CSCI E-106:Assignment 7

Problem 1

Use the fat data(copy and paste the following command into R console:library(faraway);data("fat")), and use the percentage of body fat, siri, as the response and the other variables, except brozek and density as potential predictors. Use 70% of the data for train data set and use remaining data (30% of data) for test data set (use set.seed(1023)). (50 points)

```
data("fat")
set.seed(1023)
DataSplit<-createDataPartition(y = fat$siri, p = 0.7, list = FALSE)
training_data <- fat[DataSplit,]
testing_data <- fat[-DataSplit, ]
```

a-)Linear regression with all predictors (5 pts)

```
# Fit Model
full_fat_lm <- lm(siri~.-brozek -density, data=training_data)
summary(full_fat_lm)
```

```
##
## Call:
## lm(formula = siri ~ . - brozek - density, data = training_data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4.8292 -0.6071  0.1508  0.8555  6.7490
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -11.30892    6.68382  -1.692  0.09257 .
## age           0.02713    0.01308   2.075  0.03959 *
## weight        0.36828    0.02348  15.685  < 2e-16 ***
## height        0.05074    0.03703   1.370  0.17249
## adipos       -0.38123    0.11998  -3.177  0.00178 **
## free         -0.53750    0.01494 -35.980  < 2e-16 ***
## neck          0.21100    0.10587   1.993  0.04794 *
## chest         0.02490    0.04486   0.555  0.57959
## abdom         0.13092    0.04286   3.055  0.00264 **
## hip           0.04711    0.05611   0.840  0.40229
## thigh         0.18063    0.05979   3.021  0.00293 **
## knee          0.06943    0.09716   0.715  0.47587
## ankle         0.12452    0.07662   1.625  0.10606
## biceps        0.10867    0.07551   1.439  0.15205
## forearm       0.09498    0.08276   1.148  0.25280
## wrist        -0.17428    0.21040  -0.828  0.40872
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
##
## Residual standard error: 1.399 on 162 degrees of freedom
## Multiple R-squared:  0.9755, Adjusted R-squared:  0.9732
## F-statistic: 430.2 on 15 and 162 DF,  p-value: < 2.2e-16
```

```
# Calculate MSE - Training Data
yhat.train <- predict(full_fat_lm, newdata = training_data)
mse.lm <- mean((training_data$siri - yhat.train) ^ 2)
mse.lm
```

```
## [1] 1.780429
```

```
# Calculate MSPE - Testing Data
yhat.test <- predict(full_fat_lm,newdata = testing_data, type="response")
mspe.lm <- mean((testing_data$siri - yhat.test) ^ 2)
mspe.lm
```

```
## [1] 4.190569
```

The model's performance on the training data appears robust: several predictor variables are significant, the residual standard error (RSE) is low, residuals are within a tight range, and both the *(Adjusted and Non-Adjusted)* R^2 values are high. However, it's important to examine both in-sample and out-of-sample errors for a fuller picture. Mean squared error (MSE) measures the average squared difference between observed training responses and model-fitted responses, helping to evaluate how well our model fits the training data. Here, the MSE of 1.7804291 suggests a good fit on the training data, as lower MSE values indicate a model that closely captures observed data patterns. Mean squared prediction error (MSPE), in contrast, assesses prediction quality by averaging the squared differences between predicted values and actual test values. A higher MSPE (4.1905688) than MSE indicates that the model does not generalize as well on new data as it does on the training data. This gap may signal some overfitting on the training data, where the model captures noise instead of the true pattern. It could also reflect variability in the test set that wasn't as prominent in the training set, leading to increased MSPE.

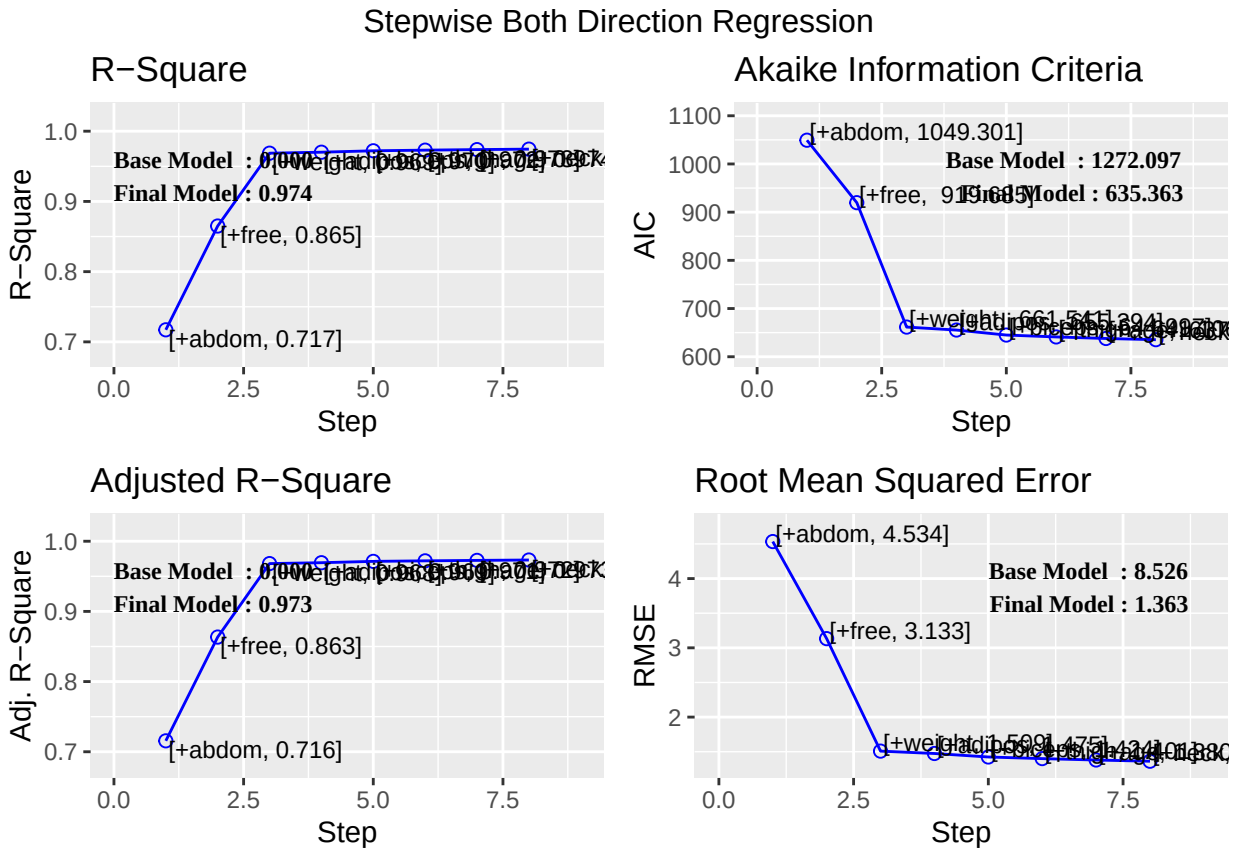
b-)Linear regression with variables selected using stepwise (both ways) selection criteria (5 pts)

```
stp_wse_fat <- ols_step_both_p(full_fat_lm) # Remove details = TRUE now that we identified the Model 3 p
stp_wse_fat
```

```
##
##
##                               Stepwise Summary
## -------------------------------------------------------------------------
## Step    Variable        AIC         SBC       SBIC      R2       Adj. R2
## -------------------------------------------------------------------------
##  0      Base Model    1272.097    1278.460      NA    0.00000    0.00000
##  1      abdom (+)     1049.301    1058.847      NA    0.71717    0.71556
##  2      free (+)       919.685     932.412      NA    0.86498    0.86343
##  3      weight (+)     661.541     677.450      NA    0.96869    0.96815
##  4      adipos (+)     655.394     674.484      NA    0.97009    0.96940
##  5      biceps (+)     644.997     667.270      NA    0.97210    0.97129
##  6      thigh (+)      641.106     666.561      NA    0.97301    0.97206
##  7      age (+)        637.886     666.522      NA    0.97379    0.97271
##  8      neck (+)       635.363     667.181      NA    0.97445    0.97324
## -------------------------------------------------------------------------
##
## Final Model Output
## ------------------
```

```
##
##                          Model Summary
## ----------------------------------------------------------------
## R                       0.987       RMSE                 1.363
## R-Squared               0.974       MSE                  1.956
## Adj. R-Squared          0.973       Coef. Var            7.279
## Pred R-Squared          0.970       AIC                635.363
## MAE                     0.961       SBC                667.181
## ----------------------------------------------------------------
##  RMSE: Root Mean Square Error
##  MSE: Mean Square Error
##  MAE: Mean Absolute Error
##  AIC: Akaike Information Criteria
##  SBC: Schwarz Bayesian Criteria
##
##                                ANOVA
## --------------------------------------------------------------------------
##                   Sum of
##                  Squares         DF    Mean Square       F         Sig.
## --------------------------------------------------------------------------
## Regression    12608.952          8       1576.119    805.625    0.0000
## Residual        330.630        169          1.956
## Total         12939.582        177
## --------------------------------------------------------------------------
##
##                                 Parameter Estimates
## ----------------------------------------------------------------------------------------------
##       model      Beta    Std. Error    Std. Beta       t        Sig      lower     upper
## ----------------------------------------------------------------------------------------------
## (Intercept)    -0.599         3.262                -0.184    0.855    -7.038     5.840
##       abdom     0.134         0.038        0.169    3.475    0.001     0.058     0.210
##        free    -0.535         0.014       -1.088  -37.785    0.000    -0.563    -0.507
##      weight     0.397         0.019        1.304   21.186    0.000     0.360     0.434
##      adipos    -0.404         0.089       -0.169   -4.520    0.000    -0.581    -0.228
##      biceps     0.138         0.071        0.050    1.954    0.052    -0.001     0.278
##       thigh     0.178         0.050        0.109    3.587    0.000     0.080     0.276
##         age     0.023         0.011        0.036    2.102    0.037     0.001     0.045
##        neck     0.193         0.093        0.052    2.086    0.039     0.010     0.376
## ----------------------------------------------------------------------------------------------
```

```
plot(stp_wse_fat)
```




```
stp_wse_fat_lm <- lm(siri ~ abdom + free + weight, data=training_data) # Extract model # 3 siri ~ abdom
summary(stp_wse_fat_lm)
```

```
##
## Call:
## lm(formula = siri ~ abdom + free + weight, data = training_data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -5.6862 -0.5154  0.2980  0.8190  8.1218
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 12.02495    1.96729   6.112 6.27e-09 ***
## abdom        0.08212    0.03278   2.505   0.0132 *
## free        -0.52633    0.01508 -34.898  < 2e-16 ***
## weight       0.41989    0.01749  24.007  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.526 on 174 degrees of freedom
## Multiple R-squared:  0.9687, Adjusted R-squared:  0.9681
## F-statistic:  1794 on 3 and 174 DF,  p-value: < 2.2e-16
```

