Exercise 2

Advanced Methods for Regression and Classification

Rita Selimi

10/28/2024

Load the necessary libraries

Load Data

```
# Load the dataset
load("building.RData")

# Check the data
# head(df)
# Check the structure of the data
str(df)
```

```
## 'data.frame':    372 obs. of  108 variables:
##  $ y                 : num  7.7 8.52 7.09 5.11 8.61 ...
##  $ START.YEAR        : num  81 84 78 72 87 87 87 88 76 80 ...
##  $ START.QUARTER     : num  1 1 1 2 1 1 2 1 3 1 ...
##  $ COMPLETION.YEAR   : num  85 89 81 73 90 90 90 89 77 80 ...
##  $ COMPLETION.QUARTER: num  1 4 4 2 2 1 1 3 4 4 ...
##  $ PhysFin1          : num  1 1 1 1 1 1 1 1 1 1 ...
##  $ PhysFin2          : num  3150 7600 4800 685 3000 2500 1810 1150 2110 3030 ...
##  $ PhysFin3          : num  920 1140 840 202 800 640 492 380 540 930 ...
##  $ PhysFin4          : num  598.5 3040 480 13.7 1230 ...
##  $ PhysFin5          : num  190 400 100 20 410 420 640 500 90 170 ...
##  $ PhysFin6          : num  1011 964 690 460 632 ...
##  $ PhysFin7          : num  16 23 15 4 13 12 11 6 5 3 ...
##  $ PhysFin8          : num  1200 2900 630 140 5000 4800 5700 5300 690 1500 ...
##  $ Econ1             : num  6713 3152 1627 2581 6790 ...
##  $ Econ2             : num  56.2 106 41 12.1 203.8 ...
##  $ Econ3             : num  61.5 103 41.2 10 162.8 ...
##  $ Econ4             : num  6.11 3.15 1.74 1.24 6.46 6.46 6.73 3.44 2.28 5.97 ...
##  $ Econ5             : num  320957 685698 160402 38194 1640293 ...
##  $ Econ6             : num  3486 3526 1218 287 10855 ...
##  $ Econ7             : num  64.5 105.5 34.4 13.6 229.3 ...
##  $ Econ8             : num  240 209 286 17 393 ...
##  $ Econ9             : num  12457 17584 6489 154 69445 ...
##  $ Econ10            : int  15 15 15 12 11 11 11 11 15 15 ...
##  $ Econ11            : num  797 1408 614 184 2739 ...
##  $ Econ12            : num  810 1474 608 211 3148 ...
```

```
##  $ Econ13           : num  1755 8842 1755 1613 9248 ...
##  $ Econ14           : num  8003 8864 7773 1649 9380 ...
##  $ Econ15           : num  67.8 105.5 45.9 11.6 158.6 ...
##  $ Econ16           : num  63.2 105.3 38.3 10.1 169.5 ...
##  $ Econ17           : num  3759 12113 1538 393 10082 ...
##  $ Econ18           : num  42587 45966 39066 8436 49572 ...
##  $ Econ19           : num  628133 1188996 524765 141543 2318397 ...
##  $ Econ1.lag1       : num  4986 2700 1580 2952 6370 ...
##  $ Econ2.lag1       : num  55.5 103 40.3 11.6 190.3 ...
##  $ Econ3.lag1       : num  60.8 101.8 40.8 8.5 154.4 ...
##  $ Econ4.lag1       : num  3.94 2.65 1.15 1.99 5.33 5.33 6.46 3.8 2.32 4.3 ...
##  $ Econ5.lag1       : num  297210 625829 150267 35859 1523167 ...
##  $ Econ6.lag1       : num  3664 4387 1150 322 12930 ...
##  $ Econ7.lag1       : num  61.5 100.4 34.1 12.7 210.7 ...
##  $ Econ8.lag1       : num  179.6 156.6 214.3 56.6 295 ...
##  $ Econ9.lag1       : num  9342 13188 4867 610 52084 ...
##  $ Econ10.lag1      : int  15 15 15 12 11 11 11 11 15 15 ...
##  $ Econ11.lag1      : num  758 1424 574 165 2595 ...
##  $ Econ12.lag1      : num  862 1584 680 209 3000 ...
##  $ Econ13.lag1      : num  1755 8777 1755 1504 9330 ...
##  $ Econ14.lag1      : num  8018 8799 6714 1582 9396 ...
##  $ Econ15.lag1      : num  65 101 43.4 10.9 148.8 ...
##  $ Econ16.lag1      : num  60.53 101.89 36.45 9.79 159 ...
##  $ Econ17.lag1      : num  3539 13572 1535 435 9700 ...
##  $ Econ18.lag1      : num  31940 34475 29300 32776 37179 ...
##  $ Econ19.lag1      : num  610503 1067772 466212 129102 1908976 ...
##  $ Econ1.lag2       : num  6788 3561 2628 2649 5909 ...
##  $ Econ2.lag2       : num  54.2 98.2 39.3 11.4 177.6 ...
##  $ Econ3.lag2       : num  59.4 98.64 40.21 6.97 147.44 ...
##  $ Econ4.lag2       : num  5.41 2.76 1.52 2.25 6.88 6.88 5.33 6.54 2.69 3.53 ...
##  $ Econ5.lag2       : num  280452 602225 143738 32794 1451176 ...
##  $ Econ6.lag2       : num  3756 3819 1284 389 8146 ...
##  $ Econ7.lag2       : num  58.1 97.2 33.5 11.7 188.9 ...
##  $ Econ8.lag2       : num  119.8 104.4 142.9 42.5 196.7 ...
##  $ Econ9.lag2       : num  6228 8792 3245 458 34722 ...
##  $ Econ10.lag2      : int  15 15 15 12 11 11 11 11 15 15 ...
##  $ Econ11.lag2      : num  795 1299 554 168 2284 ...
##  $ Econ12.lag2      : num  818 1390 664 210 2628 ...
##  $ Econ13.lag2      : num  1755 8700 1755 1450 9297 ...
##  $ Econ14.lag2      : num  8001 8735 5827 1507 9347 ...
##  $ Econ15.lag2      : num  63.7 98.1 41.8 10.2 140.9 ...
##  $ Econ16.lag2      : num  58.55 98.45 34.76 9.35 146.2 ...
##  $ Econ17.lag2      : num  3348 13596 1528 509 10149 ...
##  $ Econ18.lag2      : num  21294 22983 19533 24582 24786 ...
##  $ Econ19.lag2      : num  589390 973524 409678 123618 1681849 ...
##  $ Econ1.lag3       : num  5728 3157 2374 2312 7045 ...
##  $ Econ2.lag3       : num  52.4 92.8 38 10.6 160 ...
##  $ Econ3.lag3       : num  57.65 96.49 39.43 5.44 141.34 ...
##  $ Econ4.lag3       : num  5.4 3.05 0.92 2.58 4.72 4.72 6.88 6.73 2.68 3.39 ...
##  $ Econ5.lag3       : num  262789 552124 134548 30012 1341073 ...
##  $ Econ6.lag3       : num  2931 3897 1191 345 8245 ...
##  $ Econ7.lag3       : num  54.2 96.9 33.7 10.8 173.8 ...
##  $ Econ8.lag3       : num  59.9 52.2 71.5 28.3 98.3 ...
##  $ Econ9.lag3       : num  3114 4396 1622 305 17361 ...
```

```
##  $ Econ10.lag3       : int  15 15 15 12 11 11 11 11 15 15 ...
##  $ Econ11.lag3       : num  747 1294 575 180 2451 ...
##  $ Econ12.lag3       : num  816 1288 680 158 2526 ...
##  $ Econ13.lag3       : num  1755 8556 1755 1439 9254 ...
##  $ Econ14.lag3       : num  8013 8585 5565 1450 9306 ...
##  $ Econ15.lag3       : num  62.78 95.35 41.03 9.91 136.56 ...
##  $ Econ16.lag3       : num  56.45 94.34 33.37 8.85 138.8 ...
##  $ Econ17.lag3       : num  3388 12064 1602 591 9291 ...
##  $ Econ18.lag3       : num  10647 11492 9766 16388 12393 ...
##  $ Econ19.lag3       : num  606524 954629 403875 121857 1732938 ...
##  $ Econ1.lag4        : num  7196 3678 2693 1381 5606 ...
##  $ Econ2.lag4        : num  51.3 86.2 36.2 10 149.1 ...
##  $ Econ3.lag4        : num  56.13 83.21 37.64 3.91 134.8 ...
##  $ Econ4.lag4        : num  5.97 3.25 1.55 3 4.09 4.09 4.72 6.46 3.56 3.25 ...
##  $ Econ5.lag4        : num  249111 526596 134313 27231 1284199 ...
##  $ Econ6.lag4        : num  2562 2791 1529 316 6622 ...
##  $ Econ7.lag4        : num  52.8 94.1 31.43 9.85 147.6 ...
##  $ Econ8.lag4        : num  217 334.8 175.7 14.2 432.4 ...
##  $ Econ9.lag4        : num  10446 14489 3995 153 73144 ...
##  $ Econ10.lag4       : int  15 15 15 12 14 14 11 11 15 15 ...
##    [list output truncated]
```

