Assignment 5 sol

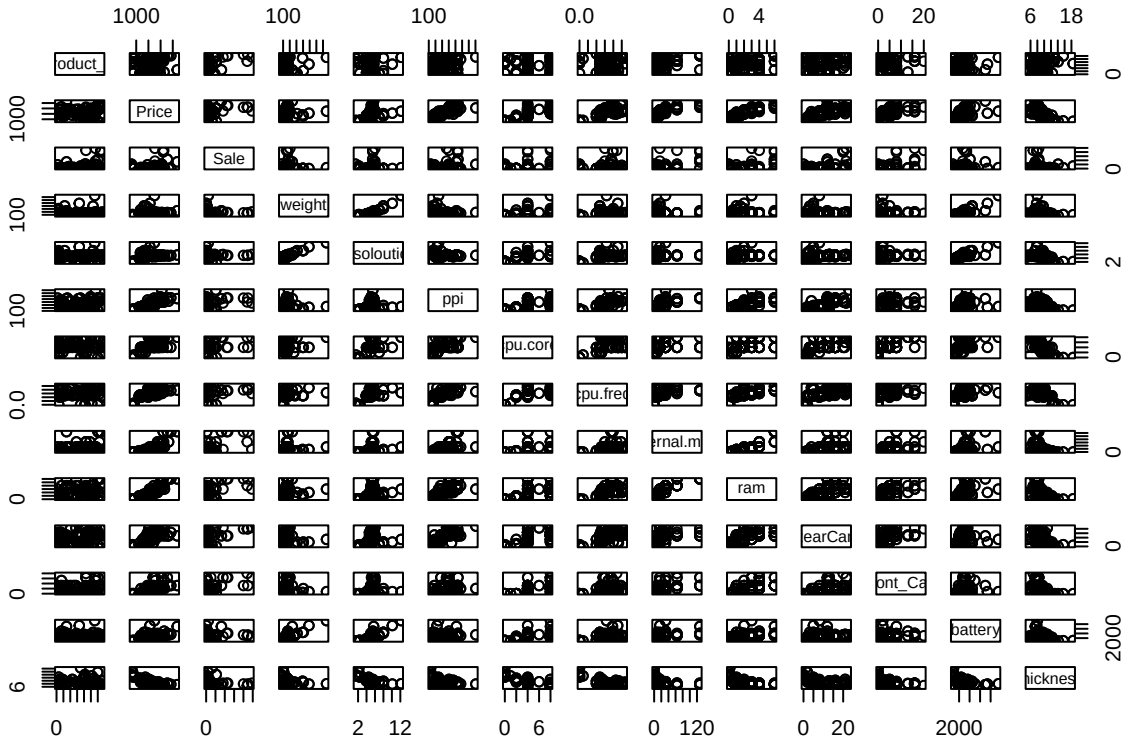
Sunny Kumar

2025-03-03

```
df <- Cellphone
head(df)
```

```
##   Product_id Price Sale weight resoloution ppi cpu.core cpu.freq internal.mem
## 1        203  2357   10  135.0         5.2 424        8     1.35           16
## 2        880  1749   10  125.0         4.0 233        2     1.30            4
## 3         40  1916   10  110.0         4.7 312        4     1.20            8
## 4         99  1315   11  118.5         4.0 233        2     1.30            4
## 5        880  1749   11  125.0         4.0 233        2     1.30            4
## 6        947  2137   12  150.0         5.5 401        4     2.30           16
##     ram RearCam Front_Cam battery thickness
## 1 3.000   13.00         8    2610       7.4
## 2 1.000    3.15         0    1700       9.9
## 3 1.500   13.00         5    2000       7.6
## 4 0.512    3.15         0    1400      11.0
## 5 1.000    3.15         0    1700       9.9
## 6 2.000   16.00         8    2500       9.5
```

```
plot(df)
```




```
summary(df)
```

```
##    Product_id         Price           Sale           weight
##  Min.   :  10.0   Min.   : 614   Min.   :  10.0   Min.   : 66.0
##  1st Qu.: 237.0   1st Qu.:1734   1st Qu.:  37.0   1st Qu.:134.1
##  Median : 774.0   Median :2258   Median : 106.0   Median :153.0
##  Mean   : 675.6   Mean   :2216   Mean   : 621.5   Mean   :170.4
##  3rd Qu.:1026.0   3rd Qu.:2744   3rd Qu.: 382.0   3rd Qu.:170.0
##  Max.   :1339.0   Max.   :4361   Max.   :9807.0   Max.   :753.0
##   resoloution          ppi          cpu.core        cpu.freq
##  Min.   : 1.40   Min.   :121.0   Min.   :0.000   Min.   :0.000
##  1st Qu.: 4.80   1st Qu.:233.0   1st Qu.:4.000   1st Qu.:1.200
##  Median : 5.15   Median :294.0   Median :4.000   Median :1.400
##  Mean   : 5.21   Mean   :335.1   Mean   :4.857   Mean   :1.503
##  3rd Qu.: 5.50   3rd Qu.:428.0   3rd Qu.:8.000   3rd Qu.:1.875
##  Max.   :12.20   Max.   :806.0   Max.   :8.000   Max.   :2.700
##   internal.mem         ram           RearCam        Front_Cam
##  Min.   :  0.0   Min.   :0.000   Min.   : 0.00   Min.   : 0.000
##  1st Qu.:  8.0   1st Qu.:1.000   1st Qu.: 5.00   1st Qu.: 0.000
##  Median : 16.0   Median :2.000   Median :12.00   Median : 5.000
##  Mean   : 24.5   Mean   :2.205   Mean   :10.38   Mean   : 4.503
##  3rd Qu.: 32.0   3rd Qu.:3.000   3rd Qu.:16.00   3rd Qu.: 8.000
##  Max.   :128.0   Max.   :6.000   Max.   :23.00   Max.   :20.000
##     battery       thickness
##  Min.   : 800   Min.   : 5.100
```

```
##  1st Qu.:2040   1st Qu.: 7.600
##  Median :2800   Median : 8.400
##  Mean   :2842   Mean   : 8.922
##  3rd Qu.:3240   3rd Qu.: 9.800
##  Max.   :9500   Max.   :18.500
```

```
library(ggplot2)
library(corrplot)
```

```
## corrplot 0.95 loaded
```