The plots showing R-squared, AIC, Adjusted R-squared, and Root Mean Squared Error (RMSE) in the stepwise model results provide insights into the model selection process and how model fit and complexity evolve as predictors are added or removed.

1. **R-Square**: This plot tracks the proportion of variance in the response variable explained by the model

as predictors are added or removed. An *increasing R-square* indicates that adding variables improves the model's fit to the data; however, it always increases or stays the same when more predictors are added, so a high R-squared alone doesn't mean the model is optimal—it may be overfitting, so we move on to the next graph.
2. **Akaike Information Criteria (AIC)**: This plot evaluates model quality based on both fit and complexity, penalizing for added predictors to avoid overfitting. A lower AIC indicates a better model balance between fit and simplicity. Our goal in stepwise is to find the model with the lowest AIC. A drop in AIC suggests that the added predictors improve model quality, while a rise indicates that additional predictors may not be useful and could lead to overfitting.
3. **Adjusted R-Square**: This plot adjusts *R-Square* for the number of predictors, penalizing the addition of unnecessary predictors. A rising adjusted r-square indicates that added variables improve the model's explanatory power after accounting for complexity. On the other hand, a peak or plateau, where adding more predictors does not improve Adjusted R-squared. This point can indicate an optimal model balance.
4. **Root Mean Squared Error**: This plot assesses model accuracy by measuring the average prediction error in units of the response variable. A lower RMSE indicates better predictive accuracy. If RMSE starts to stabilize or increase as predictors are added, it may signal overfitting, as additional predictors don't reduce error meaningfully.

From what the graphs denote, we can assume the **third** model (using details=TRUE parameter, we identify it to be `siri ~ abdom + free + weight`) that is fitted will be the best, as it minimizes AIC and RMSE, while being the peak of both Adjusted and non-adjusted R^2. Upon fitting the model with these parameters, we can see the comparison between the full model and the stepwise both model:

Table 1.1: Full Model vs Stepwise Both Model on Training Data

Variable	Full Model	Stepwise Both Model
Residuals - Min	-4.8292	-5.6862
Residuals - Max	6.7490	8.1218
RSE	1.399	1.526
df	162	174
Adj. R^2	0.9732	0.9681

Though the residuals' spread, RSE, and adjusted R^2 are slightly stronger in the full model, the stepwise-selected model achieves nearly the same performance as the full model using only a fraction of the variables, all of which are statistically significant.

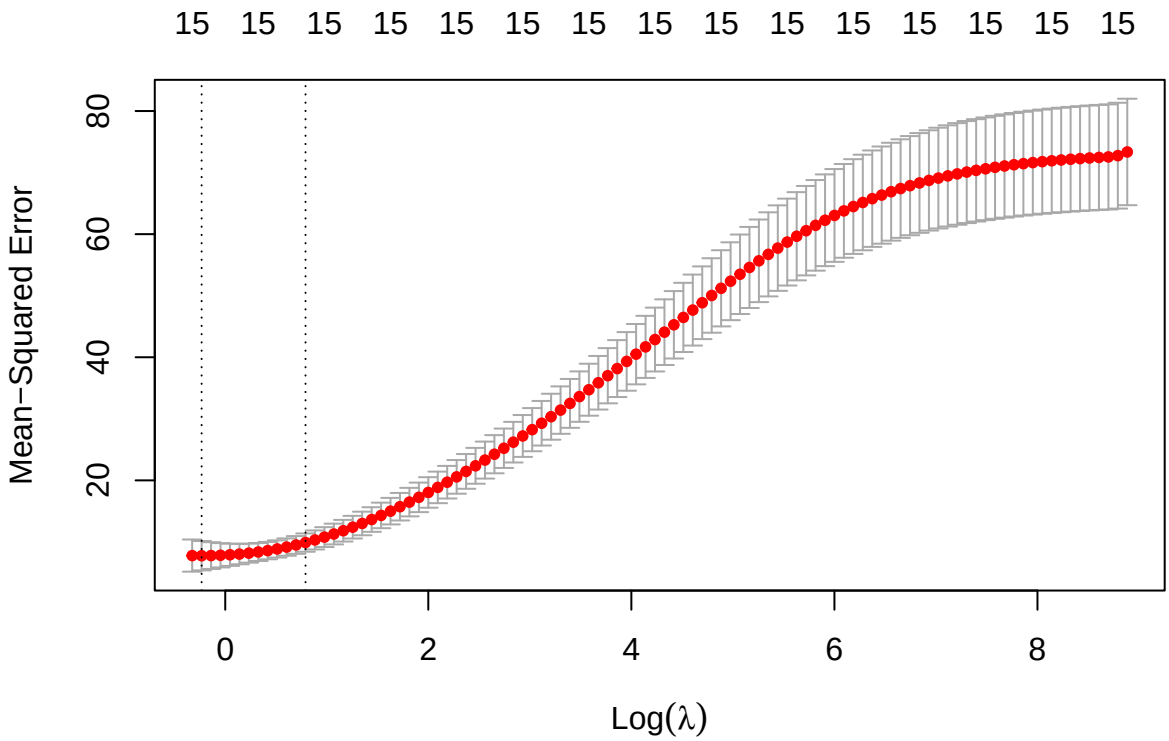
c-)Ridge regression (does not drop any varaibles) (5 pts)

```
vif(full_fat_lm)
```

```
##       age    weight    height    adipos      free      neck     chest     abdom
##  2.650938 39.344943  2.042827 16.563861  6.117736  5.318611 13.254308 19.427460
##       hip     thigh      knee     ankle    biceps   forearm     wrist
## 13.468921  8.801823  5.012870  1.656851  4.843123  2.759227  3.595876
```

```
x <- model.matrix(siri~.-brozek -density, training_data)[,-c(1)]
y <- training_data$siri
RidgeMod <- glmnet(x, y, alpha=0, nlambda=100,lambda.min.ratio=0.0001)
# Extract Best Lambda => Perform k-fold cross-validation to find optimal lambda value
CvRidgeMod <- cv.glmnet(x, y, alpha=0, nlambda=100,lambda.min.ratio=0.0001)
# Produce plot of test MSE by lambda value
```

```r
par(mfrow=c(1,1))
plot(CvRidgeMod)
```