Data Splitting

```r
set.seed(12332281)

# Split the data
sample_index <- sample(1:nrow(df), size = 2/3 * nrow(df))
train_data <- df[sample_index, ]
test_data <- df[-sample_index, ]

# Display the number of samples in each set
cat("Training set size:", nrow(train_data), "\n")
```

```
## Training set size: 248
```

```r
cat("Test set size:", nrow(test_data), "\n")
```

```
## Test set size: 124
```

1. Fit regression model on the training data

```r
# Fit linear regression model
model <- lm(y ~ ., data = train_data)
summary(model)
```

```
##
## Call:
```

```
## lm(formula = y ~ ., data = train_data)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.62167 -0.10676  0.01542  0.13256  0.79241
##
## Coefficients: (35 not defined because of singularities)
##                       Estimate Std. Error t value Pr(>|t|)
## (Intercept)          5.462e+01  2.013e+02   0.271   0.7864
## START.YEAR          -6.769e-01  3.259e+00  -0.208   0.8357
## START.QUARTER       -1.424e+00  2.867e+00  -0.497   0.6201
## COMPLETION.YEAR      9.123e-02  3.554e-02   2.567   0.0111 *
## COMPLETION.QUARTER   4.704e-02  1.859e-02   2.531   0.0123 *
## PhysFin1            -2.910e-02  4.714e-03  -6.173 4.53e-09 ***
## PhysFin2             1.068e-04  4.881e-05   2.187   0.0301 *
## PhysFin3            -3.177e-04  1.390e-04  -2.285   0.0235 *
## PhysFin4            -9.701e-05  6.768e-05  -1.433   0.1535
## PhysFin5            -3.698e-03  6.892e-04  -5.365 2.54e-07 ***
## PhysFin6             7.609e-04  1.323e-04   5.752 3.86e-08 ***
## PhysFin7                    NA         NA      NA       NA
## PhysFin8             5.139e-04  3.545e-05  14.496  < 2e-16 ***
## Econ1                8.098e-06  2.736e-04   0.030   0.9764
## Econ2               -2.633e-01  7.276e-01  -0.362   0.7178
## Econ3                4.186e-01  1.906e-01   2.196   0.0294 *
## Econ4                2.023e-01  2.335e-01   0.867   0.3873
## Econ5               -3.151e-05  5.995e-05  -0.526   0.5999
## Econ6               -5.921e-04  5.914e-04  -1.001   0.3181
## Econ7               -1.800e-01  8.049e-02  -2.237   0.0266 *
## Econ8                1.543e-02  9.111e-03   1.694   0.0921 .
## Econ9               -2.704e-04  1.256e-04  -2.153   0.0327 *
## Econ10               3.662e-01  4.471e-01   0.819   0.4139
## Econ11              -2.477e-03  1.112e-03  -2.226   0.0273 *
## Econ12               1.944e-03  3.712e-03   0.524   0.6011
## Econ13              -3.598e-04  2.656e-04  -1.355   0.1772
## Econ14               1.539e-04  9.799e-04   0.157   0.8754
## Econ15              -1.307e-01  1.246e-01  -1.049   0.2956
## Econ16               1.104e+00  6.323e-01   1.746   0.0826 .
## Econ17               1.996e-04  2.960e-04   0.674   0.5010
## Econ18              -8.322e-05  4.037e-04  -0.206   0.8369
## Econ19              -1.530e-05  8.141e-06  -1.880   0.0618 .
## Econ1.lag1          -8.527e-04  5.212e-04  -1.636   0.1036
## Econ2.lag1          -6.409e-01  6.169e-01  -1.039   0.3003
## Econ3.lag1          -1.559e-01  9.343e-02  -1.668   0.0970 .
## Econ4.lag1           1.078e+00  5.670e-01   1.901   0.0589 .
## Econ5.lag1           1.898e-05  2.076e-05   0.914   0.3619
## Econ6.lag1           6.046e-04  3.616e-04   1.672   0.0963 .
## Econ7.lag1          -5.435e-02  5.815e-02  -0.935   0.3512
## Econ8.lag1          -1.347e-02  6.783e-03  -1.986   0.0486 *
## Econ9.lag1           1.410e-04  1.682e-04   0.838   0.4031
## Econ10.lag1         -5.527e-01  3.228e-01  -1.712   0.0886 .
## Econ11.lag1         -2.559e-03  2.798e-03  -0.914   0.3617
## Econ12.lag1         -5.455e-04  2.616e-03  -0.209   0.8351
## Econ13.lag1          3.317e-04  3.851e-04   0.861   0.3903
## Econ14.lag1          1.026e-03  1.467e-03   0.699   0.4852
```

