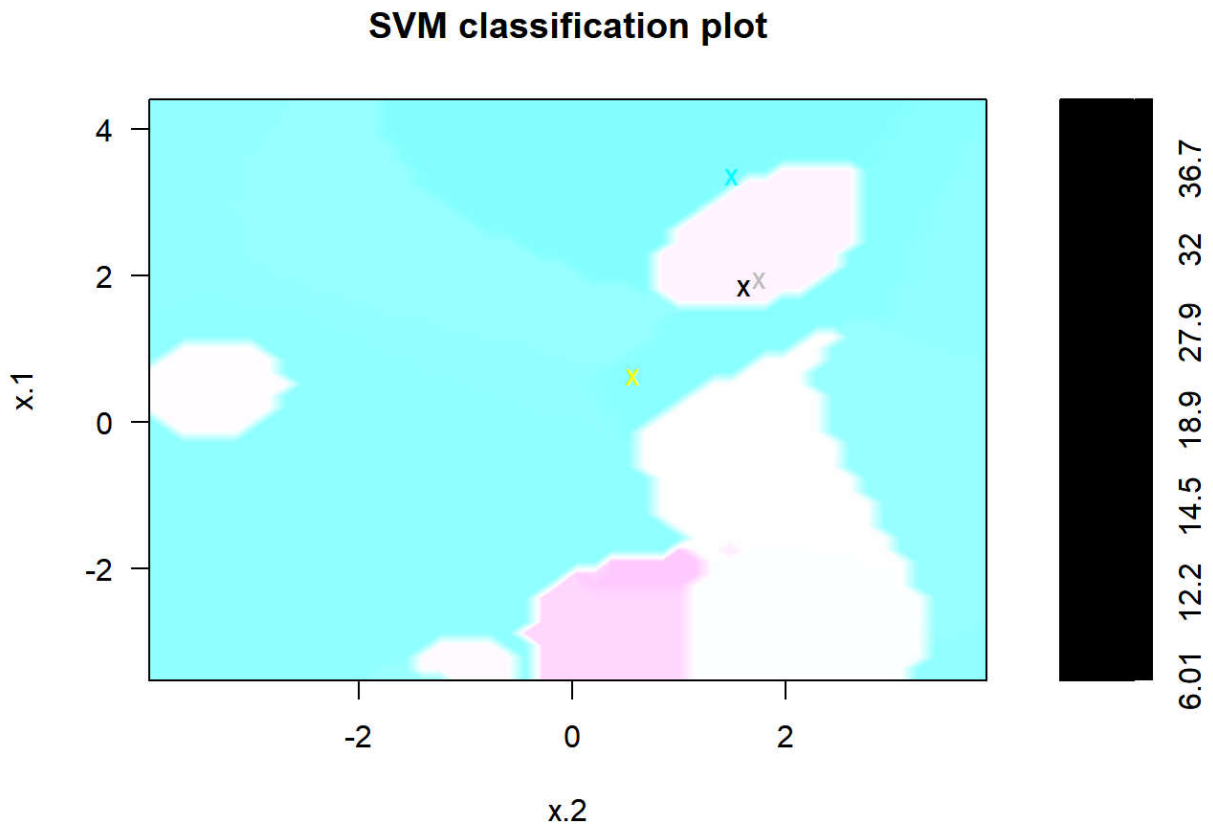
```
library(e1071)
```

```
## Warning: package 'e1071' was built under R version 3.3.3
```

```
set.seed(1)
energy <- read_excel("F:/Spring 2018/quantitative methods/Project/Data/energy.xlsx")
x=matrix(rnorm(surface_area overall_height), ncol=2)
x[1:100,]=x[1:100,] 2
x[101:150,]=x[101:150,]-2
y=c(heating_load)
dat=data.frame(x=x,y=as.factor(y))

plot(x, col=y)
```




```
train=sample(200,100)
svmfit=svm(y ., data=dat[train,], kernel="radial",  gamma=1, cost=1)
plot(svmfit, dat[train,])
```

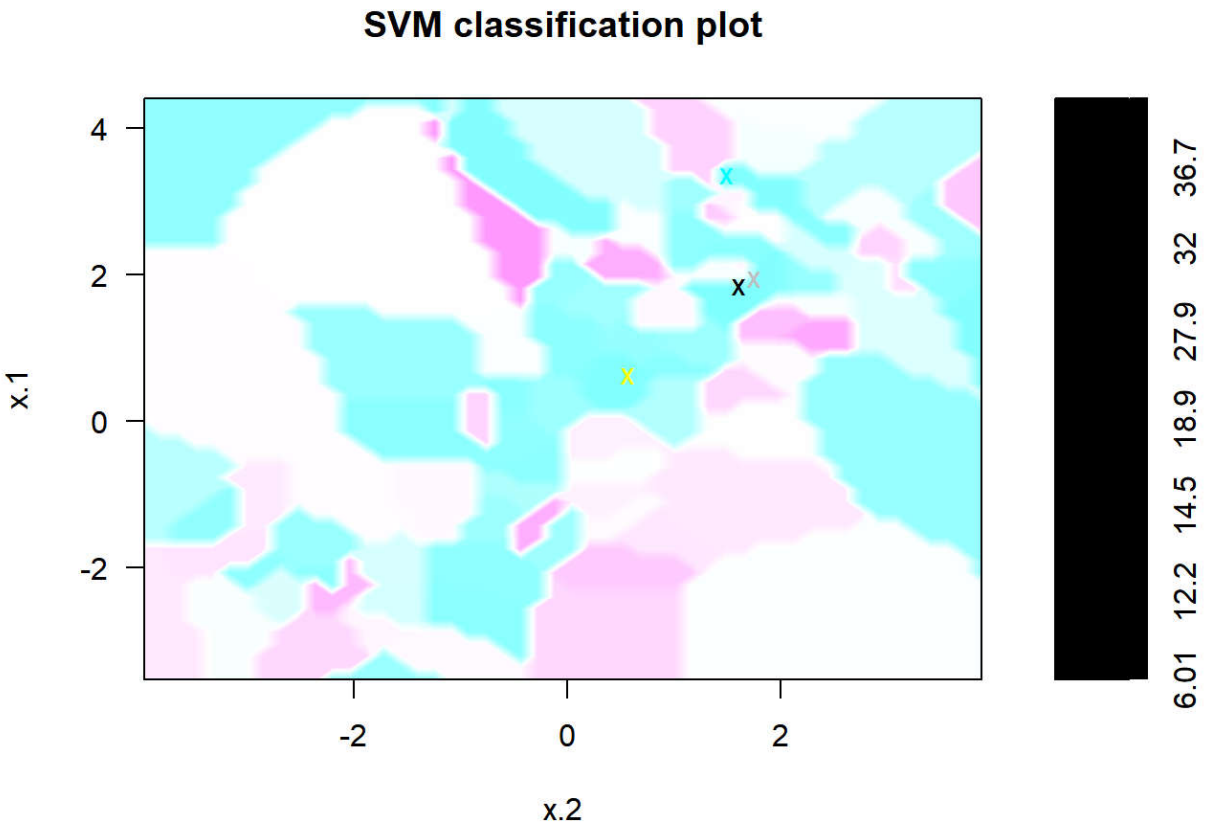



```
summary(svmfit)
```

```
##
##  all:
## svm(formula = y   ., data = dat[train, ], kernel = "radial",
##     gamma = 1, cost = 1)
##
##
## Parameters:
##    S M- ype:   -classification
##  S M- ernel:  radial
##        cost:  1
##       gamma:  1
##
##  umber of Support  ectors:  100
##
##  ( 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 2 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1
2 1 1 2 1 2 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 )
##
##
##  umber of  lasses:  93
##
##  evels:
##  6.01 6.04 6.05 6.07 6.37 6.4 6.77 6.79 6.81 6.85 7.1 7.18 8.45 8.49 8.5 8.6 10.07
10.08 10.14 10.15 10.32 10.34 10.35 10.36 10.37 10.38 10.39 10.42 10.43 10.45 10.46 1
0.47 10.53 10.54 10.55 10.56 10.64 10.66 10.67 10.68 10.7 10.71 10.72 10.75 10.77 10.7
8 10.8 10.85 11.07 11.09 11.1 11.11 11.13 11.14 11.16 11.18 11.2 11.21 11.22 11.32 11.