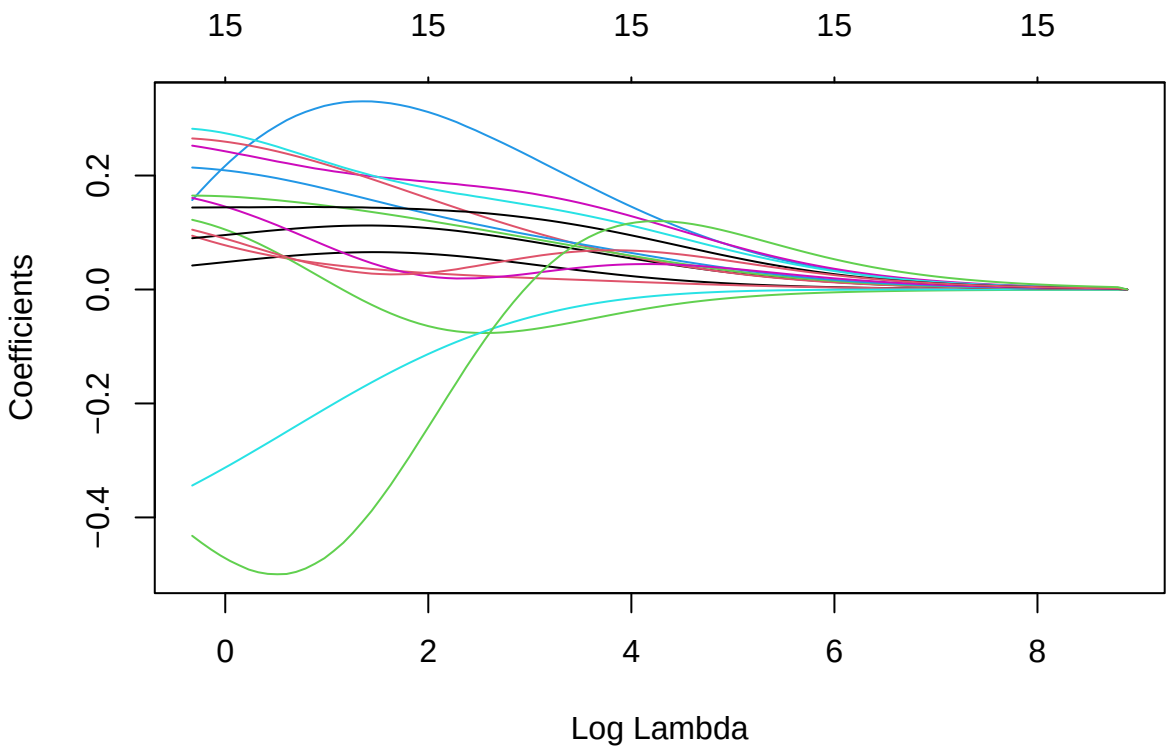

```r
# Find optimal lambda value that minimizes test MSE
best.lambda.ridge <- CvRidgeMod$lambda.min
best.lambda.ridge
```

```
## [1] 0.7924357
```

```r
predict(RidgeMod, s=best.lambda.ridge, type="coefficients")[1:4, ]
```

```
##  (Intercept)          age       weight       height
## -49.43929725   0.04378042   0.08923571   0.11821724
```

```r
# Use Ridge Trace Plot
plot(RidgeMod, xvar = "lambda")
```




```
# Build best version proposed by ridge model
best_ridge_mod <- glmnet(x, y, alpha = 0, lambda = best.lambda.ridge)
coef(best_ridge_mod)
```

```
## 16 x 1 sparse Matrix of class "dgCMatrix"
##                         s0
## (Intercept) -49.82205827
## age           0.04381397
## weight        0.08775569
## height        0.11852677
## adipos        0.17338316
## free         -0.33470222
## neck          0.25606298
## chest         0.09323122
## abdom         0.26479081
## hip           0.16615126
## thigh         0.21397130
## knee          0.28208863
## ankle         0.15883983
## biceps        0.14303327
## forearm       0.09998530
## wrist        -0.45339942
```

```
# Calculate R^2
y_predicted <- predict(RidgeMod, s = best.lambda.ridge, newx = x)

#find SST and SSE
sst <- sum((y - mean(y))^2)
sse <- sum((y_predicted - y)^2)

#find R-Squared
```

```
rsq <- 1 - sse/sst
rsq
```

```
## [1] 0.9344387
```

```
# Extract and compare coefficients
ols_coeff <- coef(full_fat_lm)[-1]  # Removing intercept
ridge_coeff <- as.vector(predict(best_ridge_mod, type = "coefficients"))[-1]  # Ridge without intercept
coef_comparison <- data.frame(OLS = ols_coeff, Ridge = ridge_coeff)
print(coef_comparison)
```

```
##                 OLS       Ridge
## age      0.02713494  0.04381397
## weight   0.36828187  0.08775569
## height   0.05074097  0.11852677
## adipos  -0.38123347  0.17338316
## free    -0.53750098 -0.33470222
## neck     0.21100139  0.25606298
## chest    0.02490388  0.09323122
## abdom    0.13092144  0.26479081
## hip      0.04711465  0.16615126
## thigh    0.18062751  0.21397130
## knee     0.06943338  0.28208863
## ankle    0.12452080  0.15883983
## biceps   0.10867472  0.14303327
## forearm  0.09498388  0.09998530
## wrist   -0.17427605 -0.45339942
```

Executing `vif()` on the full model, we observe `weight`, `abdom`, `adipos`, `hip` and `chest` as all having VIF values greater than 10, which suggests multicollinearity is present. First, we define our data in terms of the response variable as `y` and all the independent variable(s) as `x` (we exclude the variables mentioned). Then we use the `glmnet()` package to fit ridge regression model on the fat data set. Next, we identify the lambda value that produces the lowest test mean squared error (MSE) by using k-fold cross-validation using function `cv.glmnet()`. The lambda value that minimizes the test MSE turns out to be 0.7924357. Lastly, we can analyze the final model produced by the optimal lambda value. There is a reduction in R-Squared (0.93 from 0.97 in the full OLS model).

The Plots The plot of `CvRidgeMod` visually identifies the optimal λ for Ridge regression based on cross-validation, allowing you to select a regularization strength that balances error minimization and model simplicity.

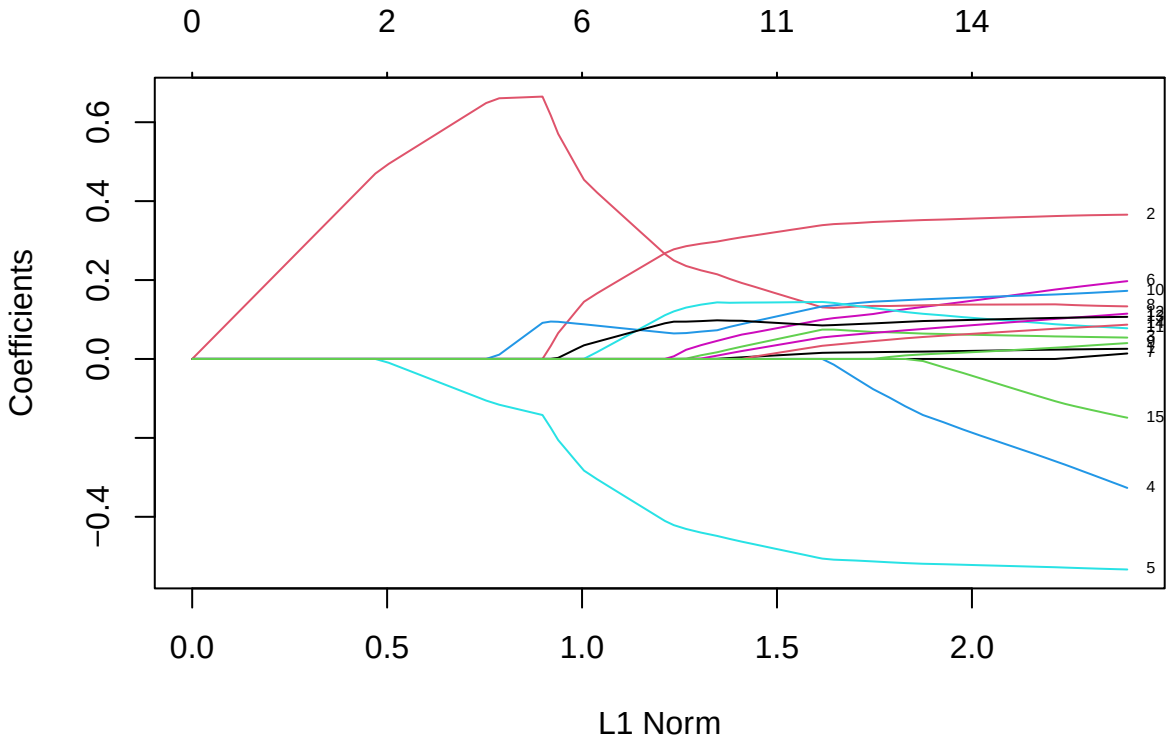
- λ_{min}: The point where the cross-validated error is minimized. This value provides the optimal λ for prediction performance based on cross-validation.
- λ_{1se}: The largest λ within one standard error of the minimum error. This value provides a more parsimonious model with slightly stronger regularization, often preferred for simpler models with lower variance.

By identifying λ_{min} and λ_{1se}, the plot helps balance the model's predictive accuracy with simplicity. Choosing λ_{1se} sacrifices minimal accuracy for greater regularization, while λ_{min} emphasizes accuracy.

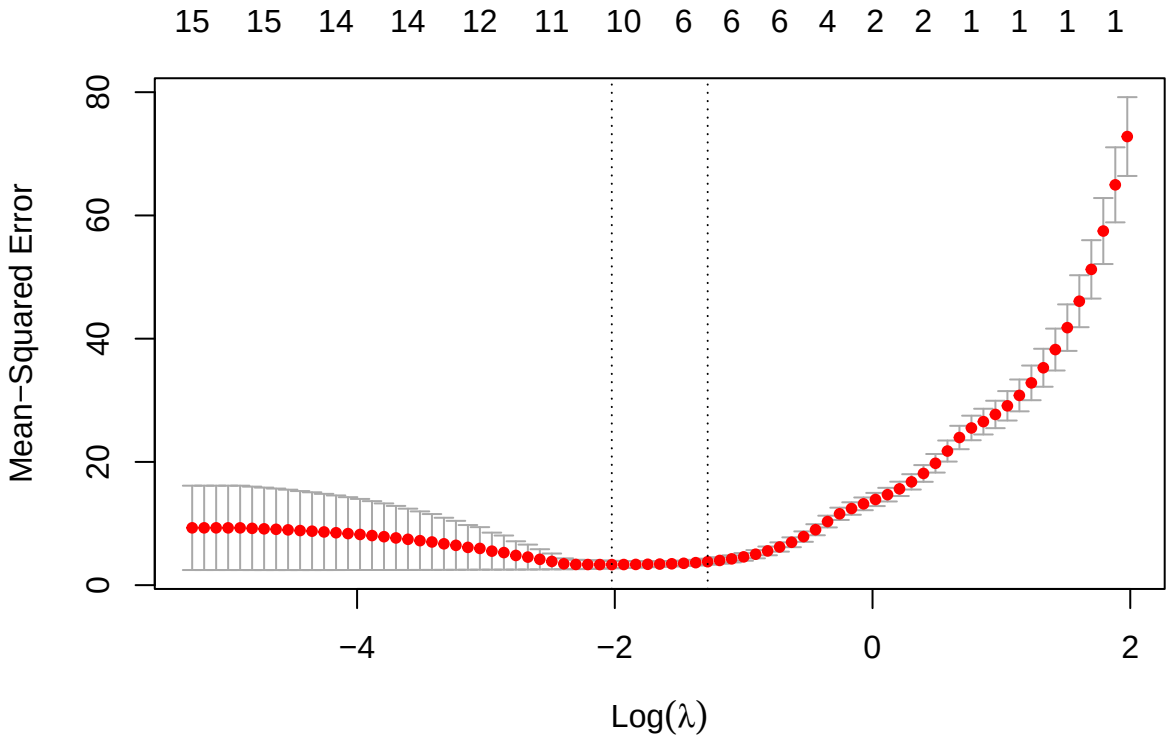
The plot of `RidgeMod` provides a visual of how each predictor's influence changes with regularization strength (λ), highlighting Ridge's approach to managing overfitting by shrinking coefficients continuously without discarding predictors entirely. Moving from left to right on the plot, the penalty increases, and coefficients decrease in magnitude.