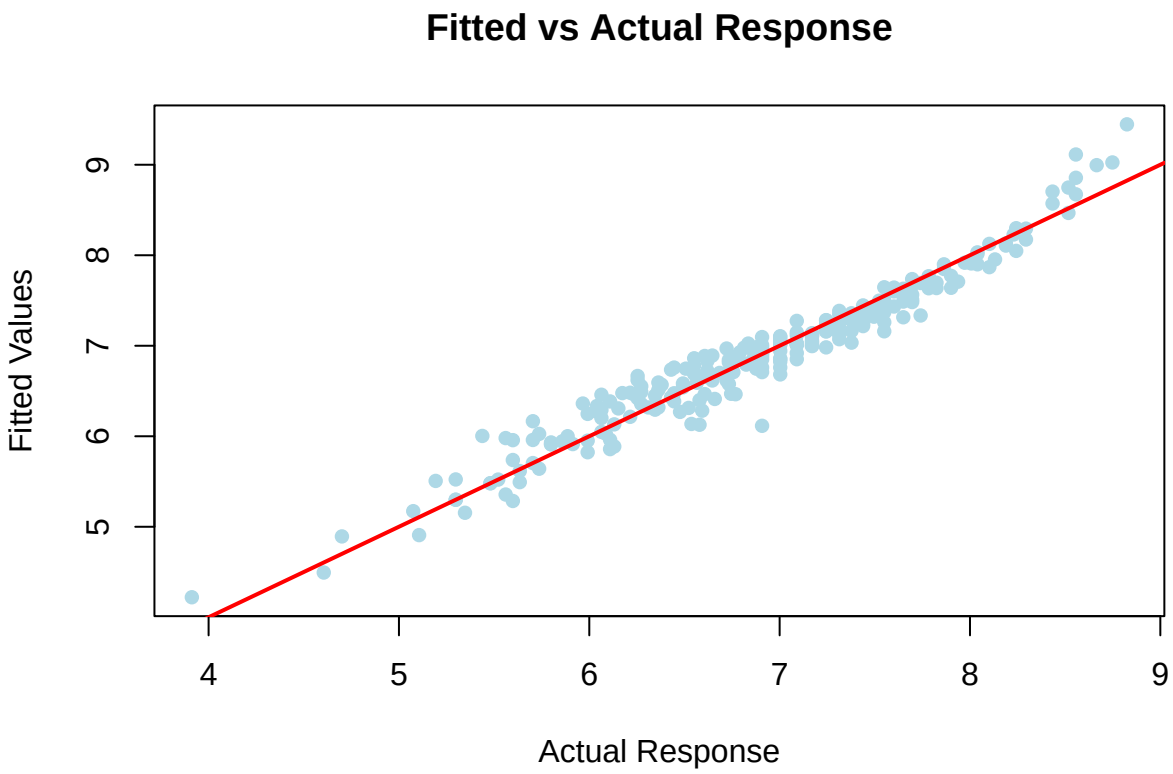
```
## Econ15.lag1         5.226e-01  4.303e-01   1.214   0.2262
## Econ16.lag1        -8.174e-01  4.237e-01  -1.929   0.0553 .
## Econ17.lag1        -5.078e-04  3.945e-04  -1.287   0.1997
## Econ18.lag1        -1.115e-04  3.938e-04  -0.283   0.7773
## Econ19.lag1        -8.797e-06  6.767e-06  -1.300   0.1953
## Econ1.lag2         -5.505e-04  2.759e-04  -1.995   0.0475 *
## Econ2.lag2          1.073e+00  6.536e-01   1.642   0.1024
## Econ3.lag2         -8.127e-02  9.709e-02  -0.837   0.4037
## Econ4.lag2         -3.551e-01  7.570e-01  -0.469   0.6396
## Econ5.lag2          7.792e-05  9.835e-05   0.792   0.4293
## Econ6.lag2          5.508e-05  4.930e-04   0.112   0.9112
## Econ7.lag2          8.836e-03  7.608e-02   0.116   0.9077
## Econ8.lag2         -1.685e-02  7.866e-03  -2.142   0.0336 *
## Econ9.lag2          4.566e-05  2.446e-04   0.187   0.8522
## Econ10.lag2         5.226e-01  3.850e-01   1.357   0.1764
## Econ11.lag2        -2.943e-03  1.547e-03  -1.902   0.0588 .
## Econ12.lag2         3.544e-03  6.200e-03   0.572   0.5683
## Econ13.lag2         1.484e-04  1.965e-04   0.755   0.4510
## Econ14.lag2        -4.028e-04  3.317e-04  -1.214   0.2263
## Econ15.lag2        -3.784e-01  3.156e-01  -1.199   0.2322
## Econ16.lag2        -2.168e-01  7.170e-01  -0.302   0.7628
## Econ17.lag2         2.241e-04  4.399e-04   0.509   0.6111
## Econ18.lag2        -4.286e-05  3.691e-04  -0.116   0.9077
## Econ19.lag2         6.054e-06  4.401e-06   1.375   0.1708
## Econ1.lag3          7.235e-04  3.441e-04   2.102   0.0370 *
## Econ2.lag3         -4.822e-01  2.840e-01  -1.698   0.0913 .
## Econ3.lag3          1.581e-01  1.368e-01   1.155   0.2495
## Econ4.lag3         -2.271e-01  2.762e-01  -0.822   0.4120
## Econ5.lag3                 NA         NA      NA       NA
## Econ6.lag3                 NA         NA      NA       NA
## Econ7.lag3                 NA         NA      NA       NA
## Econ8.lag3                 NA         NA      NA       NA
## Econ9.lag3                 NA         NA      NA       NA
## Econ10.lag3                NA         NA      NA       NA
## Econ11.lag3                NA         NA      NA       NA
## Econ12.lag3                NA         NA      NA       NA
## Econ13.lag3                NA         NA      NA       NA
## Econ14.lag3                NA         NA      NA       NA
## Econ15.lag3                NA         NA      NA       NA
## Econ16.lag3                NA         NA      NA       NA
## Econ17.lag3                NA         NA      NA       NA
## Econ18.lag3                NA         NA      NA       NA
## Econ19.lag3                NA         NA      NA       NA
## Econ1.lag4                 NA         NA      NA       NA
## Econ2.lag4                 NA         NA      NA       NA
## Econ3.lag4                 NA         NA      NA       NA
## Econ4.lag4                 NA         NA      NA       NA
## Econ5.lag4                 NA         NA      NA       NA
## Econ6.lag4                 NA         NA      NA       NA
## Econ7.lag4                 NA         NA      NA       NA
## Econ8.lag4                 NA         NA      NA       NA
## Econ9.lag4                 NA         NA      NA       NA
## Econ10.lag4                NA         NA      NA       NA
## Econ11.lag4                NA         NA      NA       NA
```

```
## Econ12.lag4                NA         NA      NA       NA
## Econ13.lag4                NA         NA      NA       NA
## Econ14.lag4                NA         NA      NA       NA
## Econ15.lag4                NA         NA      NA       NA
## Econ16.lag4                NA         NA      NA       NA
## Econ17.lag4                NA         NA      NA       NA
## Econ18.lag4                NA         NA      NA       NA
## Econ19.lag4                NA         NA      NA       NA
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.2296 on 175 degrees of freedom
## Multiple R-squared:  0.9472, Adjusted R-squared:  0.9255
## F-statistic: 43.61 on 72 and 175 DF,  p-value: < 2.2e-16
```

In the model output, some variables show `NA` values for their estimates because they are too similar to other variables. This issue, called multicollinearity, happens when certain variables contain almost the same information as others, making it hard for the model to tell their effects apart. To handle this, R leaves out these extra variables by marking them as `NA`.

