1.) We will import the Boston Housing Dataset from the ISLR package. After importing, we will do some preliminary Exploratory Data Analysis (EDA) by checking for things like -> presence of blank or 'NA' values, going through the summary of the entire dataset to visually single out values with a huge range or presence of outliers. We will also see the metadata of the dataset.

boston <- Boston
# View(boston)
contents(boston) # View the metadata
names(boston) # View the column names
str(boston) # View the structure of the data frame
sum(is.na(boston))
head(boston)

```
> contents(boston)

Data frame:boston       506 observations and 14 variables    Maximum # NAs:0


        Storage
crim     double
zn       double
indus    double
chas    integer
nox      double
rm       double
age      double
dis      double
rad     integer
tax      double
ptratio  double
black    double
lstat    double
medv     double
> names(boston)
 [1] "crim"    "zn"      "indus"   "chas"    "nox"     "rm"      "age"     "dis"     "rad"     "tax"
[11] "ptratio" "black"   "lstat"   "medv"
> str(boston)
'data.frame':   506 obs. of  14 variables:
 $ crim   : num  0.00632 0.02731 0.02729 0.03237 0.06905 ...
 $ zn     : num  18 0 0 0 0 0 12.5 12.5 12.5 12.5 ...
 $ indus  : num  2.31 7.07 7.07 2.18 2.18 2.18 7.87 7.87 7.87 7.87 ...
 $ chas   : int  0 0 0 0 0 0 0 0 0 0 ...
 $ nox    : num  0.538 0.469 0.469 0.458 0.458 0.458 0.524 0.524 0.524 0.524 ...
 $ rm     : num  6.58 6.42 7.18 7 7.15 ...
 $ age    : num  65.2 78.9 61.1 45.8 54.2 58.7 66.6 96.1 100 85.9 ...
 $ dis    : num  4.09 4.97 4.97 6.06 6.06 ...
 $ rad    : int  1 2 2 3 3 3 5 5 5 5 ...
 $ tax    : num  296 242 242 222 222 222 311 311 311 311 ...
 $ ptratio: num  15.3 17.8 17.8 18.7 18.7 18.7 15.2 15.2 15.2 15.2 ...
 $ black  : num  397 397 393 395 397 ...
 $ lstat  : num  4.98 9.14 4.03 2.94 5.33 ...
 $ medv   : num  24 21.6 34.7 33.4 36.2 28.7 22.9 27.1 16.5 18.9 ...
>
> sum(is.na(boston))
[1] 0
```

We see that the dataset is composed of 506 observations within 14 fields/variables. Also, no blanks or NA values are present.

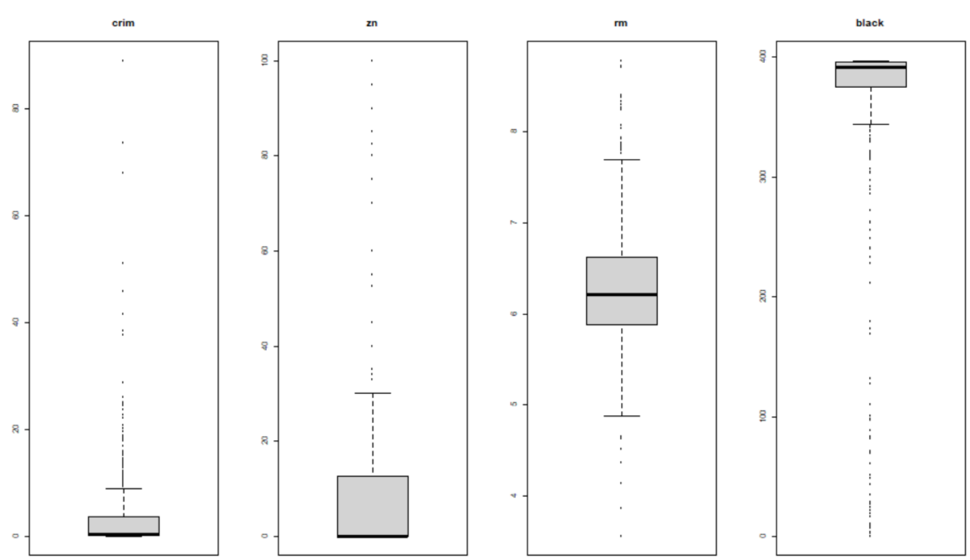
We will now see the summary statistics of the dataset for further analysis.