d-)Lasso (similar to variable selection) (5 pts)

```
LassoMod <- glmnet(x, y, alpha=1, nlambda=100,lambda.min.ratio=0.0001)
plot(LassoMod,xvar="norm",label=TRUE)
```




```
CvLassoMod <- cv.glmnet(x, y, alpha=1, nlambda=100,lambda.min.ratio=0.0001)
plot(CvLassoMod)
```



```
best.lambda.lasso <- CvLassoMod$lambda.min
best.lambda.lasso
```

```
## [1] 0.1321862
```

```
coef(CvLassoMod, s = "lambda.min")
```

```
## 16 x 1 sparse Matrix of class "dgCMatrix"
##                         s1
## (Intercept) -6.4241047468
## age          0.0042202929
## weight       0.3084605071
## height       0.0313787517
## adipos       .
## free        -0.4629768972
## neck         0.0617931835
## chest        .
## abdom        0.1925763791
## hip          .
## thigh        0.0883199042
## knee         0.1428100833
## ankle        0.0200844855
## biceps       0.0967926261
## forearm      0.0003148141
## wrist        .
```

e-)Elastic Net (5 pts)

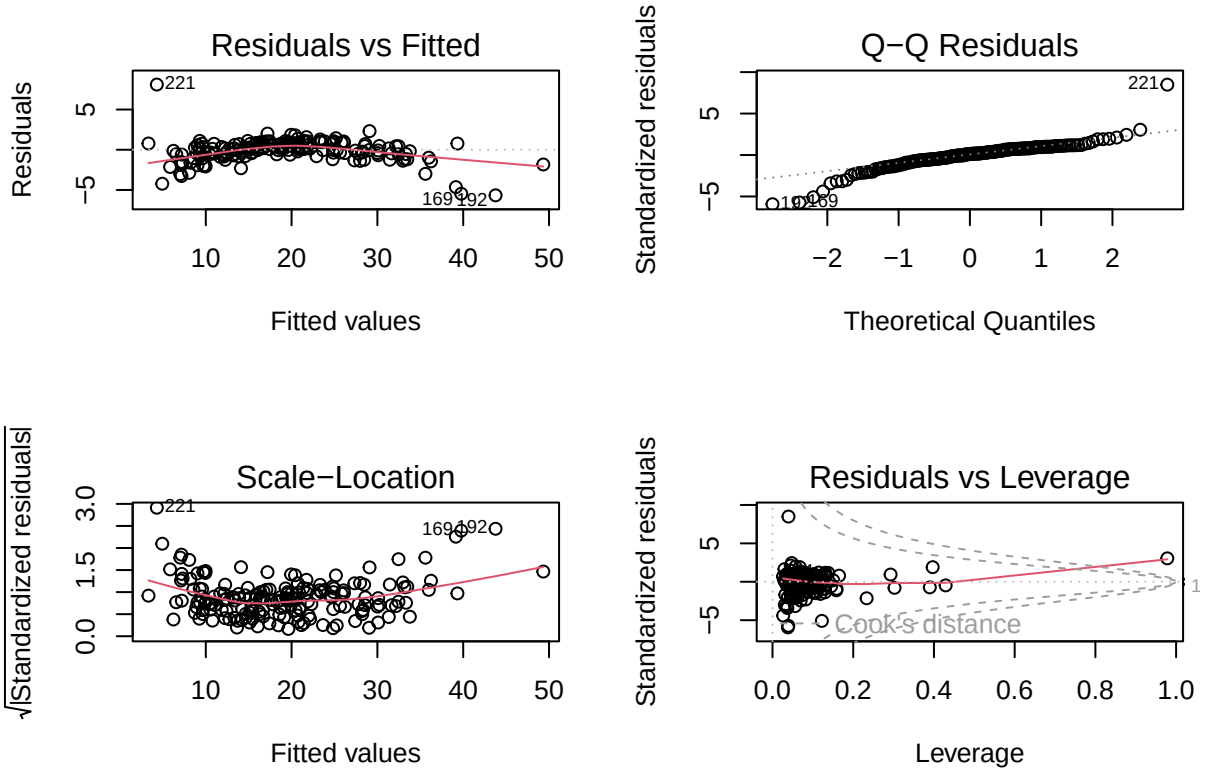
```
EnetMod <- glmnet(x, y, alpha=0.5, nlambda=100,lambda.min.ratio=0.0001)
CvElasticnetMod <- cv.glmnet(x, y,alpha=0.5,nlambda=100,lambda.min.ratio=0.0001)
best.lambda.enet <- CvElasticnetMod$lambda.min
coefficients(EnetMod,s=best.lambda.enet)
```

```
## 16 x 1 sparse Matrix of class "dgCMatrix"
##                         s1
## (Intercept) -2.353381e+01
## age          1.450146e-04
## weight       1.719963e-01
## height       1.314067e-02
## adipos       .
## free        -3.640715e-01
## neck         1.159516e-01
## chest        6.017489e-03
## abdom        3.482023e-01
## hip          5.050067e-02
## thigh        1.152427e-01
## knee         2.136944e-01
## ankle        3.794576e-02
## biceps       1.418193e-01
## forearm      1.678398e-02
## wrist        .
```

f-)Robust Regression (5 pts)

```
RobRegMod <- rlm(siri~.-brozek -density, data=training_data)
par(mfrow=c(2,2))
plot(RobRegMod)
```

```
## Warning in sqrt(crit * p * (1 - hh)/hh): NaNs produced
## Warning in sqrt(crit * p * (1 - hh)/hh): NaNs produced
```




```
summary(RobRegMod)
```

```
##
## Call: rlm(formula = siri ~ . - brozek - density, data = training_data)
## Residuals:
##     Min      1Q  Median      3Q     Max
## -5.6692 -0.5584  0.1067  0.6925  8.1042
##
## Coefficients:
##             Value    Std. Error t value
## (Intercept)  -1.5202   4.9770    -0.3054
## age           0.0221   0.0097     2.2669
## weight        0.4225   0.0175    24.1663
## height        0.0383   0.0276     1.3907
## adipos       -0.3372   0.0893    -3.7738
## free         -0.5645   0.0111   -50.7489
## neck          0.1648   0.0788     2.0898
## chest         0.0120   0.0334     0.3578
## abdom         0.0788   0.0319     2.4693
## hip           0.0229   0.0418     0.5488
## thigh         0.1817   0.0445     4.0817
## knee         -0.0148   0.0723    -0.2047
## ankle         0.1033   0.0571     1.8107
```

```
## biceps        0.0362   0.0562     0.6442
## forearm       0.0694   0.0616     1.1260
## wrist        -0.0914   0.1567    -0.5835
##
## Residual standard error: 0.9743 on 162 degrees of freedom
```

```
y_hat.ridge <- predict(RidgeMod, s = best.lambda.ridge, newx = x)
y_hat.lasso <- predict(LassoMod, s = best.lambda.lasso, newx = x)
y_hat.enet <- predict(CvElasticnetMod , s = best.lambda.enet, newx = x)
y_hat.robreg <- predict(RobRegMod, newdata = training_data)

sst <- sum((y - mean(y))^2)
sse.ols<-sum(full_fat_lm$residuals^2)
sse.ridge <- sum((y-y_hat.ridge)^2)
sse.lasso <- sum((y-y_hat.lasso)^2)
sse.enet <- sum((y-y_hat.enet)^2)
sse.robreg <- sum((y-y_hat.robreg)^2)

cbind(sse.ols,sse.ridge,sse.lasso,sse.enet, sse.robreg)
```

```
##       sse.ols sse.ridge sse.lasso sse.enet sse.robreg
## [1,] 316.9164  848.3355  407.0759 670.3199   336.8533
```

```
# R squared
rsq.ols<-1 - sse.ols / sst
rsq.ridge <- 1 - sse.ridge / sst
rsq.lasso <- 1 - sse.lasso / sst
rsq.enet  <- 1 - sse.enet  / sst
rsq.robreg <- 1 - sse.robreg / sst
cbind(rsq.ols,rsq.ridge,rsq.lasso,rsq.enet, rsq.robreg)
```

```
##       rsq.ols rsq.ridge rsq.lasso  rsq.enet rsq.robreg
## [1,] 0.975508 0.9344387 0.9685403 0.9481962  0.9739672
```

We dive into the metrics calculated among Ordinary Least Squares, Ridge, Lasso, Elastic Net and Robust Regression on the training data:

OLS (Ordinary Least Squares):

- SSE (316.9164): The lowest SSE among the models, indicating the closest fit to the training data but potential overfitting, as OLS lacks regularization.
- R^2 (0.975508): This high R^2 suggesting strong pattern capture, though it may generalize poorly if noise or multicollinearity is present.