a) Comparing Observed vs Predicted Values (training data)

```r
train_pred <- predict(model, newdata = train_data)

# Plot for Training Data
plot(train_data$y, train_pred,
     main = "Fitted vs Actual Response",
     xlab = "Actual Response",
     ylab = "Fitted Values",
     pch = 16, col = "lightblue")
abline(0, 1, col = "red", lwd = 2)
```




```
# Compute RMSE for training data
train_rmse <- sqrt(mean((train_data$y - train_pred)^2))
cat("Training RMSE:", train_rmse, "\n")
```

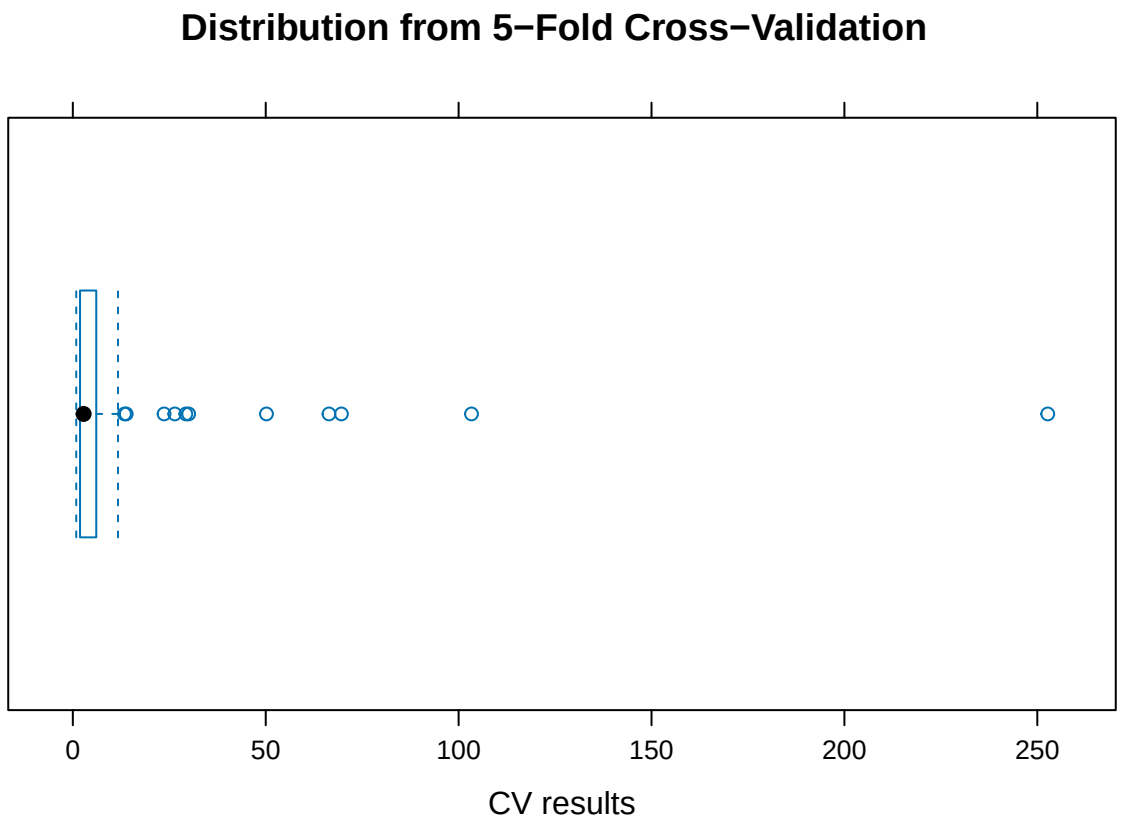
```
## Training RMSE: 0.1928942
```

b) 5-Fold Cross-Validation with 100 Replications (cost=rmspe)

```
# Fit the model using cross-validation
cv_results <- cvFit(
  model,
  data = train_data,
  y = train_data$y,
  cost = rmspe,
  K = 5,
  R = 100
)
print(cv_results)
```

```
## 5-fold CV results:
##       CV
## 9.960448
```

```r
plot(cv_results, main = "Distribution from 5-Fold Cross-Validation")
```



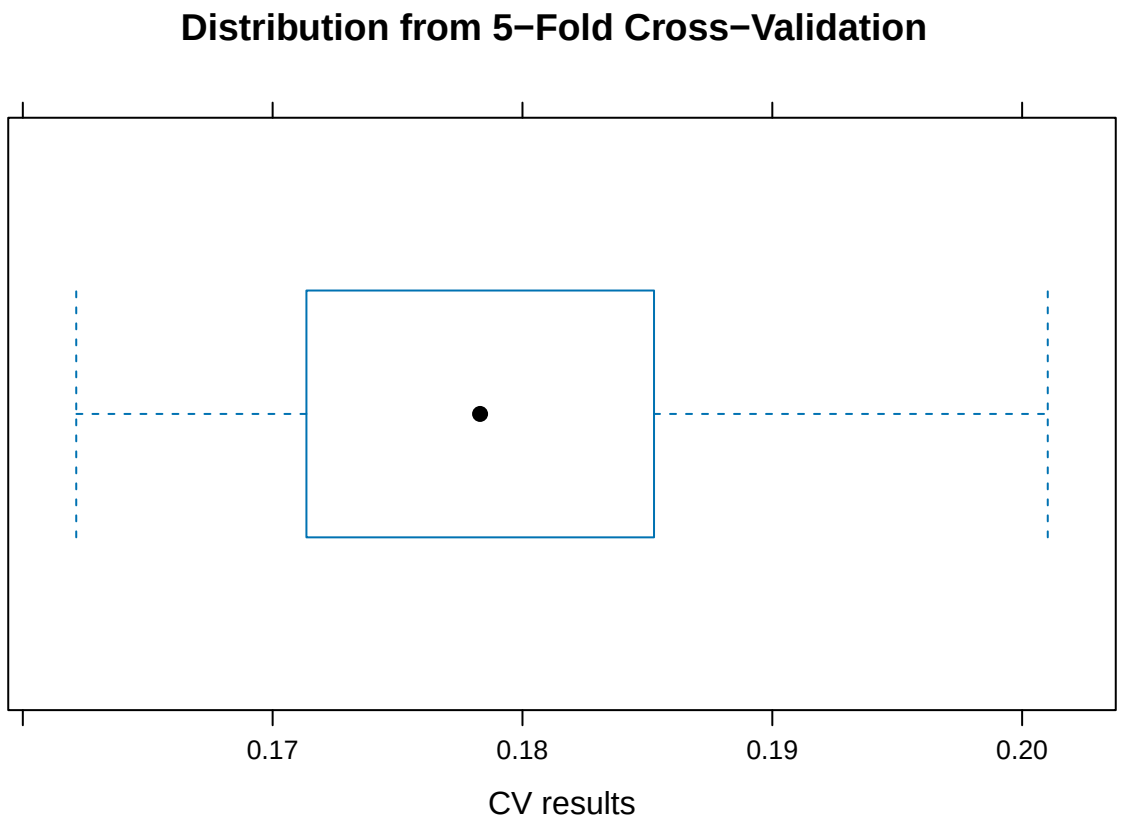
The cross-validation plot shows that most of the RMSE values are close to 10, meaning the model generally predicts well across different parts of the data. However, there are a few outliers with much higher error values, suggesting that the model struggles on certain data subsets.