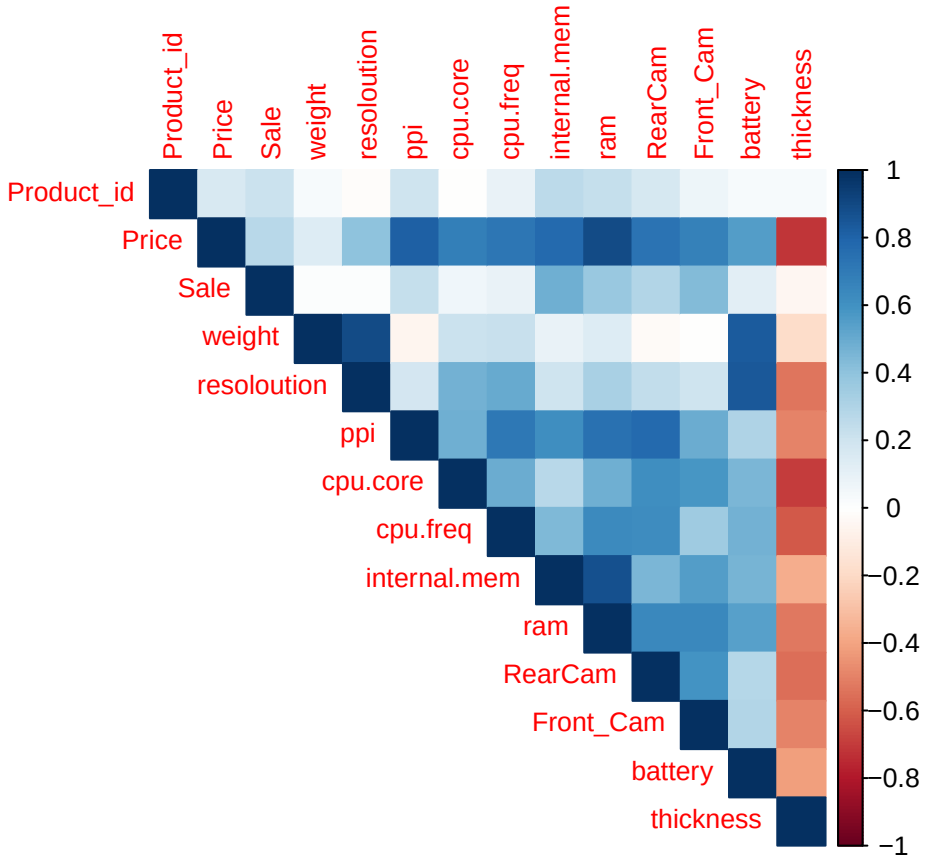
```
cor_matrix <- cor(df[, sapply(df, is.numeric)], use = "complete.obs")# Correlation matrix
cor_matrix
```

```
##                Product_id      Price        Sale       weight resoloution
## Product_id    1.000000000  0.1651358  0.21854822  0.035868997 -0.01781743
## Price         0.165135812  1.0000000  0.27326252  0.144555117  0.40400956
## Sale          0.218548215  0.2732625  1.00000000  0.016114462  0.01715418
## weight        0.035868997  0.1445551  0.01611446  1.000000000  0.89041650
## resoloution  -0.017817433  0.4040096  0.01715418  0.890416497  1.00000000
## ppi           0.207270573  0.8176145  0.23589646 -0.054682633  0.18129159
## cpu.core     -0.008634003  0.6868106  0.06951745  0.216257283  0.47222591
## cpu.freq      0.092427978  0.7273828  0.09913282  0.222729918  0.50545380
## internal.mem  0.261186427  0.7767378  0.48930802  0.098849256  0.20265951
## ram           0.236194427  0.8969151  0.37312667  0.149283264  0.32721632
## RearCam       0.172811675  0.7395376  0.29216432 -0.029448190  0.24989197
## Front_Cam     0.071020207  0.6752864  0.43723322 -0.005970849  0.20272023
## battery       0.031744856  0.5599457  0.12033837  0.833782666  0.84346176
## thickness     0.039807160 -0.7167731 -0.04799095 -0.185262384 -0.53370786
##                      ppi     cpu.core    cpu.freq internal.mem        ram
## Product_id    0.20727057 -0.008634003  0.09242798   0.26118643  0.2361944
## Price         0.81761445  0.686810645  0.72738283   0.77673777  0.8969151
## Sale          0.23589646  0.069517451  0.09913282   0.48930802  0.3731267
## weight       -0.05468263  0.216257283  0.22272992   0.09884926  0.1492833
## resoloution   0.18129159  0.472225906  0.50545380   0.20265951  0.3272163
## ppi           1.00000000  0.487990138  0.71316830   0.61855960  0.7487245
## cpu.core      0.48799014  1.000000000  0.49151900   0.27625082  0.4831275
## cpu.freq      0.71316830  0.491518995  1.00000000   0.44140011  0.6335475
## internal.mem  0.61855960  0.276250819  0.44140011   1.00000000  0.8753536
## ram           0.74872447  0.483127510  0.63354749   0.87535357  1.0000000
## RearCam       0.77400811  0.611352632  0.62510407   0.45191665  0.6480732
## Front_Cam     0.49137066  0.586698382  0.35831903   0.55573337  0.6474698
## battery       0.30251667  0.459728437  0.47313689   0.46150566  0.5410011
## thickness    -0.49679079 -0.697935474 -0.61445827  -0.36741160 -0.5210736
##                  RearCam    Front_Cam     battery   thickness
## Product_id    0.17281168  0.071020207  0.03174486  0.03980716
## Price         0.73953757  0.675286410  0.55994569 -0.71677306
## Sale          0.29216432  0.437233222  0.12033837 -0.04799095
## weight       -0.02944819 -0.005970849  0.83378267 -0.18526238
## resoloution   0.24989197  0.202720227  0.84346176 -0.53370786
## ppi           0.77400811  0.491370660  0.30251667 -0.49679079
## cpu.core      0.61135263  0.586698382  0.45972844 -0.69793547
## cpu.freq      0.62510407  0.358319034  0.47313689 -0.61445827
```