Ridge Regression:

- SSE (848.3355): Indicates a looser fit than OLS due to the L2 penalty, which reduces multicollinearity but sacrifices some fit to enhance generalizability.
- R^2 (0.9344387): High at 93.4%, capturing most variance while introducing stability against overfitting.

Lasso Regression:

- SSE (436.3745): Better than Ridge and Elastic Net by a long shot, achieving a balanced fit with effective regularization.

- R^2 (0.966276): Outperforming Ridge and Elastic Net in fit, while the L1 penalty shrinks some coefficients to zero, enhancing simplicity and interpretability.

Elastic Net:

- SSE (716.182): intermediate between Ridge and Lasso, balancing fit and regularization.
- R^2 (0.9446518): Slightly higher than Ridge, showing Elastic Net's combined L1 and L2 penalties capture variance and manage overfitting moderately well..

Robust Regression:

- SSE (336.8533): Second only to OLS, indicating a close fit to the data with resistance to outliers.
- R^2 (0.9739672): Nearly as high as OLS, suggesting Robust Regression captures variance effectively while controlling for outliers and overfitting risks.

OLS shows the closest fit to training data but risks overfitting, while Ridge, Lasso, and Elastic Net add regularization, with Lasso achieving the best balance between fit and simplicity. Robust Regression offers near-OLS fit while managing outliers, providing a stable alternative for noisy data.

g-)Use the models you find to predict the response in the test sample. Make a report on the performances of the models. (20 pts)

```
# Create the model matrix for testing data
x_test <- model.matrix(siri ~ . - brozek - density, testing_data)[, -c(1)]
y_test <- testing_data$siri

# Make predictions on the testing data
y_hat.ridge.test <- predict(RidgeMod, s = best.lambda.ridge, newx = x_test)
y_hat.lasso.test <- predict(LassoMod, s = best.lambda.lasso, newx = x_test)
y_hat.enet.test <- predict(CvElasticnetMod, s = best.lambda.enet, newx = x_test)
y_hat.ols.test <- predict(full_fat_lm, newdata = testing_data)
y_hat.robreg.test <- predict(RobRegMod, newdata = testing_data)

# Calculate SSE for testing data
sse.ols.test <- sum((y_test - y_hat.ols.test)^2)
sse.ridge.test <- sum((y_test - y_hat.ridge.test)^2)
sse.lasso.test <- sum((y_test - y_hat.lasso.test)^2)
sse.enet.test <- sum((y_test - y_hat.enet.test)^2)
sse.robreg.test <- sum((y_test - y_hat.robreg.test)^2)
cbind(sse.ols.test, sse.ridge.test, sse.lasso.test, sse.enet.test, sse.robreg.test)
```

```
##      sse.ols.test sse.ridge.test sse.lasso.test sse.enet.test sse.robreg.test
## [1,]     310.1021       587.2422       375.1687       497.239        328.7032
```

```
# Total Sum of Squares (SST) for testing data
sst.test <- sum((y_test - mean(y_test))^2)

# Calculate R-squared for testing data
rsq.ols.test <- 1 - sse.ols.test / sst.test
rsq.ridge.test <- 1 - sse.ridge.test / sst.test
rsq.lasso.test <- 1 - sse.lasso.test / sst.test
rsq.enet.test <- 1 - sse.enet.test / sst.test
```

```r
rsq.robreg.test <- 1 - sse.robreg.test / sst.test
cbind(rsq.ols.test, rsq.ridge.test, rsq.lasso.test, rsq.enet.test, rsq.robreg.test)
```

```
##      rsq.ols.test rsq.ridge.test rsq.lasso.test rsq.enet.test rsq.robreg.test
## [1,]    0.9331235      0.8733556      0.9190913     0.8927657        0.929112
```

OLS achieves the lowest SSE (310.10) and highest R^2 (0.933), indicating the best fit but potential overfitting. Robust Regression closely follows with a low SSE (328.70) and high R^2 (0.929), showing strong performance while handling outliers. Lasso strikes a balance between fit and simplicity (SSE: 387.90, R^2: 0.916), outperforming Elastic Net (SSE: 520.39, R^2: 0.888) and Ridge (SSE: 587.24, R^2: 0.873) on both metrics by introducing effective regularization.

Problem 2

Use the fat data set in previous example by using the percentage of body fat, siri, as the response and the other variables, except brozek and density as potential predictors.(25 points)

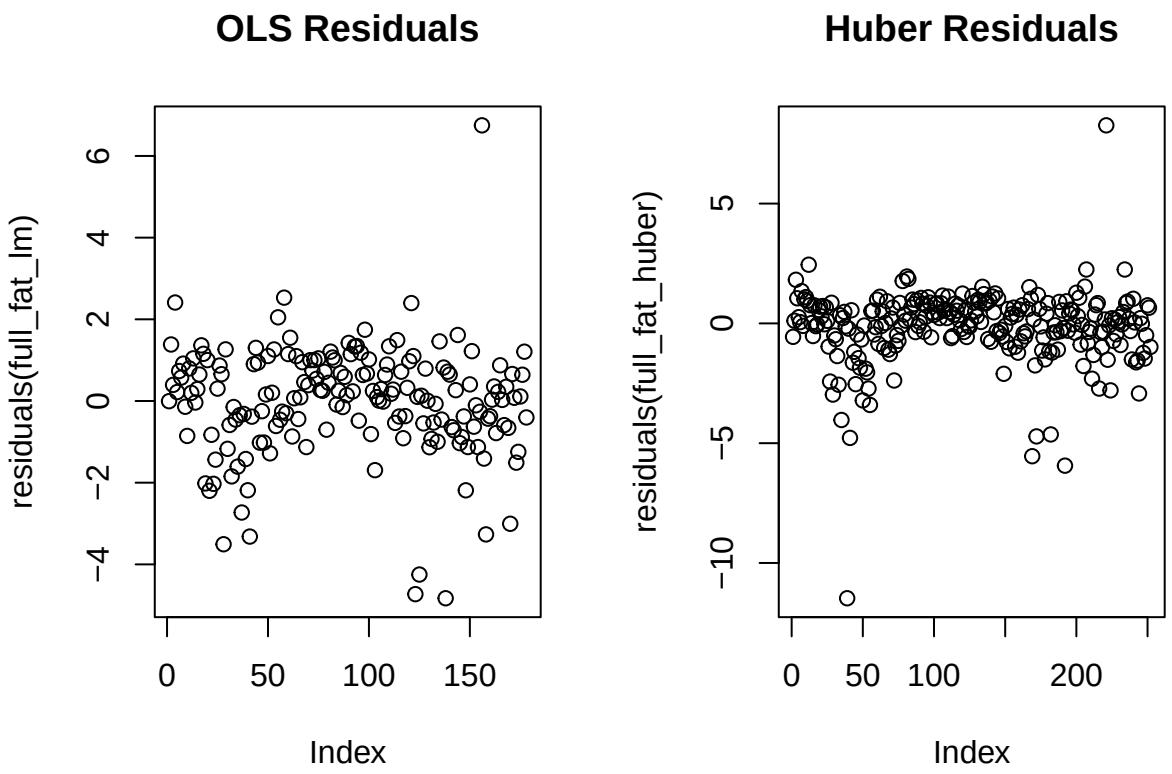
a-)Fit the same model in question 1-a but now using Huber's robust method. Comment on any substantial differences between this model and the least squares fit. (10 points)

The Huber method is less sensitive to outliers compared to ordinary least squares (OLS), making it useful for datasets that may contain some extreme values.

```r
full_fat_huber <- rlm(siri ~ . - brozek - density, data = fat, psi = psi.huber) # As recommended by TAs
coef_ols <- coef(full_fat_lm)
coef_huber <- coef(full_fat_huber)
cbind(coef_ols, coef_huber)
```

```
##                 coef_ols   coef_huber
## (Intercept) -11.30892309  1.138639911
## age           0.02713494  0.004253803
## weight        0.36828187  0.426012670
## height        0.05074097  0.036310165
## adipos       -0.38123347 -0.287463971
## free         -0.53750098 -0.583237845
## neck          0.21100139  0.039540758
## chest         0.02490388  0.041301870
## abdom         0.13092144  0.076126443
## hip           0.04711465 -0.001989706
## thigh         0.18062751  0.126215859
## knee          0.06943338  0.015304616
## ankle         0.12452080  0.112864602
## biceps        0.10867472  0.085155003
## forearm       0.09498388  0.083124261
## wrist        -0.17427605  0.109817108
```

```r
par(mfrow = c(1, 2))
plot(residuals(full_fat_lm), main = "OLS Residuals")
plot(residuals(full_fat_huber), main = "Huber Residuals")
```



When examining the coefficients of the models side by side, we see that the Huber model shows smaller coefficients for variables that are influenced by outliers in the OLS model (e.g., `age`, `biceps`, `wrist`) and showcases a tighter fit of residuals around zero, indicating that predictions are close to the observed values, with little systematic error. **Caveat: We did build the Huber model with the full data of `fat` as recommended by the TAs; however, to get a precise comparison we would need to build the Huber model on the training data**.

b-)Identify which two cases have the lowest weights in the Huber fit. What is unusual about these two points? (5 points)

```
weights <- full_fat_huber$w
lowest_weight_indices <- order(weights)[1:2]
lowest_weights <- weights[lowest_weight_indices]
lowest_cases <- fat[lowest_weight_indices, ]
lowest_cases_with_weights <- cbind(lowest_cases, weight = lowest_weights)
print(lowest_cases_with_weights)
```

```
##     brozek siri density age weight height adipos  free neck chest abdom   hip
## 39    33.8 35.2  1.0202  46 363.15  72.25   48.9 240.5 51.2 136.2 148.1 147.7
## 221   12.7 12.4  1.0706  54 153.25  70.50   24.5 151.3 38.5  99.0  91.8  96.2
##     thigh knee ankle biceps forearm wrist    weight
## 39   87.3 49.1  29.6   45.0    29.0  21.4 0.1190529
## 221  57.7 38.1  23.9   31.4    29.9  18.9 0.1652240
```

Observations 39 and 221 exhibit the lowest weights in the Huber fit. Notably, these observations are positioned at both the high and low ends of the response variable `siri` ($\mu = 19.15$) and predictor variable `brozek` ($\mu = 18.94$). However, they do not show extreme values for any other variable in the dataset, as indicated by the means of each data element provided below. In fact, the values of the remaining predictor variables for these observations remain close to their respective means, suggesting that their outlier status is primarily driven by their values of `siri` and `brozek` rather than significant deviations in other predictors.