summary(boston)

```
> summary(boston)
      crim                zn             indus            chas              nox               rm
 Min.   : 0.00632   Min.   :  0.00   Min.   : 0.46   Min.   :0.00000   Min.   :0.3850   Min.   :3.561
 1st Qu.: 0.08205   1st Qu.:  0.00   1st Qu.: 5.19   1st Qu.:0.00000   1st Qu.:0.4490   1st Qu.:5.886
 Median : 0.25651   Median :  0.00   Median : 9.69   Median :0.00000   Median :0.5380   Median :6.208
 Mean   : 3.61352   Mean   : 11.36   Mean   :11.14   Mean   :0.06917   Mean   :0.5547   Mean   :6.285
 3rd Qu.: 3.67708   3rd Qu.: 12.50   3rd Qu.:18.10   3rd Qu.:0.00000   3rd Qu.:0.6240   3rd Qu.:6.623
 Max.   :88.97620   Max.   :100.00   Max.   :27.74   Max.   :1.00000   Max.   :0.8710   Max.   :8.780
      age              dis              rad              tax           ptratio          black
 Min.   :  2.90   Min.   : 1.130   Min.   : 1.000   Min.   :187.0   Min.   :12.60   Min.   :  0.32
 1st Qu.: 45.02   1st Qu.: 2.100   1st Qu.: 4.000   1st Qu.:279.0   1st Qu.:17.40   1st Qu.:375.38
 Median : 77.50   Median : 3.207   Median : 5.000   Median :330.0   Median :19.05   Median :391.44
 Mean   : 68.57   Mean   : 3.795   Mean   : 9.549   Mean   :408.2   Mean   :18.46   Mean   :356.67
 3rd Qu.: 94.08   3rd Qu.: 5.188   3rd Qu.:24.000   3rd Qu.:666.0   3rd Qu.:20.20   3rd Qu.:396.23
 Max.   :100.00   Max.   :12.127   Max.   :24.000   Max.   :711.0   Max.   :22.00   Max.   :396.90
     lstat            medv
 Min.   : 1.73   Min.   : 5.00
 1st Qu.: 6.95   1st Qu.:17.02
 Median :11.36   Median :21.20
 Mean   :12.65   Mean   :22.53
 3rd Qu.:16.95   3rd Qu.:25.00
 Max.   :37.97   Max.   :50.00
```

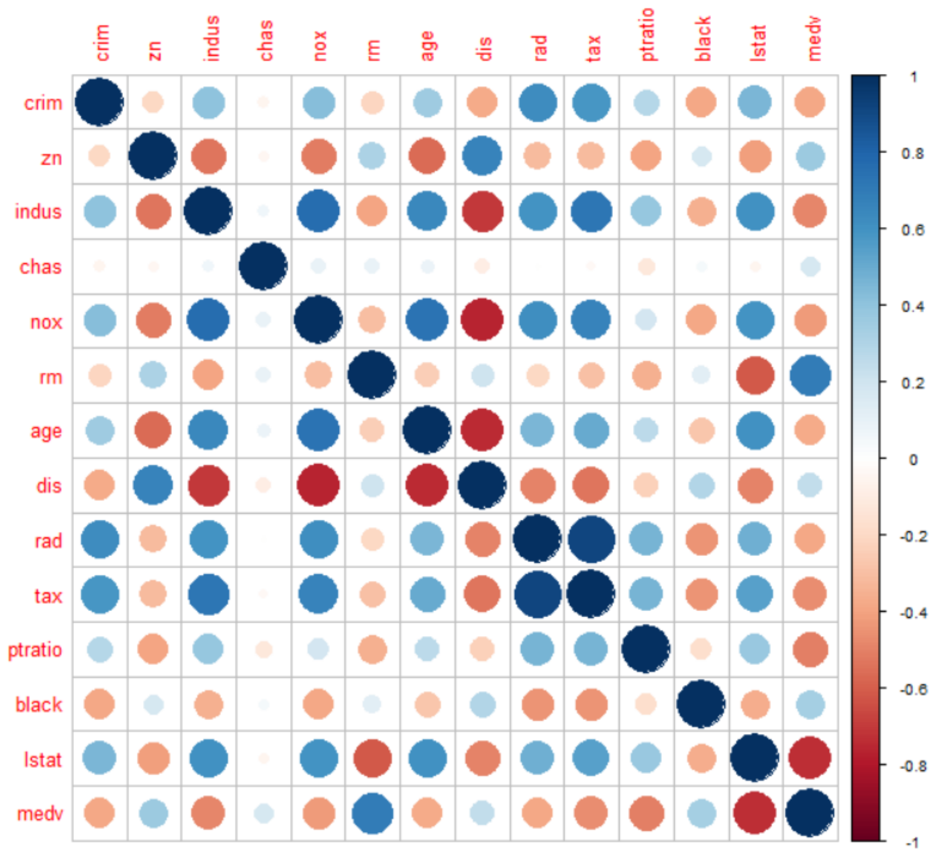
We see that the fields – zn, crim, black, and rm have a huge range along with significant differences in their mean and median. This indicates the presence of outliers. We can verify the same by looking at the boxplots of these variables below.

```
par(mfrow = c(1,4))
boxplot(boston$crim, main = "crim")
boxplot(boston$zn, main = "zn")
boxplot(boston$rm, main = "rm")
boxplot(boston$black, main = "black")
```



As expected, there are a lot of outliers in these four variables, especially in crim, zn, and black. We will now see the linear correlations between variables. This will give us a fair bit of hint regarding the presence of multi-collinearity. To get a better idea for multi-collinearity we can also perform the Farrar-Glauber Test.

corrplot(cor(boston))



We see that there are some high degrees of correlation between the fields. Example – 'rad' and 'tax' have a high positive correlation which makes sense as properties on or near the highways may have higher sale value owing to ease of access. Conversely, there seems to be a high negative correlation between the variables 'nox' and 'dis' which is again logical as the concentration of pollutants tends to be higher in urban work-centers rather than sub-urban areas.