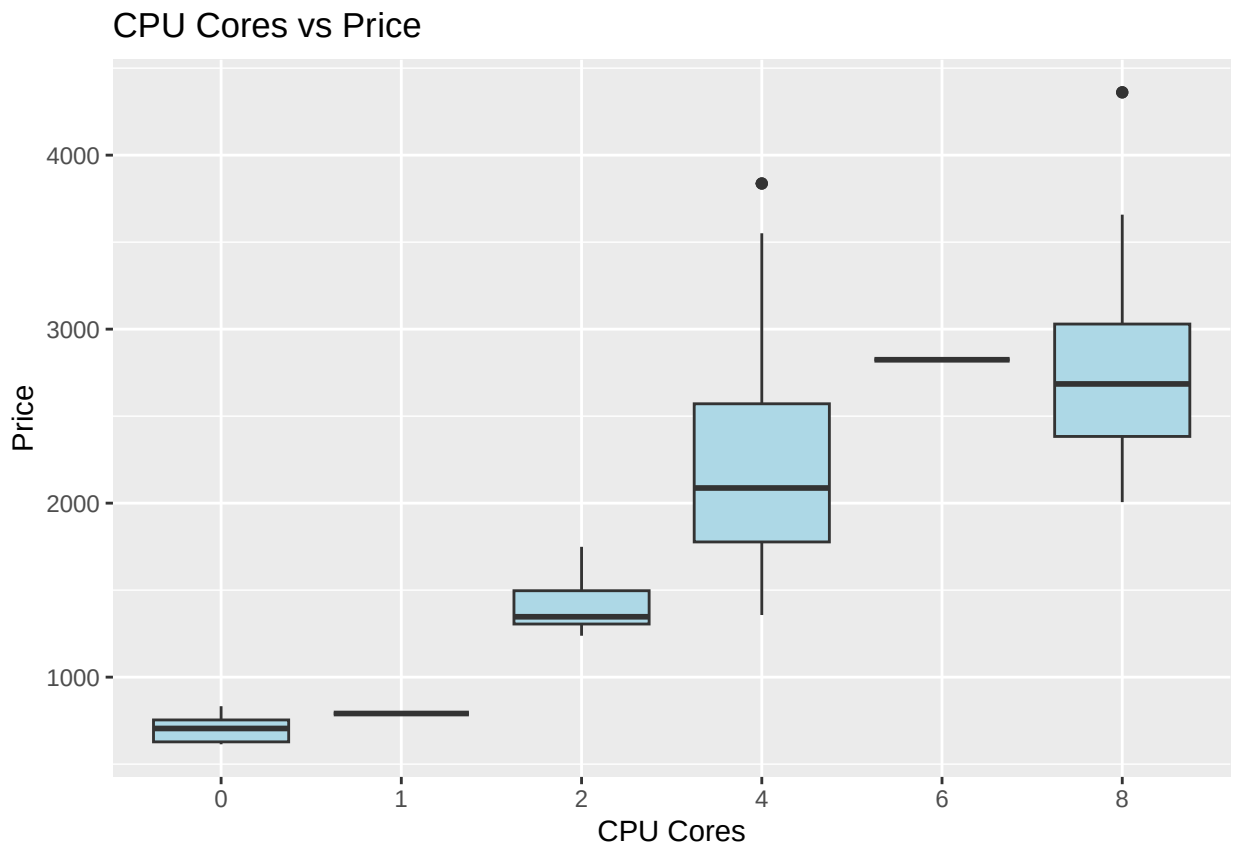
```
## internal.mem  0.45191665  0.555733370  0.46150566 -0.36741160
## ram           0.64807316  0.647469774  0.54100108 -0.52107357
## RearCam       1.00000000  0.596373692  0.28782142 -0.55099793
## Front_Cam     0.59637369  1.000000000  0.29528337 -0.49354169
## battery       0.28782142  0.295283372  1.00000000 -0.41268185
## thickness    -0.55099793 -0.493541691 -0.41268185  1.00000000
```