33 11.34 11.38 11.42 11.43 11.44 11.45 11.46 11.49 11.53 11.59 11.6 11.61 11.64 11.67
11.68 11.69 11.7 11.8 11.98 12.02 12.03 12.1 12.12 12.16 12.17 12.18 12.19 12.2 12.25
12.27 12.28 12.29 12.3 12.31 12.32 12.33 12.34 12.35 12.36 12.41 12.42 12.43 12.45 12.
46 12.47 12.49 12.5 12.57 12.59 12.62 12.63 12.65 12.67 12.68 12.71 12.72 12.73 12.74
12.76 12.77 12.78 12.8 12.82 12.84 12.85 12.86 12.87 12.88 12.91 12.92 12.93 12.95 12.
96 12.97 13 13.01 13.02 13.04 13.05 13.17 13.18 13.68 13.69 13.78 13.86 13.91 13.94 1
3.95 13.97 13.99 14.03 14.07 14.08 14.1 14.12 14.16 14.17 14.18 14.19 14.21 14.22 14.2
8 14.32 14.33 14.34 14.37 14.39 14.4 14.41 14.42 14.44 14.45 14.47 14.48 14.5 14.51 1
4.52 14.53 14.54 14.55 14.56 14.58 14.6 14.61 14.62 14.65 14.66 14.7 14.71 14.72 14.7
5 14.9 14.92 14.96 15.08 15.09 15.12 15.16 15.18 15.19 15.2 15.21 15.23 15.29 15.3 15.
32 15.34 15.36 15.37 15.4 15.41 15.42 15.55 15.98 16.35 16.44 16.47 16.48 16.54 16.55
16.56 16.62 16.64 16.66 16.69 16.73 16.74 16.76 16.77 16.83 16.84 16.86 16.9 16.92 16.
93 16.94 16.95 16.99 17.02 17.05 17.11 17.14 17.15 17.17 17.23 17.26 17.35 17.37 17.4
1 17.5 17.52 17.69 17.88 18.16 18.19 18.31 18.46 18.48 18.71 18.84 18.88 18.9 19 19.0
6 19.12 19.13 19.2 19.34 19.36 19.42 19.48 19.5 19.52 19.68 19.95 20.71 20.84 21.46 2
2.58 22.79 22.8 22.89 22.93 23.53 23.54 23.59 23.67 23.75 23.8 23.84 23.86 23.87 23.8
9 23.93 24.03 24.04 24.11 24.13 24.17 24.23 24.24 24.25 24.26 24.28 24.29 24.31 24.32
24.33 24.35 24.37 24.38 24.4 24.58 24.59 24.6 24.63 24.7 24.77 24.94 24.96 25.17 25.2
7 25.36 25.37 25.38 25.41 25.43 25.48 25.49 25.66 25.7 25.74 25.98 26 26.19 26.28 26.3
3 26.37 26.44 26.45 26.46 26.47 26.48 26.84 26.89 26.91 26.97 27.02 27.03 27.27 27.9 2
8.01 28.03 28.05 28.07 28.09 28.15 28.17 28.18 28.31 28.37 28.4 28.41 28.42 28.52 28.5
5 28.56 28.57 28.58 28.6 28.61 28.62 28.63 28.64 28.65 28.66 28.67 28.69 28.7 28.75 2
```

```
8.83 28.86 28.88 28.91 28.93 28.95 29.01 29.02 29.03 29.05 29.06 29.07 29.08 29.09 29.