2.) In this dataset – Y (Dependent Variable) will be 'medv' i.e the median price/property value of the house that needs to be predicted.

X(Independent Variables) will be the remaining variables. However, the number of independent variables will vary across models and may have different interactions in each model.

3.)

i.) **<u>Multiple Linear Regression</u>** -> I will be pasting only the parsimonious model's code here for ease.

```
# We will now include the following interactions -> rm*lstat, rm*rad, and lstat*rad to see if
# a parsimonious model is possible. We will drop crim, zn, and black altogether owing to huge
# variances present internally amongst the variables.

multiple_lm_4<-
lm(medv~crim+chas+nox+rm+dis+rad+tax+ptratio+lstat+rm*lstat+rm*rad+lstat*rad, data =
train)
summary(multiple_lm_4)
residuals <- data.frame('Residuals' = multiple_lm_4$residuals)
res_hist <- ggplot(residuals, aes(x=Residuals)) + geom_histogram(color='black', fill='red') +
ggtitle('Histogram of Residuals')
res_hist
par(mfrow=c(2,2))
plot(multiple_lm_4)
glance(multiple_lm_4)
mlm4_mse <- mean(residuals(multiple_lm_4)^2)
mlm4_mse
mlm4_rmse <- sqrt(mlm4_mse)
mlm4_rmse
mlm4_rss <- sum(residuals(multiple_lm_4)^2)
mlm4_rss
mlm4_rse <- sqrt(mlm4_rss/341)
mlm4_rse

# As the fourth model is the better of the four models, we'll consider it as the parsimonious
model and
# use it to predict the testing data set

fit_predict <- predict(multiple_lm_4,test)
summary(fit_predict)
fit_ssl <- sum((test$medv-fit_predict)^2)
sprintf("SSL/SSR/SSE: %f", fit_ssl)
fit_test_mse <- fit_ssl/nrow(test)
sprintf("MSE: %f", fit_test_mse)
fit_test_rmse <- sqrt(fit_test_mse)
sprintf("RMSE: %f", fit_test_mse)

# Let us create a new column to store the predicted prices/property values in the testing data
set
test$pred_price <- fit_predict
pred_plot <- test %>% ggplot(aes(medv,pred_price)) + geom_point(alpha = 0.75) +
geom_smooth(method = "loess") + stat_smooth(aes(color = "black")) + xlab("Actual Property
Value") + ylab("Predicted Property Value")
```