From the output we can see that,Columns like PPI , CPU core , Cpu.freq, internal.mem, ram, RearCam,Front_Cam, thickness have high correleation with the Price

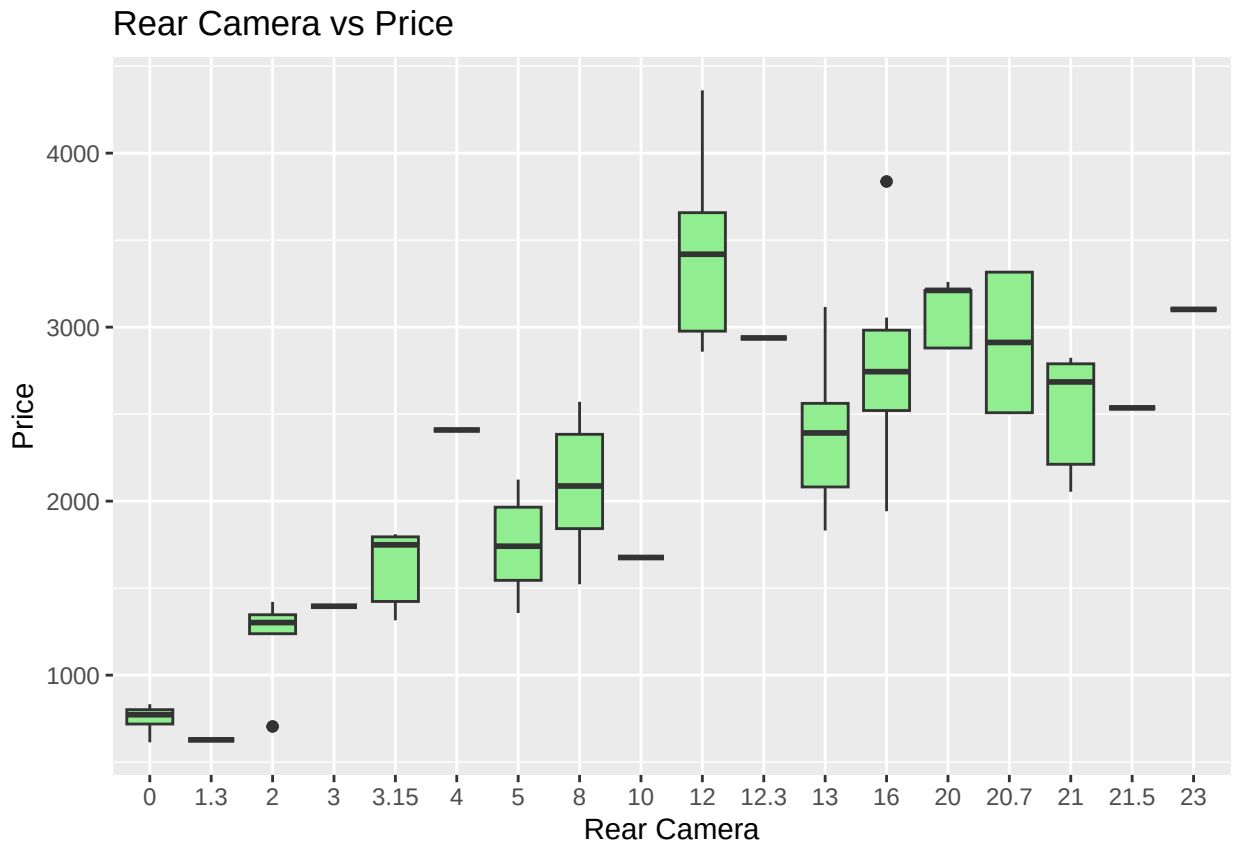
```
corrplot(cor_matrix, method = "color", type = "upper", tl.cex = 0.8)
```




```
ggplot(df, aes(x = as.factor(cpu.core), y = Price)) +
  geom_boxplot(fill = "lightblue") +
  labs(title = "CPU Cores vs Price", x = "CPU Cores", y = "Price")
```




```
ggplot(df, aes(x = as.factor(RearCam), y = Price)) +
  geom_boxplot(fill = "lightgreen") +
  labs(title = "Rear Camera vs Price", x = "Rear Camera", y = "Price")
```




```
# Performing Best Subset selection
library(leaps)
df <- df[, !names(df) %in% "Product_id"]
best_model <- regsubsets(Price ~ ., data = df, nvmax =  12) # Total 13 cols , taking all cols
summary(best_model)
```

```
## Subset selection object
## Call: regsubsets.formula(Price ~ ., data = df, nvmax = 12)
## 12 Variables  (and intercept)
##              Forced in Forced out
## Sale             FALSE      FALSE
## weight           FALSE      FALSE
## resoloution      FALSE      FALSE
## ppi              FALSE      FALSE
## cpu.core         FALSE      FALSE
## cpu.freq         FALSE      FALSE
## internal.mem     FALSE      FALSE
## ram              FALSE      FALSE
## RearCam          FALSE      FALSE
## Front_Cam        FALSE      FALSE
## battery          FALSE      FALSE
## thickness        FALSE      FALSE
## 1 subsets of each size up to 12
## Selection Algorithm: exhaustive
##          Sale weight resoloution ppi cpu.core cpu.freq internal.mem ram
## 1  ( 1 )  " "  " "    " "        " " " "      " "      " "          "*"
```