14 29.22 29.27 29.34 29.39 29.4 29.43 29.47 29.49 29.5 29.52 29.53 29.54 29.6 29.62 2
9.63 29.67 29.68 29.71 29.79 29.83 29.87 29.88 29.9 29.91 29.92 30 30.05 31.12 31.28 3
1.29 31.53 31.63 31.64 31.66 31.69 31.81 31.84 31.89 32 32.05 32.06 32.07 32.09 32.12
32.13 32.15 32.21 32.23 32.24 32.26 32.29 32.31 32.33 32.38 32.39 32.4 32.41 32.46 32.
48 32.49 32.52 32.53 32.67 32.68 32.69 32.71 32.72 32.73 32.74 32.75 32.82 32.84 32.8
5 32.94 32.96 33.08 33.09 33.12 33.13 33.16 33.21 33.24 33.27 33.28 33.48 34.24 34.29
34.72 34.95 35.01 35.05 35.24 35.4 35.45 35.48 35.56 35.64 35.65 35.67 35.69 35.73 35.
78 35.84 35.89 35.94 35.96 35.99 36.03 36.06 36.13 36.26 36.28 36.43 36.45 36.47 36.5
2 36.57 36.59 36.64 36.66 36.7 36.71 36.77 36.81 36.86 36.9 36.91 36.95 36.96 36.97 3
7.03 37.1 37.12 37.24 37.26 38.33 38.35 38.57 38.65 38.67 38.82 38.84 38.89 38.98 39.0
1 39.04 39.31 39.32 39.68 39.72 39.81 39.83 39.84 39.86 39.89 39.97 40 40.03 40.11 40.
12 40.15 40.19 40.4 40.42 40.43 40.57 40.6 40.68 40.71 40.78 40.79 41.09 41.26 41.3 4
1.32 41.4 41.64 41.67 41.73 41.92 41.96 42.08 42.11 42.49 42.5 42.62 42.74 42.77 42.9
6 43.1
```

```
svmfit=svm(y ., data=dat[train,], kernel="radial",gamma=1,cost=1e5)
plot(svmfit,dat[train,])
```



```
set.seed(1)
tune.out=tune(svm, y ., data=dat[train,], kernel="radial", ranges=list(cost=c(0.1,1,1
0,100,1000),gamma=c(0.5,1,2,3,4)))
summary(tune.out)
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost gamma
##   0.1   0.5
##
## - best performance: 1
##
## - Detailed performance results:
##     cost gamma error dispersion
## 1  1e-01   0.5     1          0
## 2  1e 00   0.5     1          0
## 3  1e 01   0.5     1          0
## 4  1e 02   0.5     1          0
## 5  1e 03   0.5     1          0
## 6  1e-01   1.0     1          0
## 7  1e 00   1.0     1          0
## 8  1e 01   1.0     1          0
## 9  1e 02   1.0     1          0
## 10 1e 03   1.0     1          0
## 11 1e-01   2.0     1          0
## 12 1e 00   2.0     1          0
## 13 1e 01   2.0     1          0
## 14 1e 02   2.0     1          0
## 15 1e 03   2.0     1          0
## 16 1e-01   3.0     1          0
## 17 1e 00   3.0     1          0
## 18 1e 01   3.0     1          0
## 19 1e 02   3.0     1          0
## 20 1e 03   3.0     1          0
## 21 1e-01   4.0     1          0
## 22 1e 00   4.0     1          0
## 23 1e 01   4.0     1          0
## 24 1e 02   4.0     1          0
## 25 1e 03   4.0     1          0
```

```
#table(true=dat[-train,"y"], pred=predict(tune.out$best.model,newdata=dat[-train,]))
```