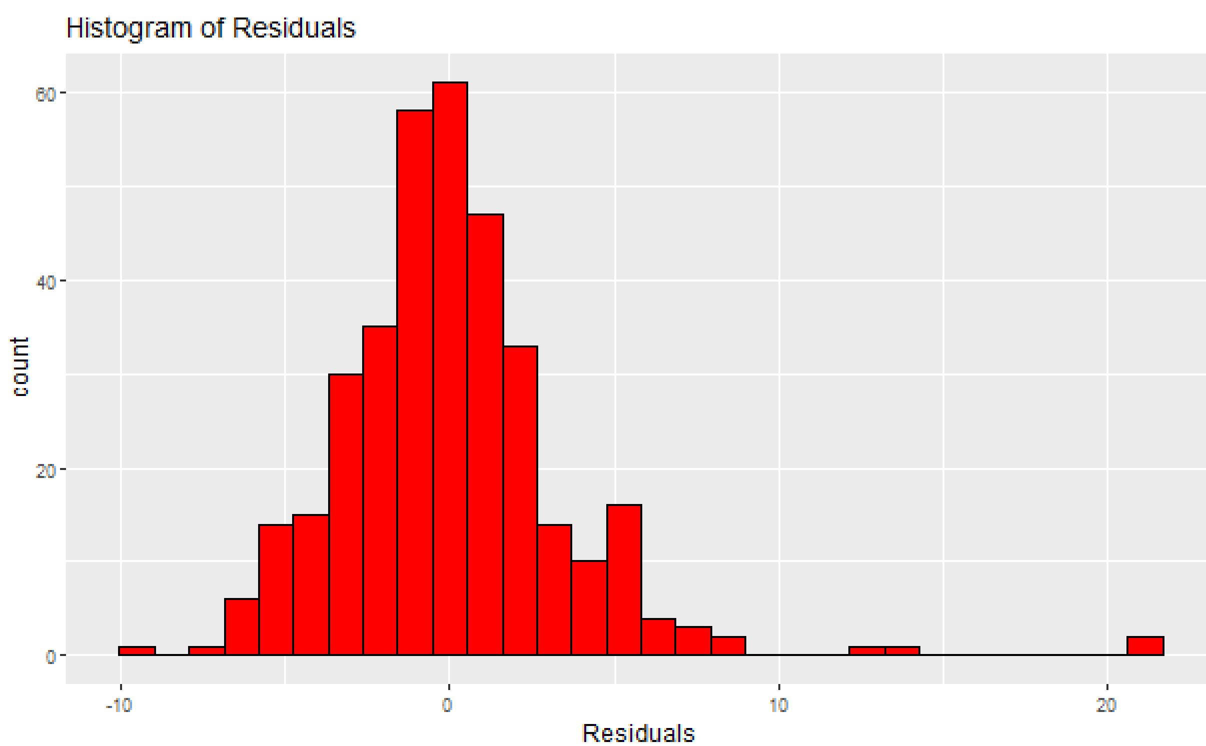
pred_plot

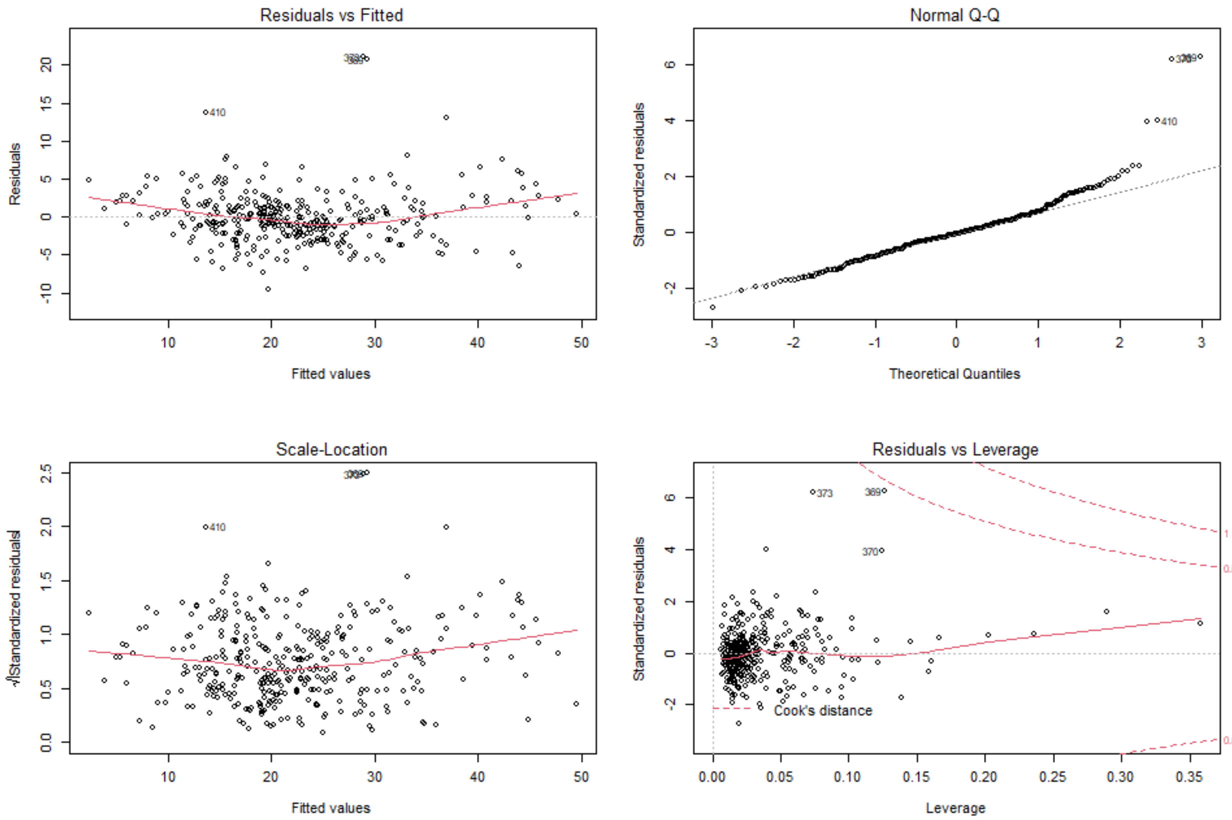
The parsimonious model has a lower value for all the relevant metrics – RMSE, AIC, and BIC. It also has a higher R^2 and F-statistic value.