```
## 2  ( 1 )  " "  " "    " "         " " " "      " "      " "          "*"
## 3  ( 1 )  " "  " "    " "         "*" " "      " "      " "          "*"
## 4  ( 1 )  " "  " "    " "         "*" "*"      " "      " "          "*"
## 5  ( 1 )  " "  " "    " "         "*" "*"      " "      "*"          "*"
## 6  ( 1 )  " "  " "    " "         "*" "*"      "*"      "*"          "*"
## 7  ( 1 )  " "  "*"    " "         "*" "*"      " "      "*"          "*"
## 8  ( 1 )  " "  "*"    " "         "*" "*"      "*"      "*"          "*"
## 9  ( 1 )  " "  " "    "*"         "*" "*"      "*"      "*"          "*"
## 10  ( 1 ) "*"  " "    "*"         "*" "*"      "*"      "*"          "*"
## 11  ( 1 ) "*"  " "    "*"         "*" "*"      "*"      "*"          "*"
## 12  ( 1 ) "*"  "*"    "*"         "*" "*"      "*"      "*"          "*"
##           RearCam Front_Cam battery thickness
## 1  ( 1 )  " "     " "       " "     " "
## 2  ( 1 )  " "     " "       " "     "*"
## 3  ( 1 )  " "     " "       " "     "*"
## 4  ( 1 )  " "     " "       " "     "*"
## 5  ( 1 )  " "     " "       " "     "*"
## 6  ( 1 )  " "     " "       " "     "*"
## 7  ( 1 )  " "     " "       "*"     "*"
## 8  ( 1 )  " "     " "       "*"     "*"
## 9  ( 1 )  " "     "*"       "*"     "*"
## 10  ( 1 ) " "     "*"       "*"     "*"
## 11  ( 1 ) "*"     "*"       "*"     "*"
## 12  ( 1 ) "*"     "*"       "*"     "*"
```