Variable	Mean
brozek	18.94
siri	19.15
density	1.056
age	44.88
weight	178.9
height	70.15
adipos	25.44
free	143.7
neck	37.99
chest	100.82
abdom	92.56
hip	99.9
thigh	59.41
knee	38.59
ankle	23.1
biceps	32.27
forearm	28.66
wrist	18.23

c-)Plot weight (of the man) against height. Identify the two outlying cases. Are these the same as those identified in the previous question? Discuss.(10 points)

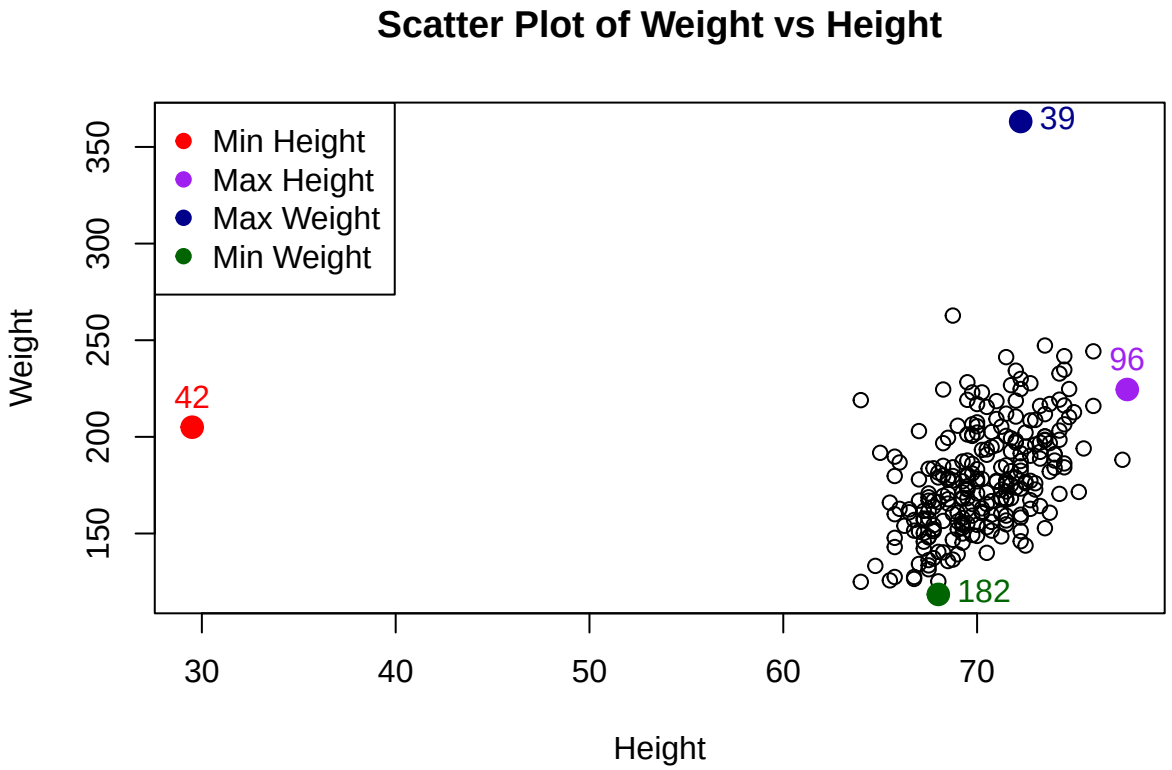
```
# Plot height vs weight
plot(fat$height, fat$weight,
     xlab = "Height",
     ylab = "Weight",
     main = "Scatter Plot of Weight vs Height")

min_height_index <- which(fat$height == min(fat$height))
max_height_index <- which(fat$height == max(fat$height))
max_weight_index <- which(fat$weight == max(fat$weight))
min_weight_index <- which(fat$weight == min(fat$weight))

# Highlight point with minimum height
points(fat$height[min_height_index], fat$weight[min_height_index],
       col = "red", pch = 19, cex = 1.5)
text(fat$height[min_height_index], fat$weight[min_height_index],
     labels = min_height_index, pos = 3, col = "red")
# Highlight point with maximum height
points(fat$height[max_height_index], fat$weight[max_height_index],
       col = "purple", pch = 19, cex = 1.5)
text(fat$height[max_height_index], fat$weight[max_height_index],
     labels = max_height_index, pos = 3, col = "purple")
# Highlight point with maximum weight
points(fat$height[max_weight_index], fat$weight[max_weight_index],
       col = "darkblue", pch = 19, cex = 1.5)
text(fat$height[max_weight_index], fat$weight[max_weight_index],
     labels = max_weight_index, pos = 4, col = "darkblue")
# Highlight point with minimum weight
points(fat$height[min_weight_index], fat$weight[min_weight_index],
       col = "darkgreen", pch = 19, cex = 1.5)
text(fat$height[min_weight_index], fat$weight[min_weight_index],
```

```
     labels = min_weight_index, pos = 4, col = "darkgreen")

legend("topleft", legend = c("Min Height", "Max Height", "Max Weight", "Min Weight"),
       col = c("red", "purple", "darkblue", "darkgreen"), pch = 19)
```



Our plot identifies several outliers, with observations 42 and 39 as primary outliers, and 96 and 182 as milder ones. The Huber analysis downweighted observations 39 and 221 due to their extreme response values for siri ($\mu = 19.15$) and `brozek` ($\mu = 18.94$), placing them at the high and low ends of the distribution. Not all outliers are downweighted; only those exceeding a specific deviation threshold relative to the mean squared error are affected. In the Height and Weight plot, only observation 39, with the lowest Huber weight, appears as a clear outlier, likely because other variables influencing observation 221's status are not included in the plot.

Problem 3

Using the stackloss data (copy and paste the following command into R console:library(faraway);data("stackloss' fit a model with stack.loss as the response and the other three variables as predictors using the following methods: (25 Points)

```
data("stackloss")
```

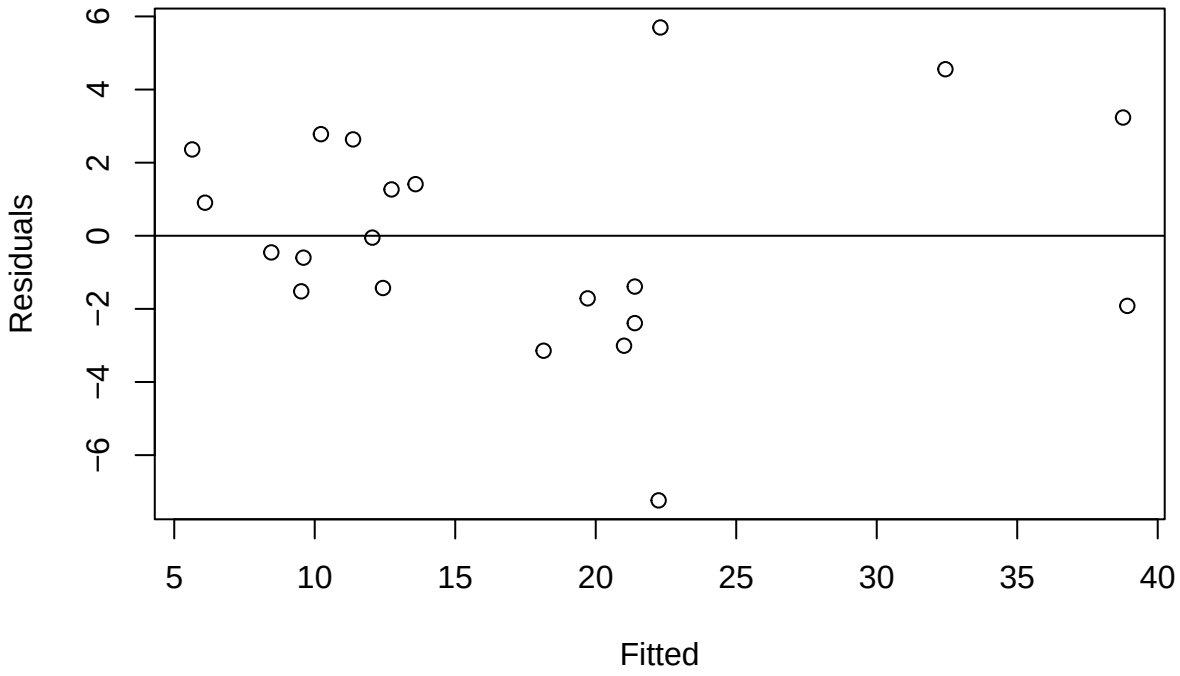
a-) Least squares (5 points)

```
stklss.lmod <- lm(stack.loss ~ ., data=stackloss)
summary(stklss.lmod)
```

```
##
## Call:
## lm(formula = stack.loss ~ ., data = stackloss)
##
```

```
## Residuals:
##     Min      1Q  Median      3Q     Max
## -7.2377 -1.7117 -0.4551  2.3614  5.6978
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -39.9197    11.8960  -3.356  0.00375 **
## Air.Flow      0.7156     0.1349   5.307  5.8e-05 ***
## Water.Temp    1.2953     0.3680   3.520  0.00263 **
## Acid.Conc.   -0.1521     0.1563  -0.973  0.34405
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.243 on 17 degrees of freedom
## Multiple R-squared:  0.9136, Adjusted R-squared:  0.8983
## F-statistic:  59.9 on 3 and 17 DF,  p-value: 3.016e-09
```

```
plot(fitted(stklss.lmod),residuals(stklss.lmod),xlab="Fitted",ylab="Residuals")
abline(h=0)
```