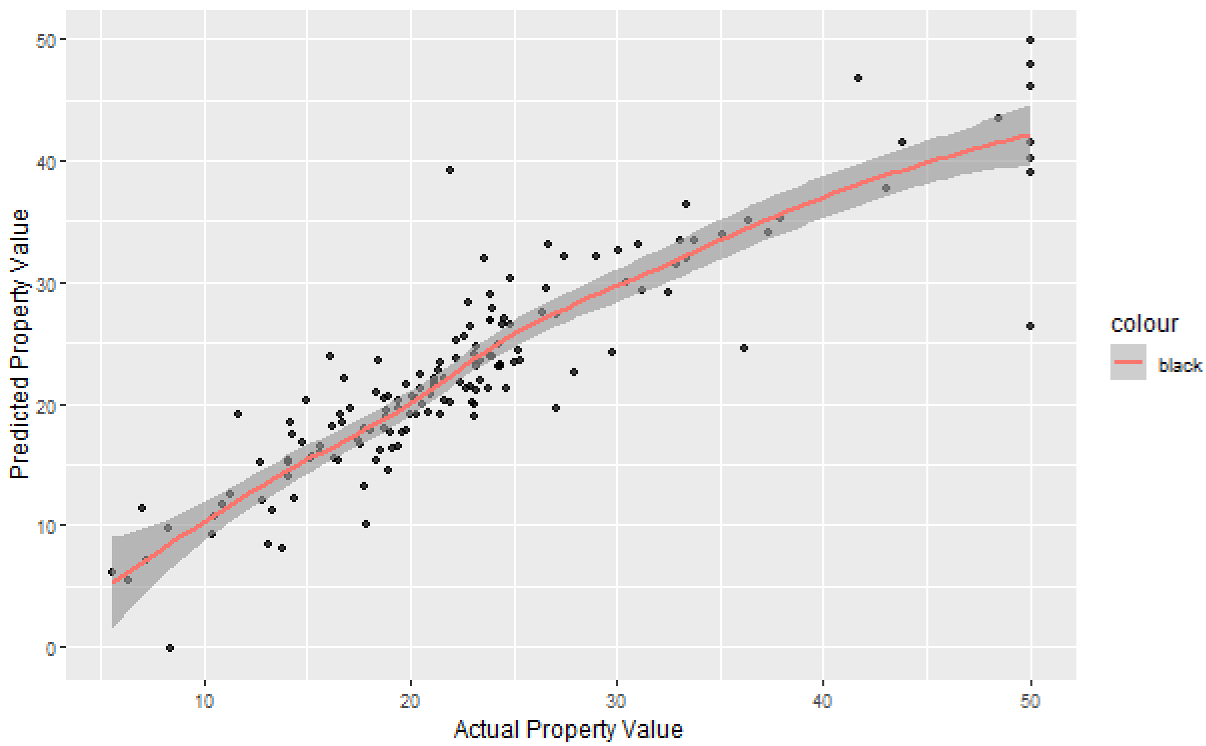
The interaction terms were used basis the correlation plot created above which showed –

a.) Rm and medv have a high positive correlation and as we have seen rm also has a lot of outliers. Taking it into an interaction term can also help us deal with heteroscedasticity.
b.) Rm and lstat have high negative correlation which makes sense given that the lower the population status of the neighborhood, the smaller will be the house size and thus, the lesser number of rooms.
c.) Rm and rad have a low but noticeable negative correlation. Given the previous correlation, it is likely that small houses belong to poorer sections of the population who might have a relatively difficult time accessing infrastructure and thus have lower property values.
d.) Lstat and rad have a noticeably high positive correlation which runs counter to the above explanation. This means that poorer residents tend to be living closer to radial highways and other infrastructure which may contribute to things like noise etc. and thus reduce the property value owing to a lower standard of living. Including this could help capture the variance in the dataset to a higher degree.

Now, we could have also removed rad and tax variables from the model but doing that led to a negligible increase in the R^2 value.




Residuals vs Fitted
Normal Q-Q
Scale-Location
Residuals vs Leverage
Residuals
Fitted values
Standardized residuals
Theoretical Quantiles
√|Standardized residuals|
Leverage
Cook's distance
410
373
369
370



By looking at the above plot we can say that our model has done a pretty good job of predicting the property values.

Mathematically,

Medv = -26.88 + (-0.115) * crim + 2.73 * chas + (-13.20) * nox + 11.85 * rm + (-0.74) * dis + 2.71 * rad + (-0.008) * tax + (-0.625) * ptratio + 1.83 * lstat + (-0.327) * (rm:lstat) + (-0.314) * (rm:rad) + (-0.035) * (rad:lstat)

Before carrying out both Ridge and Lasso Regression we'd need to do some preparatory steps to separate the dependent and independent variables. We'll be doing them below:

grid <- 10 ^ seq(6, -3, length = 10)

Independent/Action Variables

x <- model.matrix(medv~., boston)[,-1] #-1 is to remove the Intercept column which auto-creates

upon running the model for the first time

head(x)

#Dependent/Response Variable

y <- boston$medv

ii.) Ridge Regression ->

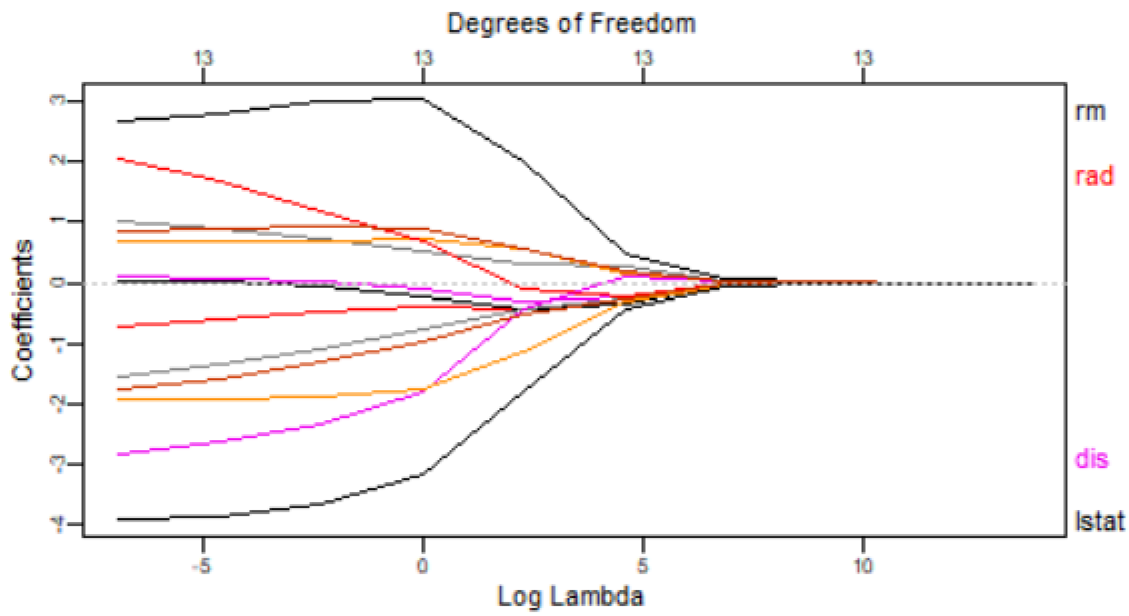
Perform the first ridge regression with a random lambda function obtained from the grid