From the Above output we can say that if we only consider one column then best column is "ram" and if we have to select two best columns then we will take "ram" and "thickness". Similarly , it goes and according to our best no. of variables to select ,we can select using this model summary.

```
which.min(summary(best_model)$cp) # The model which contains the lowest cp variables
```

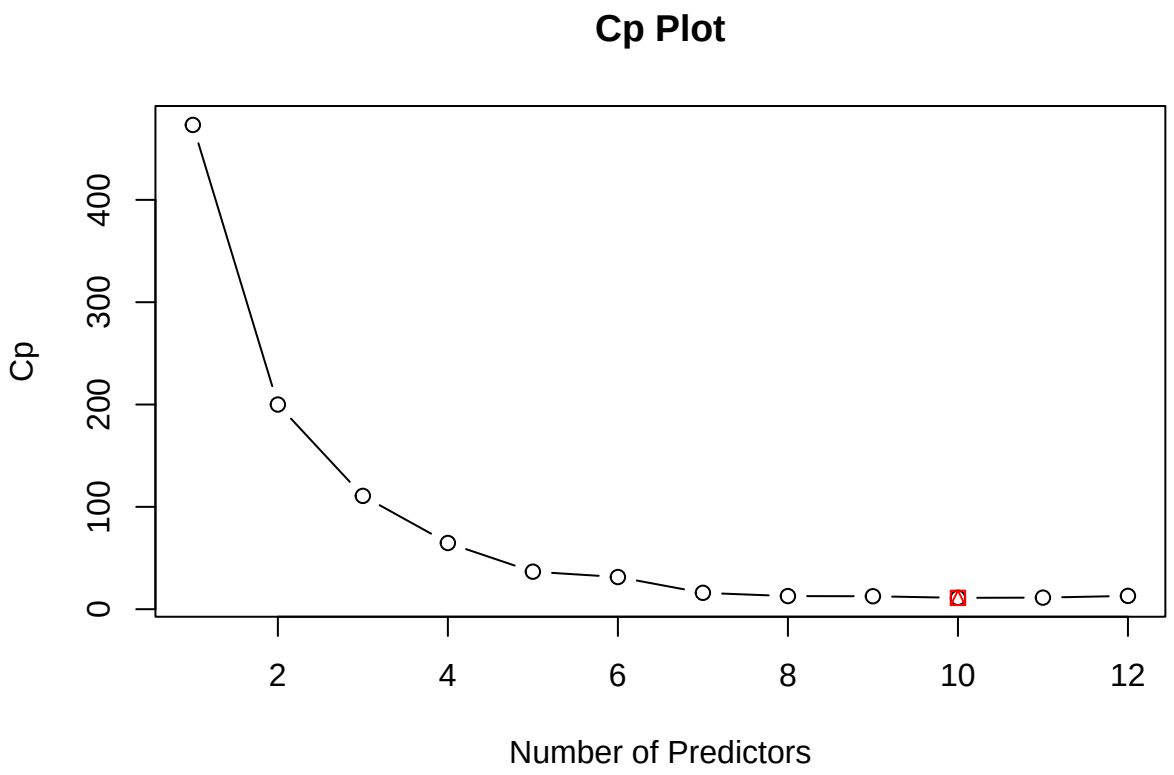
```
## [1] 10
```

```
coef(best_model, which.min(summary(best_model)$cp)) # The 10 selected variables  coefficients
```

```
##   (Intercept)          Sale   resoloution           ppi      cpu.core
## 1812.04486648   -0.02074425  -96.01276129    1.13475997   57.06636066
##      cpu.freq  internal.mem           ram     Front_Cam       battery
##  143.83971623    5.93891924   98.77624751   10.60825400    0.11428398
##     thickness
##  -77.65115406
```

These above are coefficients of 10 best selected columns

```
plot(summary(best_model)$cp, type = "b", main = "Cp Plot", xlab = "Number of Predictors", ylab = "Cp")
points(10, summary(best_model)$cp[10], pch = 14 , col = "red")
```