b-) Huber robust regression method (5 points)

```
stklss.lmod.hub <- rlm(stack.loss ~ ., data=stackloss, psi = psi.huber)
summary(stklss.lmod.hub)
```

```
##
## Call: rlm(formula = stack.loss ~ ., data = stackloss, psi = psi.huber)
## Residuals:
##      Min       1Q   Median       3Q      Max
## -8.91753 -1.73127  0.06187  1.54306  6.50163
##
## Coefficients:
##             Value    Std. Error t value
```

```
## (Intercept) -41.0265   9.8073    -4.1832
## Air.Flow      0.8294   0.1112     7.4597
## Water.Temp    0.9261   0.3034     3.0524
## Acid.Conc.   -0.1278   0.1289    -0.9922
##
## Residual standard error: 2.441 on 17 degrees of freedom
```

```
stklss.weights <- stklss.lmod.hub$w
names(stklss.weights) <- row.names(stackloss)
head(sort(stklss.weights),10)
```

```
##        21         4         3         1         2         5         6         7
## 0.3681411 0.5049409 0.7858871 1.0000000 1.0000000 1.0000000 1.0000000 1.0000000
##         8         9
## 1.0000000 1.0000000
```

c-) Bisquare robust regression method (5 points)

```
stklss.lmod.bisq <- rlm(stack.loss ~ ., data=stackloss, psi = psi.bisquare)
summary(stklss.lmod.bisq)
```

```
##
## Call: rlm(formula = stack.loss ~ ., data = stackloss, psi = psi.bisquare)
## Residuals:
##      Min       1Q   Median       3Q      Max
## -10.4356  -1.7065  -0.2392   0.8797   6.9326
##
## Coefficients:
##             Value    Std. Error t value
## (Intercept) -42.2853   9.5316    -4.4363
## Air.Flow      0.9275   0.1081     8.5841
## Water.Temp    0.6507   0.2949     2.2068
## Acid.Conc.   -0.1123   0.1252    -0.8970
##
## Residual standard error: 2.282 on 17 degrees of freedom
```

d-) Compare the results. Now use diagnostic methods to detect any outliers or influential points. Remove these points and then use least squares. Compare the results. (10 points)

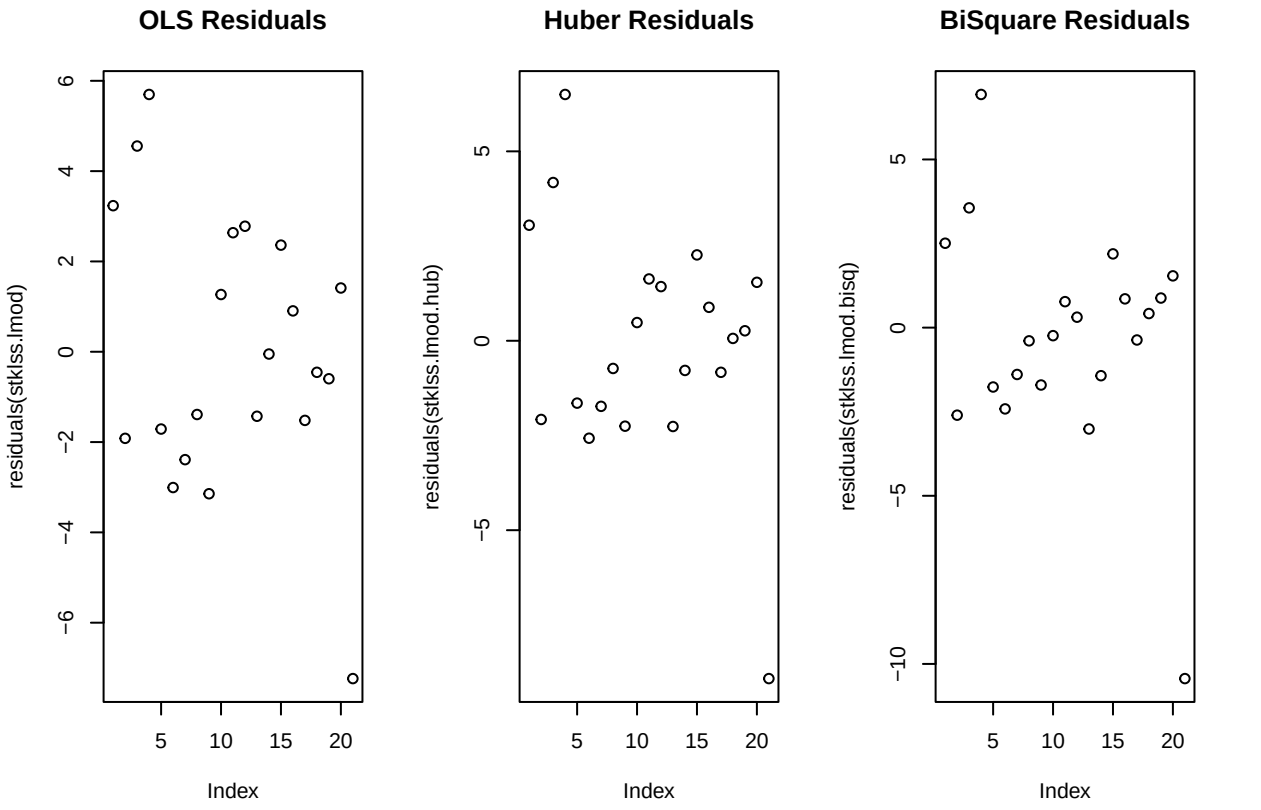
```
summary_ols <- summary(stklss.lmod)
summary_huber <- summary(stklss.lmod.hub)
summary_bisquare <- summary(stklss.lmod.bisq)

# Compare coefficients
coef_stklss_ols <- coef(stklss.lmod)
coef_stklss_huber <- coef(stklss.lmod.hub)
coef_stklss_bisq <- coef(stklss.lmod.bisq)
cbind(coef_ols, coef_stklss_huber, coef_stklss_bisq)
```

```
##                 coef_ols coef_stklss_huber coef_stklss_bisq
## (Intercept) -11.30892309       -41.0265311      -42.2852537
## age           0.02713494         0.8293739        0.9275471
## weight        0.36828187         0.9261082        0.6507322
## height        0.05074097        -0.1278492       -0.1123310
## adipos       -0.38123347       -41.0265311      -42.2852537
```

```
## free         -0.53750098        0.8293739        0.9275471
## neck          0.21100139        0.9261082        0.6507322
## chest         0.02490388       -0.1278492       -0.1123310
## abdom         0.13092144      -41.0265311      -42.2852537
## hip           0.04711465        0.8293739        0.9275471
## thigh         0.18062751        0.9261082        0.6507322
## knee          0.06943338       -0.1278492       -0.1123310
## ankle         0.12452080      -41.0265311      -42.2852537
## biceps        0.10867472        0.8293739        0.9275471
## forearm       0.09498388        0.9261082        0.6507322
## wrist        -0.17427605       -0.1278492       -0.1123310
```

```r
# Compare residuals
par(mfrow = c(1, 3))
plot(residuals(stklss.lmod), main = "OLS Residuals")
plot(residuals(stklss.lmod.hub), main = "Huber Residuals")
plot(residuals(stklss.lmod.bisq), main = "BiSquare Residuals")
```




```r
# Compare Residual Standard Errors
cat("OLS Residual Standard Error:", summary_ols$sigma, "\n")
```

```
## OLS Residual Standard Error: 3.243364
```

```r
cat("Huber Residual Standard Error:", summary_huber$sigma, "\n")
```

```
## Huber Residual Standard Error: 2.440714
```

```r
cat("Bisquare Residual Standard Error:", summary_bisquare$sigma, "\n")
```