c) 5-Fold Cross-Validation with 100 Replications (cost=rtmspe)

```r
# Fit the model using cross-validation
cv_results2 <- cvFit(
  model,
  data = train_data,
  y = train_data$y,
  cost = rtmspe,
  K = 5, # 5-fold cross-validation
  R = 100 # 100 replications
)
print(cv_results2)
```

```
## 5-fold CV results:
##        CV
## 0.1790902
```

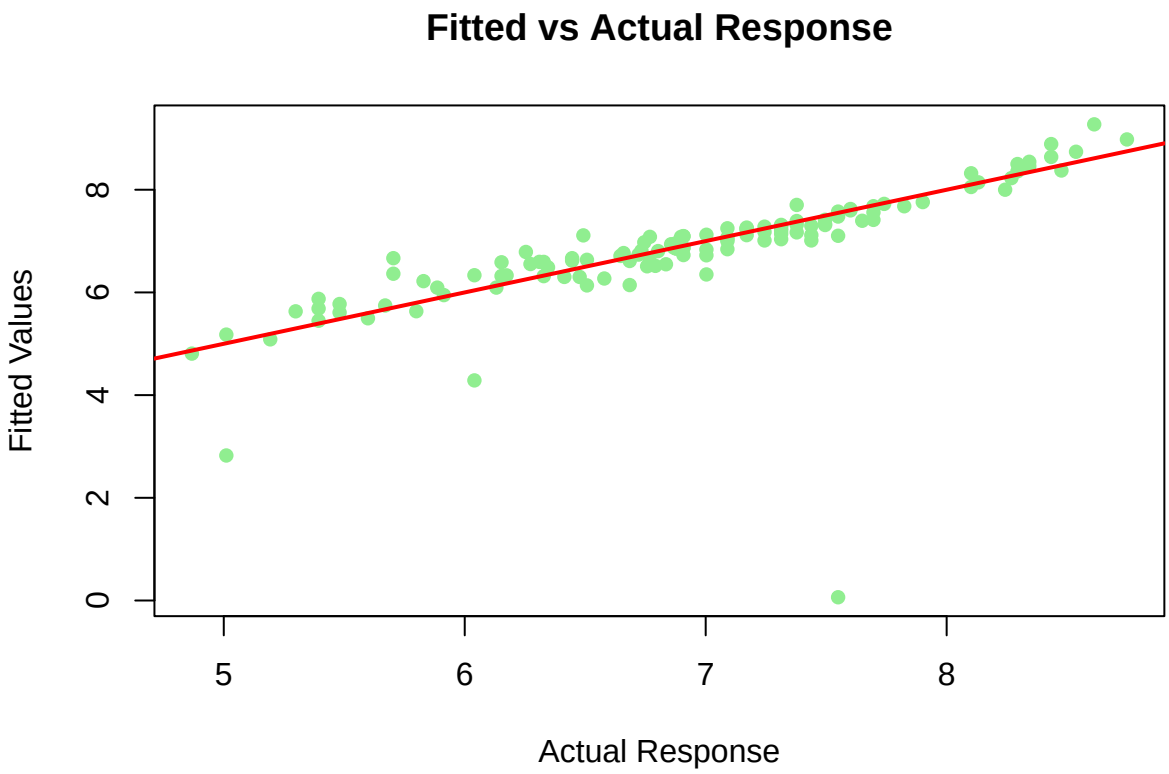
```r
plot(cv_results2, main = "Distribution from 5-Fold Cross-Validation")
```



We are using `rtmspe`, an error metric that reduces the influence of outliers, meaning that the focus in on predicting errors and ignores extreme cases that could skew the results. The plot shows that most of the errors fall within a tight range (between 0.17 and 0.20), indicating that the model performs consistently well across different test sets when outliers are down-weighted.

d) Comparing Observed vs Predicted Values (test data)

```r
test_pred <- predict(model, newdata = test_data)

# Plot for Test Data
plot(test_data$y, test_pred,
     main = "Fitted vs Actual Response",
     xlab = "Actual Response",
     ylab = "Fitted Values",
     pch = 16, col = "lightgreen")
abline(0, 1, col = "red", lwd = 2)
```




```r
# Compute RMSE for testing data
test_rmse <- sqrt(mean((test_data$y - test_pred)^2))
cat("Testing RMSE:", test_rmse, "\n")
```

```
## Testing RMSE: 0.7586301
```

The testing RMSE (0.7586) is higher than the training RMSE (0.1929), suggesting that the model performs well on the training data but may be overfitting, as it is less accurate on new test data. The scatter plot for the test data shows that while most predictions follow the red line (indicating a good fit), there is some spread and a few deviations, which contribute to the higher RMSE.