ridge_mod <- glmnet(scale(x), y, alpha = 0, lambda = grid, thresh = 1e-2, standardize = TRUE)

P.S. - I need help in understanding the output of the below two lines

coef(ridge_mod)

plot_glmnet(ridge_mod, xvar = "lambda", label = 4)

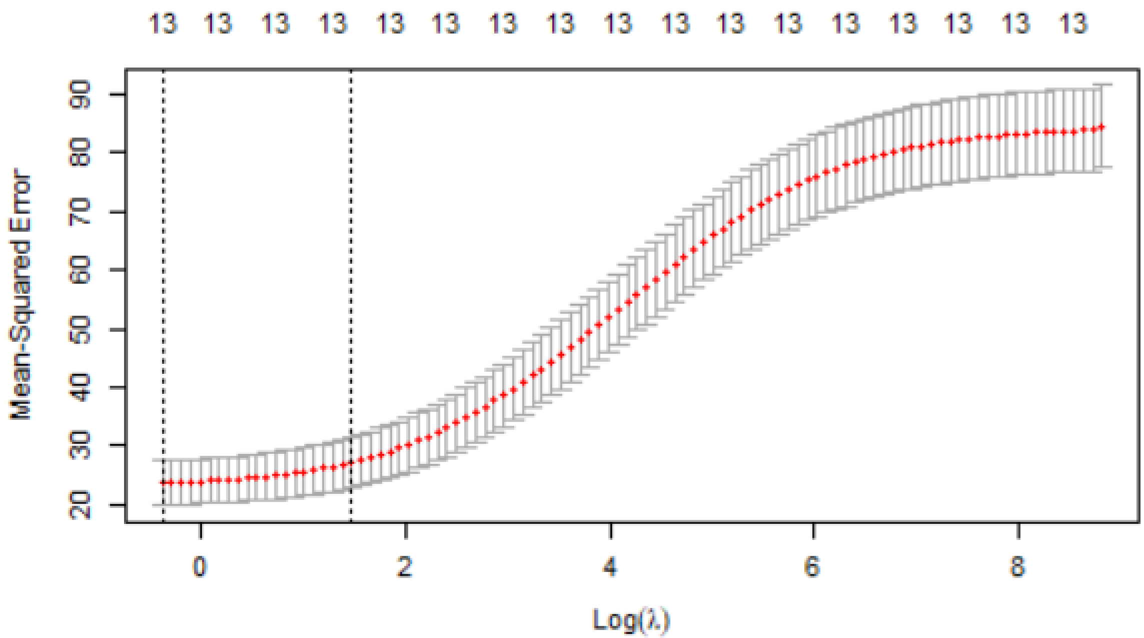


plot_glmnet(ridge_mod, xvar = "lambda", label = 2) # Optional Step. Only minor changes observed by changing the value of the label argument

cv_ridge <- cv.glmnet(scale(x), y, alpha = 0, nfolds = 10)

cv_ridge

plot(cv_ridge)




```
best_lambda_ridge <- cv_ridge$lambda.1se

best_lambda_ridge

ridge_mod_final <- glmnet(scale(x), y, alpha = 0, lambda = best_lambda_ridge, thresh = 1e-
2, standardize = TRUE)

predict(ridge_mod_final, type = "coefficients", s = best_lambda_ridge)

ridge_pred <- predict(ridge_mod_final, s=best_lambda_ridge, newx = scale(x))

sprintf("MSE: %f", mean((ridge_pred - y)^2))

sprintf("RMSE: %f", sqrt(mean((ridge_pred - y)^2)))
```

```
> cv_ridge <- cv.glmnet(scale(x), y, alpha = 0, nfolds = 10)
> cv_ridge

Call:  cv.glmnet(x = scale(x), y = y, nfolds = 10, alpha = 0)

Measure: Mean-Squared Error

    Lambda Index Measure    SE Nonzero
min  0.678   100   23.66 3.709      13
1se  4.357    80   27.17 4.325      13
> plot(cv_ridge)
> best_lambda_ridge <- cv_ridge$lambda.1se
> best_lambda_ridge
[1] 4.356725
> ridge_mod_final <- glmnet(scale(x), y, alpha = 0, lambda = best_lambda_ridge, thresh = 1e-2, standardize = T
RUE)
> predict(ridge_mod_final, type = "coefficients", s = best_lambda_ridge)
14 x 1 sparse Matrix of class "dgCMatrix"
                    s1
(Intercept) 22.5328063
crim        -0.4247083
zn           0.2690482
indus       -0.3797568
chas         0.7028395
nox         -0.6933039
rm           2.6364177
age         -0.4069157
dis         -0.9666998
rad          0.1538589
tax         -0.5124996
ptratio     -1.4767120
black        0.7185586
lstat       -2.3070190
> ridge_pred <- predict(ridge_mod_final, s=best_lambda_ridge, newx = scale(x))
> sprintf("MSE: %f", mean((ridge_pred - y)^2))
[1] "MSE: 26.301543"
> sprintf("RMSE: %f", sqrt(mean((ridge_pred - y)^2)))
[1] "RMSE: 5.128503"
```