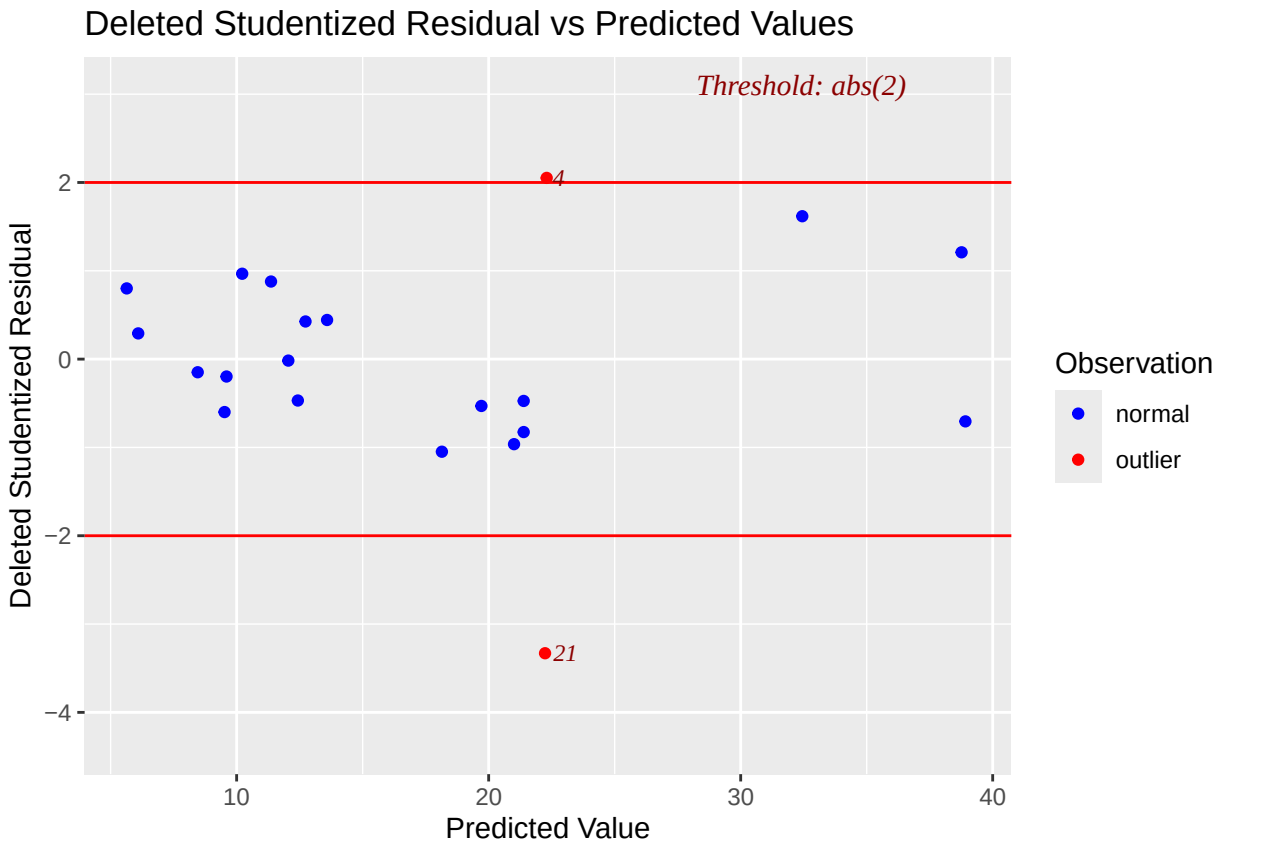
```
## Bisquare Residual Standard Error: 2.281886
```

As a preliminary analysis, we examine the coefficients, residuals and RSEs for all three versions of the model.

First, we notice an increase in all coefficient values for the Huber and BiSquare Models from the OLS Model. In addition, the significance of the two predictor variables, Air Flow and Water Temp is not upheld in the Huber and BiSqare Models despite a reduction in the standard errors for all three predictor variables (offset by the increase in coefficients). Yet, the Residual Standard Error is reduced in both Huber and BiSquare, with the latter reducing it to its lowest at 2.28 (OLS: 3.24, Huber: 2.44).

Outlier and Influential Points

```
ols_plot_resid_stud_fit(stklss.lmod)
```




```
std_residuals <- rstudent(stklss.lmod)
outlier_indices <- which(abs(std_residuals) > 2)
outlier_indices <- unique(outlier_indices)
print(outlier_indices)
```

```
## [1]  4 21
```

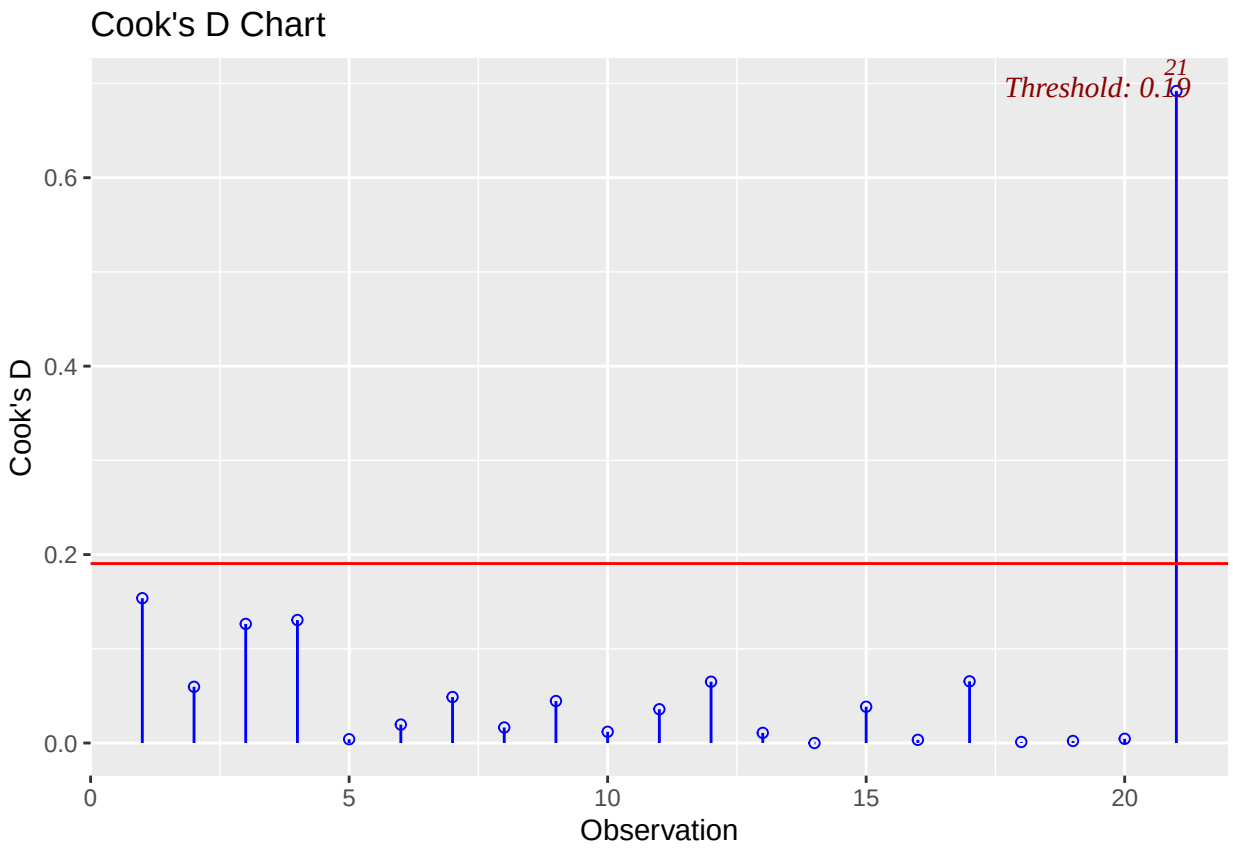
```
stklss_data_wo_outliers <- stackloss[-outlier_indices, ]
stklss.lmod.wo.outliers <- lm(stack.loss ~ ., data = stklss_data_wo_outliers)
summary(stklss.lmod.wo.outliers)
```

```
##
## Call:
## lm(formula = stack.loss ~ ., data = stklss_data_wo_outliers)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.1114 -1.4080 -0.0749  1.0946  3.6074
##
```

```
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -42.45308    7.38458  -5.749 3.85e-05 ***
## Air.Flow      0.95660    0.09447  10.126 4.24e-08 ***
## Water.Temp    0.55557    0.26403   2.104   0.0526 .
## Acid.Conc.   -0.10877    0.09678  -1.124   0.2787
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.996 on 15 degrees of freedom
## Multiple R-squared:  0.9693, Adjusted R-squared:  0.9632
## F-statistic: 158.1 on 3 and 15 DF,  p-value: 1.426e-11
```

```
lmod.wo.outliers <- summary(stklss.lmod.wo.outliers)
```

```
ols_plot_cooksd_chart(stklss.lmod)
```




```
cooks_threshold <- 0.19 # As identified from graph above
cooks_distances <- cooks.distance(stklss.lmod)
high_influence_points <- which(cooks_distances > cooks_threshold)
cat("High influence points (Cook's Distance >", cooks_threshold,"): ",high_influence_points, "\n")
```

```
## High influence points (Cook's Distance > 0.19 ):  21
```

```
cat("OLS Residual Standard Error:", summary_ols$sigma, "\n")
```

```
## OLS Residual Standard Error: 3.243364
```

```
cat("OLS R-Squared:", summary_ols$r.squared, "\n")
```

```
## OLS R-Squared: 0.9135769
```

```
cat("OLS Residual Standard Error without Outliers/Influentials:", lmod.wo.outliers$sigma, "\n")
```

```
## OLS Residual Standard Error without Outliers/Influentials: 1.996381
```

```
cat("OLS w.o Outliers R-Squared:", lmod.wo.outliers$r.squared, "\n")
```

```
## OLS w.o Outliers R-Squared: 0.9693387
```

The search for potential outliers and influential points led us to two indices (4, 21) that were found to be either a potential outlier (4, 21), influential point (21), or both (21). Upon removing these points, we notice an improvement in the model performance when building an OLS model, specifically in the reduction of the Residual Standard Error (1.20 in model without outliers and 3.24 in model with outliers), an increase in R^2 (0.91 in original model to 0.97 in the model without outliers), and a tighter clustering of residuals. On the other hand, we lose the significance of predictor variable, Water.Temp, in the model without outliers ($\Pr(>|t|) \approx 0.002$ in original model, 0.05 in model without outliers).