2. Best Subset Regression

a) Principal Component Regression (PCR)

```r
# Reduce predictors to top 50 based on variance
variances <- apply(train_data[, -ncol(train_data)], 2, var)
top_50_vars <- names(sort(variances, decreasing = TRUE))[1:50]
train_data_reduced <- train_data[, c(top_50_vars, "y")]

# Fit PCR model on the reduced set of 50 predictors
pcr_model <- pcr(y ~ ., data = train_data_reduced, scale = TRUE, validation = "CV")
summary(pcr_model)
```

```
## Data:    X dimension: 248 50
##  Y dimension: 248 1
## Fit method: svdpc
## Number of components considered: 50
##
## VALIDATION: RMSEP
## Cross-validated using 10 random segments.
##        (Intercept)  1 comps  2 comps  3 comps  4 comps  5 comps  6 comps
## CV          0.8429   0.5917   0.5932   0.5908   0.5422   0.5442   0.5442
## adjCV       0.8429   0.5915   0.5930   0.5901   0.5416   0.5436   0.5440
##        7 comps  8 comps  9 comps  10 comps  11 comps  12 comps  13 comps
## CV      0.5105   0.5146   0.5225    0.3437    0.3325    0.3172    0.3193
## adjCV   0.5088   0.5131   0.5218    0.3416    0.3307    0.3153    0.3182
##        14 comps  15 comps  16 comps  17 comps  18 comps  19 comps  20 comps
## CV       0.3187    0.3192    0.3229    0.3233    0.3219    0.3242    0.3250
## adjCV    0.3177    0.3182    0.3218    0.3222    0.3206    0.3229    0.3236
##        21 comps  22 comps  23 comps  24 comps  25 comps  26 comps  27 comps
## CV       0.3278    0.3246    0.3252    0.3255    0.3265    0.3240    0.3216
## adjCV    0.3263    0.3229    0.3236    0.3239    0.3249    0.3219    0.3196
##        28 comps  29 comps  30 comps  31 comps  32 comps  33 comps  34 comps
## CV       0.3247    0.3268    0.3314    0.3326    0.3359    0.3323    0.3333
## adjCV    0.3228    0.3249    0.3292    0.3306    0.3334    0.3294    0.3302
##        35 comps  36 comps  37 comps  38 comps  39 comps  40 comps  41 comps
## CV       0.3332    0.3338    0.3343    0.3360    0.3362    0.3401    0.3412
## adjCV    0.3303    0.3310    0.3313    0.3328    0.3331    0.3369    0.3378
##        42 comps  43 comps  44 comps  45 comps  46 comps  47 comps  48 comps
## CV       0.3450    0.3488    0.3499    0.3469    0.3481    0.3520    0.4033
## adjCV    0.3414    0.3447    0.3457    0.3429    0.3440    0.3477    0.3949
##        49 comps  50 comps
## CV       0.4093    0.3927
## adjCV    0.4004    0.3850
##
## TRAINING: % variance explained
##    1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps  8 comps
## X    62.80    71.77    77.59    83.03    86.65    89.14    91.29    93.26
## y    50.82    51.02    53.05    60.54    60.57    60.81    66.68    67.39
##    9 comps  10 comps  11 comps  12 comps  13 comps  14 comps  15 comps
## X    94.93     96.06     96.85     97.46     97.96     98.36     98.68
## y    67.53     84.98     85.89     86.99     87.00     87.05     87.10
##    16 comps  17 comps  18 comps  19 comps  20 comps  21 comps  22 comps
## X     98.94     99.11     99.27     99.39     99.49     99.58     99.63
## y     87.10     87.15     87.32     87.32     87.34     87.34     87.58
##    23 comps  24 comps  25 comps  26 comps  27 comps  28 comps  29 comps
## X     99.69     99.73     99.78     99.81     99.84     99.87     99.89
## y     87.60     87.63     87.71     88.03     88.14     88.27     88.29
##    30 comps  31 comps  32 comps  33 comps  34 comps  35 comps  36 comps
## X     99.91     99.93     99.94     99.95     99.96     99.97     99.97
## y     88.30     88.35     88.48     88.89     88.91     88.94     88.95
##    37 comps  38 comps  39 comps  40 comps  41 comps  42 comps  43 comps
## X     99.98     99.99     99.99     99.99    100.00    100.00    100.00
## y     89.07     89.11     89.11     89.14     89.25     89.26     89.36
##    44 comps  45 comps  46 comps  47 comps  48 comps  49 comps  50 comps
## X     100.0    100.00    100.00    100.00    100.00    100.00    100.00
## y      89.4     89.53     89.53     89.53     89.53     89.62     89.74
```

```r
# Extract the top 10 components from the PCR model
pcr_scores <- as.data.frame(pcr_model$scores[, 1:10])
pcr_scores$y <- train_data$y  # Add the response variable
```

First, I calculated the variance of each predictor in the training data and selected the top 50 predictors with the highest variance. This approach ensures that we retain the predictors most likely to contain significant information. Next, I applied Principal Component Regression (PCR) on this reduced dataset of 50 predictors, which creates new components that capture the main patterns in the data. Finally, we extracted the top 10 components from the PCR model, allowing us to limit the model size to 10 regressors. These components, along with the response variable, form a new dataset that can be used for further modeling.

Best Subset Regression

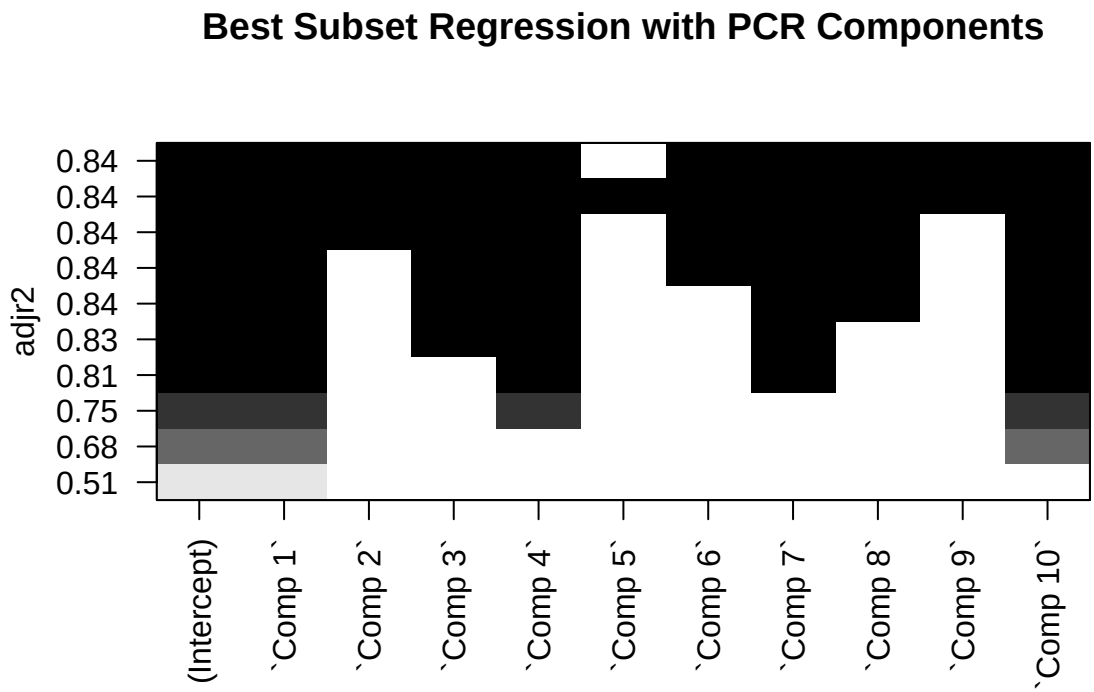
```r
# Run best subset regression with the modified data
best_subset <- regsubsets(y ~ ., data = pcr_scores, nvmax = 10, really.big = TRUE)
summary(best_subset)
```

```
## Subset selection object
## Call: regsubsets.formula(y ~ ., data = pcr_scores, nvmax = 10, really.big = TRUE)
## 10 Variables  (and intercept)
##           Forced in Forced out
## `Comp 1`      FALSE      FALSE
## `Comp 2`      FALSE      FALSE
## `Comp 3`      FALSE      FALSE
## `Comp 4`      FALSE      FALSE
## `Comp 5`      FALSE      FALSE
## `Comp 6`      FALSE      FALSE
## `Comp 7`      FALSE      FALSE
## `Comp 8`      FALSE      FALSE
## `Comp 9`      FALSE      FALSE
## `Comp 10`     FALSE      FALSE
## 1 subsets of each size up to 10
## Selection Algorithm: exhaustive
##           `Comp 1` `Comp 2` `Comp 3` `Comp 4` `Comp 5` `Comp 6` `Comp 7`
## 1  ( 1 )  "*"      " "      " "      " "      " "      " "      " "
## 2  ( 1 )  "*"      " "      " "      " "      " "      " "      " "
## 3  ( 1 )  "*"      " "      " "      "*"      " "      " "      " "
## 4  ( 1 )  "*"      " "      " "      "*"      " "      " "      "*"
## 5  ( 1 )  "*"      " "      "*"      "*"      " "      " "      "*"
## 6  ( 1 )  "*"      " "      "*"      "*"      " "      " "      "*"
## 7  ( 1 )  "*"      " "      "*"      "*"      " "      "*"      "*"
## 8  ( 1 )  "*"      "*"      "*"      "*"      " "      "*"      "*"
## 9  ( 1 )  "*"      "*"      "*"      "*"      " "      "*"      "*"
## 10  ( 1 ) "*"      "*"      "*"      "*"      "*"      "*"      "*"
##           `Comp 8` `Comp 9` `Comp 10`
## 1  ( 1 )  " "      " "      " "
## 2  ( 1 )  " "      " "      "*"
## 3  ( 1 )  " "      " "      "*"
## 4  ( 1 )  " "      " "      "*"
## 5  ( 1 )  " "      " "      "*"
## 6  ( 1 )  "*"      " "      "*"
```

```
## 7  ( 1 )  "*"      " "      "*"
## 8  ( 1 )  "*"      " "      "*"
## 9  ( 1 )  "*"      "*"      "*"
## 10  ( 1 ) "*"      "*"      "*"
```