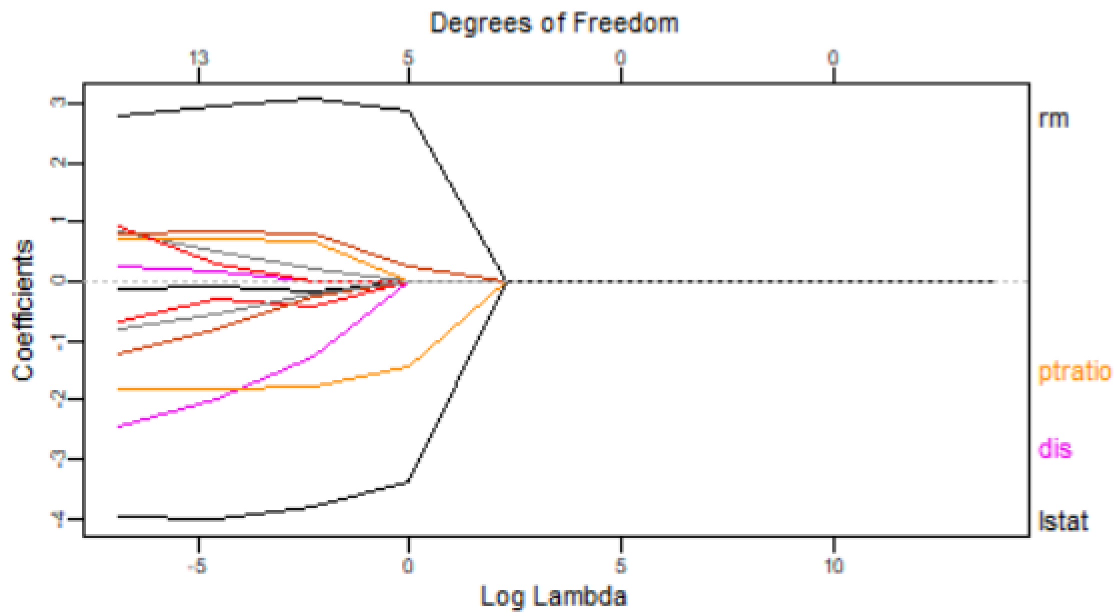
Mathematically,

Medv = 22.53 + (-0.42) * crim + 0.26 * zn + (-0.379) * indus + 0.702 * chas + (-0.693) * nox + 2.636 * rm + (-0.406) * age + (-0.966) * dis + 0.153 * rad + (-0.512) * tax + (-1.476) * ptratio + 0.718 * black + (-2.307) * lstat

iii.) <u>Lasso Regression</u> ->

lasso_mod <- glmnet(scale(x), y, alpha = 1, lambda = grid, thresh = 1e-2, standardize = TRUE)

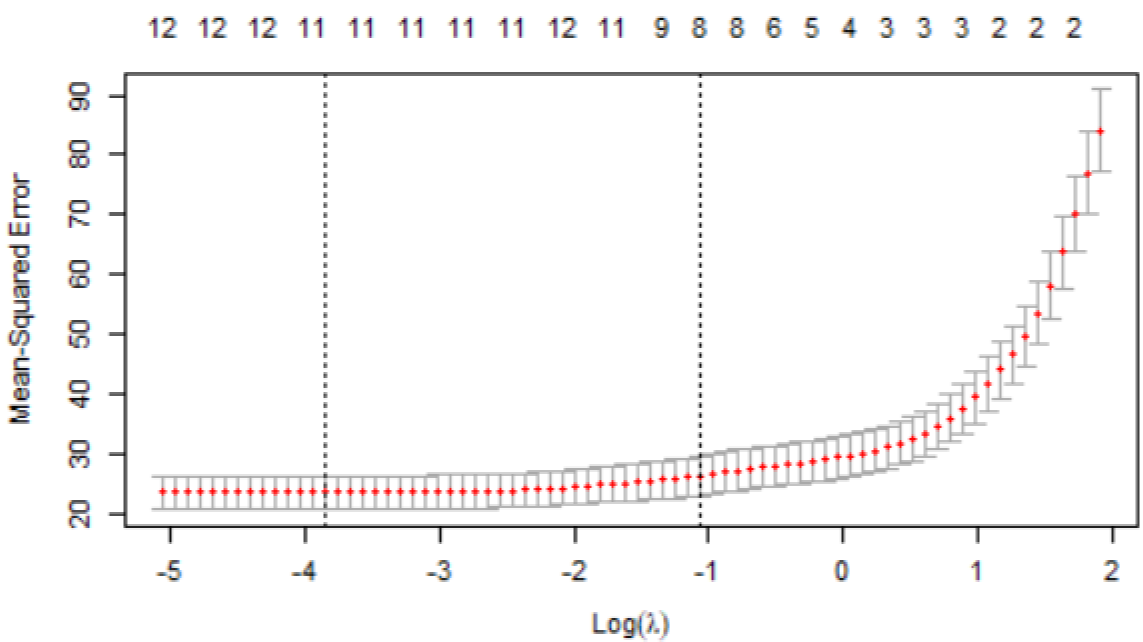
plot_glmnet(lasso_mod, xvar = "lambda", label = 4)



lasso_cv <- cv.glmnet(scale(x), y, alpha = 1, nfolds = 10)

lasso_cv

plot(lasso_cv)



best_lambda_lasso <- lasso_cv$lambda.1se

best_lambda_lasso

lasso_mod_final <- glmnet(scale(x), y, alpha = 0, lambda = best_lambda_lasso,thresh = 1e-2, standardize = TRUE)

predict(lasso_mod, type = "coefficients", s = best_lambda_lasso)

lasso_pred <- predict(lasso_mod_final,s = best_lambda_lasso, newx = scale(x))

sprintf("MSE: %f", mean((lasso_pred - y)^2))

sprintf("RMSE: %f", sqrt(mean((lasso_pred - y)^2)))

```
> lasso_mod <- glmnet(scale(x), y, alpha = 1, lambda = grid, thresh = 1e-2, standardize = TRUE)
> plot_glmnet(lasso_mod, xvar = "lambda", label = 4)
> lasso_cv <- cv.glmnet(scale(x), y, alpha = 1, nfolds = 10)
> lasso_cv

Call:  cv.glmnet(x = scale(x), y = y, nfolds = 10, alpha = 1)

Measure: Mean-Squared Error

    Lambda Index Measure    SE Nonzero
min 0.0212    63   23.59 2.736      11
1se 0.3453    33   26.28 3.257       8
> plot(lasso_cv)
> best_lambda_lasso <- lasso_cv$lambda.1se
> best_lambda_lasso
[1] 0.345263
> lasso_mod_final <- glmnet(scale(x), y, alpha = 0, lambda = best_lambda_lasso,thresh = 1e-2, standardize = TR
UE)
> predict(lasso_mod, type = "coefficients", s = best_lambda_lasso)
14 x 1 sparse Matrix of class "dgCMatrix"
                     s1
(Intercept) 22.532806324
crim        -0.295679571
zn           0.150979488
indus       -0.122420215
chas         0.489321987
nox         -0.190135011
rm           3.019694373
age          0.003252189
dis         -0.917878647
rad          .
tax         -0.167509467
ptratio     -1.680750872
black        0.642568075
lstat       -3.678623923
> lasso_pred <- predict(lasso_mod_final,s = best_lambda_lasso, newx = scale(x))
> sprintf("MSE: %f", mean((lasso_pred - y)^2))
[1] "MSE: 22.769636"
> sprintf("RMSE: %f", sqrt(mean((lasso_pred - y)^2)))
[1] "RMSE: 4.771754"
```

Mathematically,

Medv = 22.53 + (-0.295) * crim + 0.15 * zn + (-0.122) * indus + 0.489 * chas + (-0.190) * nox + 3.01 * (rm) + 0.003 * age + (-0.917) * dis + (-0.167) * tax + (-1.68) * ptratio + 0.642 * black + (-3.678) * lstat

4.) <u>Summary Tables</u> ->

a.) <u>Ridge & Lasso Regression</u> ->

	Ridge Regression (α = 0)	Lasso Regression (α = 1)
min λ	0.678	0.0212
1se λ (optimal lambda, **λ***))	4.356725	0.345263
MSE (**at λ***)	26.301543	22.769636
RMSE (**at λ***)	5.128503	4.771754

b.) **Multiple Linear Regression** ->

Statistic	Multiple Regression	
	Train	Test
MSE	12.02844	16.946752
RMSE	3.468205	4.116643
RSS	4258.069	-
RSE	3.533696	-
R^2	0.8545	-
Adj. R^2	0.8494	-
F-Statistic	166.9	-

5.) **CV in Lasso Regression** -> The Cross-Validation (CV) approach applied here is called "Leave - One-Out- Cross-Validation" where a single observation is used to validate the training set containing the remaining variables. This approach gives an unbiased approximation of the test error but suffers from high variability owing to it being trained on a single variable.

While the model is trained (n-1) times it must be fit 'n' times. This can be very time-consuming if 'n' is large regardless of the n-folds value. The n-folds value is analogous to running a sample of size 'n' from the original sample 'n-folds' times. For large 'n' this can be very tedious and time-consuming as mentioned above.

6.) **References** ->

a.) Introduction to Statis tical Learning
b.) R-bloggers -> (i), (ii), and (iii)
c.) Cross-Validated