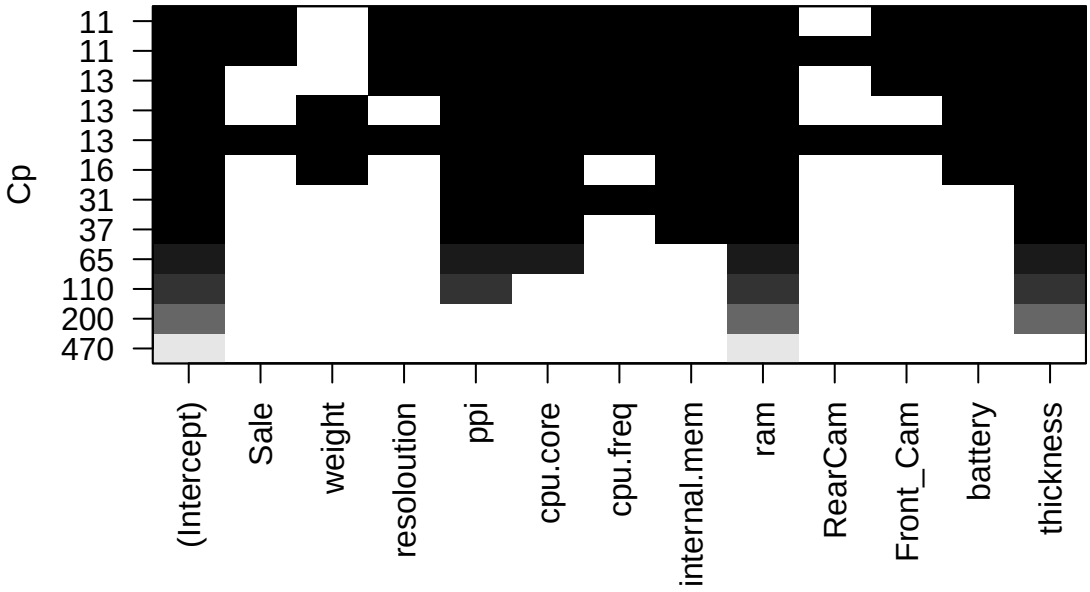



```
summary(best_model)$cp[10]
```

```
## [1] 11.10326
```

So, the value of Cp at number of predictors = 10 is 11.10326. So, we can clearly see that for no of predictors = 10 we get the lowest Cp

```
plot(best_model, scale = "Cp")
```



So this graph show that when we include very few variables like ram then we are getting Cp of 470 and as we are including more variables like 10 important variables which is shown in the black on the top rows then we are getting the minimum Cp of 11 and even it will reduce when we will increase more no of variables and that can be case of overfitting.

```
library(pls)
```

```
## Warning: package 'pls' was built under R version 4.4.3
```

```
##
## Attaching package: 'pls'
```

```
## The following object is masked from 'package:corrplot':
##
##     corrplot
```

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

```
pcr_5 <- pcr(Price ~ ., data = df, scale = TRUE, validation = "CV", ncomp = 5)
pcr_7 <- pcr(Price ~ ., data = df, scale = TRUE, validation = "CV", ncomp = 7)

# Check variance explained by the first 5 and 7 components
explained_variance_5 <- summary(pcr_5)$explvar
```

```
## Data:    X dimension: 161 12
##  Y dimension: 161 1
## Fit method: svdpc
## Number of components considered: 5
##
## VALIDATION: RMSEP
## Cross-validated using 10 random segments.
##        (Intercept)  1 comps  2 comps  3 comps  4 comps  5 comps
## CV           770.6    278.5    236.3    232.7    226.5    187.1
## adjCV        770.6    278.1    236.0    232.5    226.1    186.5
##
## TRAINING: % variance explained
##        1 comps  2 comps  3 comps  4 comps  5 comps
## X        48.24    67.71    78.61    85.55    90.02
## Price    87.40    90.91    91.15    91.92    94.50
```

So, here with 5 components total variance that is explained is 94.50% which is not greater than 95% but good enough because more than 90% variance is explained by only 5 Principal components

```
explained_variance_7 <- summary(pcr_7)$explvar
```

```
## Data:    X dimension: 161 12
##  Y dimension: 161 1
## Fit method: svdpc
## Number of components considered: 7
##
## VALIDATION: RMSEP
## Cross-validated using 10 random segments.
##        (Intercept)  1 comps  2 comps  3 comps  4 comps  5 comps  6 comps
## CV           770.6    282.6    246.4    245.8    243.8    189.4    189.6
## adjCV        770.6    282.0    245.3    244.7    242.4    188.5    188.8
##        7 comps
## CV       185.9
## adjCV    185.2
##
## TRAINING: % variance explained
##        1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps
## X        48.24    67.71    78.61    85.55    90.02    93.40    95.59
## Price    87.40    90.91    91.15    91.92    94.50    94.52    95.01
```