b) Plot the Best Subset Results

```r
# Plot the best subset regression results to determine the best model
plot(best_subset, scale = "adjr2", main = "Best Subset Regression with PCR Components")
```



In this best subset regression plot, we are comparing models with different combinations of components to find the one with the highest adjusted R^2, which indicates the model's accuracy while adjusting for the number of components used. The model that seems to be the best includes Comp 1+Comp 3+Comp 4+Comp 7+Comp 8+Comp 10. This model is likely the best choice, as it explains the most variability in the data with a minimal number of components.

c) Apply lm() on the final best model

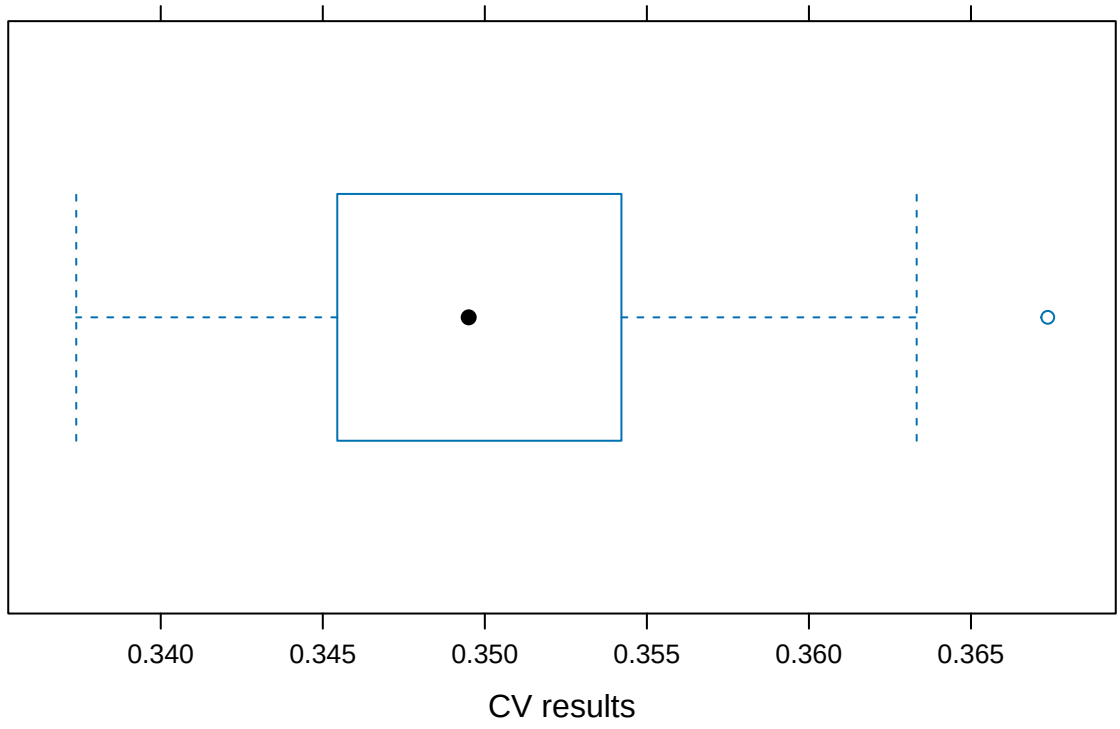
```r
# Fit the linear model using the selected components
final_model <- lm(y ~ `Comp 1` + `Comp 3` + `Comp 4` + `Comp 7` +
                    `Comp 8` + `Comp 10`, data = pcr_scores)
summary(final_model)
```

```
##
## Call:
## lm(formula = y ~ `Comp 1` + `Comp 3` + `Comp 4` + `Comp 7` +
##     `Comp 8` + `Comp 10`, data = pcr_scores)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.36875 -0.16630  0.03118  0.21907  0.80714
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  6.886883   0.021379 322.128  < 2e-16 ***
## `Comp 1`     0.107015   0.003823  27.992  < 2e-16 ***
## `Comp 3`    -0.070281   0.012553  -5.599 5.87e-08 ***
## `Comp 4`     0.139650   0.012994  10.747  < 2e-16 ***
## `Comp 7`     0.196474   0.020663   9.509  < 2e-16 ***
## `Comp 8`     0.071575   0.021596   3.314  0.00106 **
## `Comp 10`   -0.466545   0.028437 -16.406  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.3367 on 241 degrees of freedom
## Multiple R-squared:  0.8437, Adjusted R-squared:  0.8398
## F-statistic: 216.8 on 6 and 241 DF,  p-value: < 2.2e-16
```

```r
# Evaluate the model with 5-fold cross-validation and 100 replications
# (1b) Using rmspe as cost
cv_results_rmspe <- cvFit(
  final_model,
  data = pcr_scores,
  y = pcr_scores$y,
  cost = rmspe,
  K = 5,
  R = 100
)

# Display and plot the cross-validation results for rmspe
print(cv_results_rmspe)
```

```
## 5-fold CV results:
##        CV
## 0.3501755
```

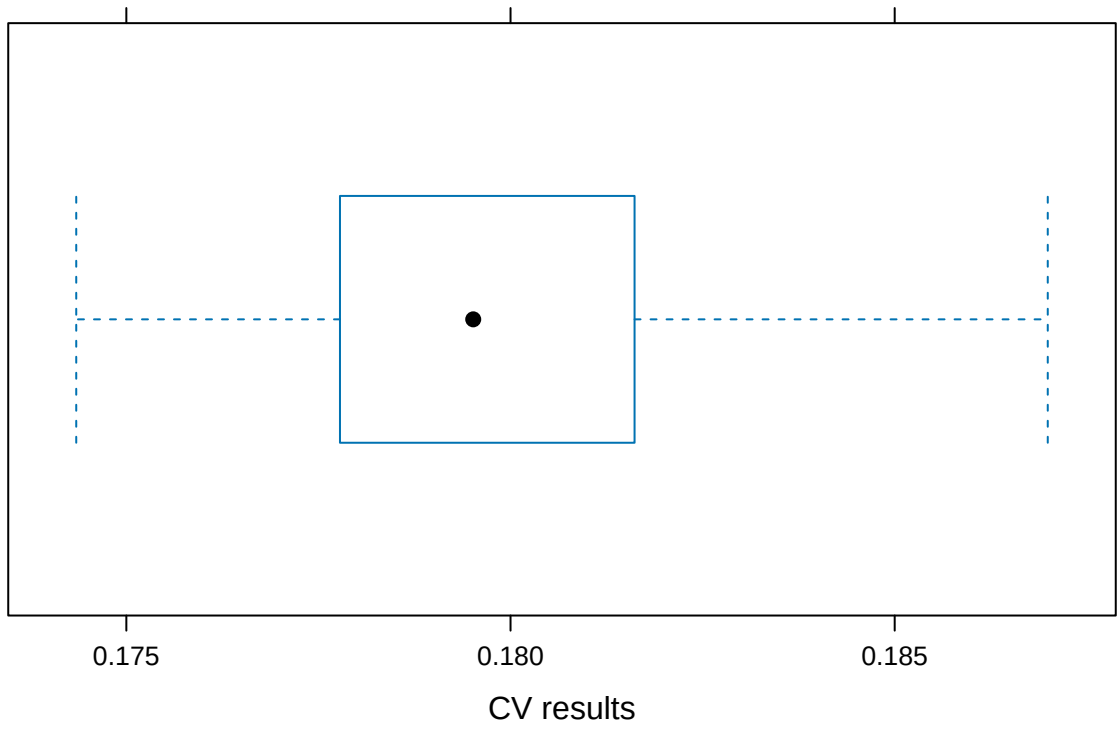
```r
plot(cv_results_rmspe, main = "5-Fold Cross-Validation with RMSPE")
```




```
# (1c) Using rtmspe as cost
cv_results_rtmspe <- cvFit(
  final_model,
  data = pcr_scores,
  y = pcr_scores$y,
  cost = rtmspe,
  K = 5,
  R = 100
)

# Display and plot the cross-validation results for rtmspe
print(cv_results_rtmspe)
```

```
## 5-fold CV results:
##        CV
## 0.1798701
```

```
plot(cv_results_rtmspe, main = "5-Fold Cross-Validation with RTMSPE")
```

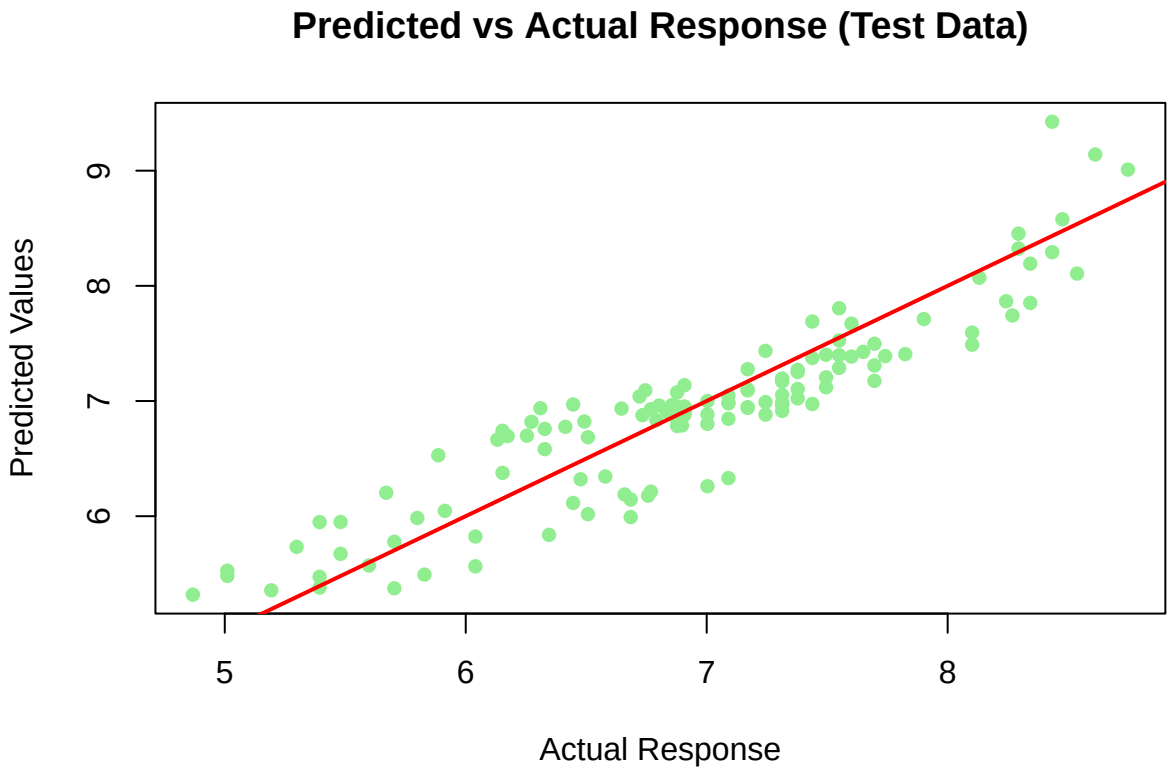


The cross-validation results using two error metrics show that, with RMSPE, the model achieved a cross-validated error of approximately 0.35, and with RTMSPE, the model had a lower error of approximately 0.17. These results indicate that the selected model performs well with a relatively low prediction error.

d) Comparing Observed vs Predicted Values (test data)

```
# Ensure `test_data_reduced` matches `train_data_reduced` in terms of columns and order
test_data_reduced <- test_data[, colnames(train_data_reduced)[-ncol(train_data_reduced)]]

# Manually extract means and sds from `train_data_reduced` if needed
train_means <- apply(train_data_reduced[, -ncol(train_data_reduced)], 2, mean)
train_sds <- apply(train_data_reduced[, -ncol(train_data_reduced)], 2, sd)

# Center and scale `test_data_reduced` using these means and sds
test_data_scaled <- scale(test_data_reduced, center = train_means, scale = train_sds)

# Calculate the components manually by projecting on `pcr_model$loadings`
pcr_test_scores <- as.data.frame(test_data_scaled %*% pcr_model$loadings[, 1:10])

# Set column names to match `pcr_scores`
colnames(pcr_test_scores) <- colnames(pcr_scores)[1:10]

# Ensure pcr_test_scores includes the actual `y` values
pcr_test_scores$y <- test_data$y
```

```r
# Predict response for the test data using the final model
test_pred <- predict(final_model, newdata = pcr_test_scores)

# Plot Predicted vs Actual for Test Data
plot(pcr_test_scores$y, test_pred,
     main = "Predicted vs Actual Response (Test Data)",
     xlab = "Actual Response",
     ylab = "Predicted Values",
     pch = 16, col = "lightgreen")
abline(0, 1, col = "red", lwd = 2)
```




```r
# Calculate RMSE for Test Data
test_rmse <- sqrt(mean((pcr_test_scores$y - test_pred)^2))
cat("Test RMSE:", test_rmse, "\n")
```

```
## Test RMSE: 0.3480036
```

The RMSE for the test data dropped from 0.76 in the original model to 0.34 after using the best subset model, by this we can conclude that the best subset model significantly improves prediction accuracy. This means the best subset model makes more accurate predictions because it focuses on the most important parts of the data.