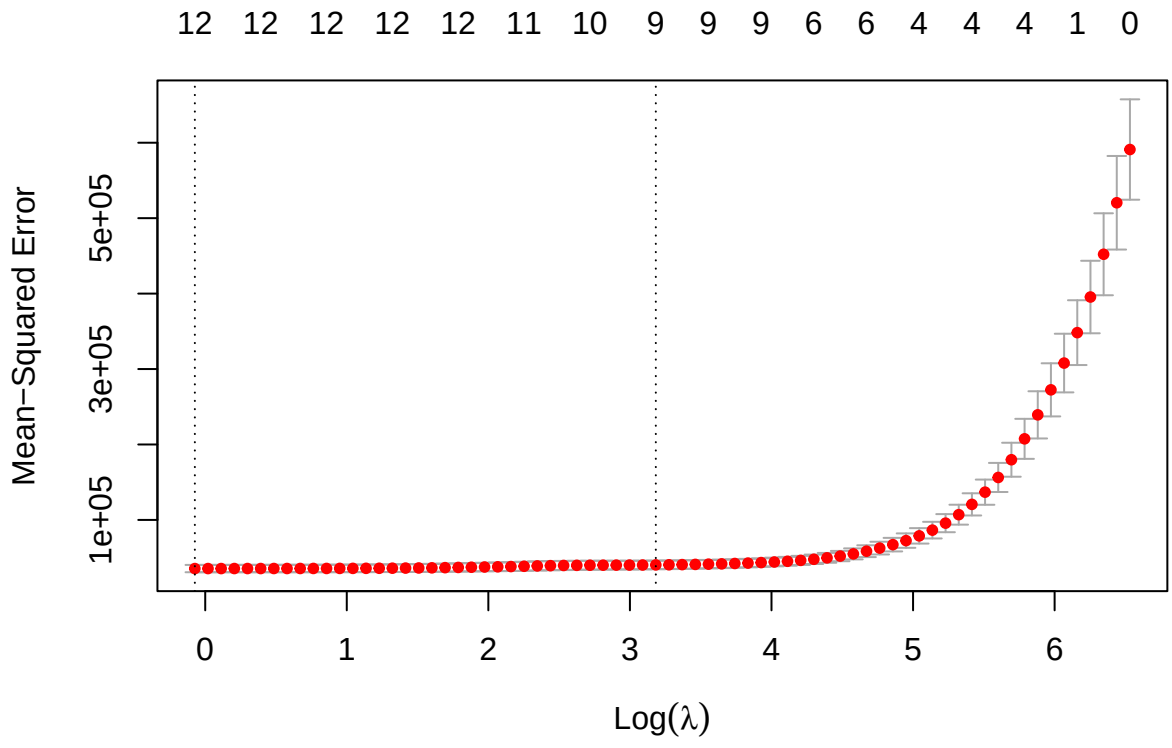
So, here with 7 components total variance that is explained is 95.01% which is greater than 95% .So, Good enough because more than 95% variance is explained by only 7 Principal components

```
library(glmnet)
```

```
## Loading required package: Matrix
```

```
## Loaded glmnet 4.1-8
```

```
X <- as.matrix(df[, !names(df) %in% "Price"])  # Predictors
y <- df$Price  # Response variable
# Fit Lasso model with cross-validation to find best lambda
set.seed(3)  # Ensure reproducibility
cv_lasso <- cv.glmnet(X, y, alpha = 1, standardize = TRUE, nfolds = 10)
plot(cv_lasso)
```




```
# Best lambda value
best_lambda <- cv_lasso$lambda.min
best_lambda
```

```
## [1] 0.9293484
```

Since we are getting high value of lambda , so we can say that stronger regularization, shrinking more coefficients to zero

```
lasso_model <- glmnet(X, y, alpha = 1, lambda = best_lambda, standardize = TRUE)
lasso_coef <- coef(lasso_model)
lasso_coef[lasso_coef != 0]
```

```
##  [1] 1705.74933498   -0.02135994   -0.45523499  -66.71562740    1.02291850
##  [6]   54.11494752  125.48189555    6.19528538   95.91893217    4.68732450
## [11]    8.58641496    0.12081951  -71.71894132
```

So We can clearly see from above that , there are some variables whose coefficients are very close to zero which means they are irrelevant columns . Like that 2nd coefficient etc.

R Markdown

This is an R Markdown document. Markdown is a simple formatting syntax for authoring HTML, PDF, and MS Word documents. For more details on using R Markdown see http://rmarkdown.rstudio.com.

When you click the **Knit** button a document will be generated that includes both content as well as the output of any embedded R code chunks within the document. You can embed an R code chunk like this: