Assignment\_2

Zhongya Ge 28479745

# Load required packages  
library(ggplot2)  
library(caret)

## Loading required package: lattice

library(MASS)  
library(ISLR)  
library(leaps)  
library(knitr)  
library(magrittr)  
library(kableExtra)

## Warning in !is.null(rmarkdown::metadata$output) && rmarkdown::metadata$output  
## %in% : 'length(x) = 3 > 1' in coercion to 'logical(1)'

#installed.packages("corrplot")  
library(corrgram)

##   
## Attaching package: 'corrgram'

## The following object is masked from 'package:lattice':  
##   
## panel.fill

library(ggcorrplot)  
library(caret)  
library(glmnet)

## Loading required package: Matrix

## Loaded glmnet 4.1-7

getwd()

## [1] "P:/STAT448-23S1 - Big Data/ass2"

# Q1

## (a)

# Load the data  
load("Residen.RData")  
# check if there is NAs in the dataset  
sum(is.na(Residen))

## [1] 0

# create PF variable list for PROJECT PHYSICAL AND FINANCIAL VARIABLES  
PF\_var <- Residen [,5:12]  
  
# create Eco variable list for variables from project lag 1  
Eco\_var<- Residen [,13:31]   
  
#create a full variable list for PROJECT PHYSICAL AND FINANCIAL VARIABLES and variables from project lag 1  
allVar <- Residen [,5:31]

# show dimensions after removing rows containing NA's  
dim(Residen)

## [1] 372 109

# show available output variables.  
names(Residen)

## [1] "START YEAR" "START QUARTER" "COMPLETION YEAR"   
## [4] "COMPLETION QUARTER" "V1" "V2"   
## [7] "V3" "V4" "V5"   
## [10] "V6" "V7" "V8"   
## [13] "V9" "V10" "V11"   
## [16] "V12" "V13" "V14"   
## [19] "V15" "V16" "V17"   
## [22] "V18" "V19" "V20"   
## [25] "V21" "V22" "V23"   
## [28] "V24" "V25" "V26"   
## [31] "V27" "V28" "V29"   
## [34] "V30" "V31" "V32"   
## [37] "V33" "V34" "V35"   
## [40] "V36" "V37" "V38"   
## [43] "V39" "V40" "V41"   
## [46] "V42" "V43" "V44"   
## [49] "V45" "V46" "V47"   
## [52] "V48" "V49" "V50"   
## [55] "V51" "V52" "V53"   
## [58] "V54" "V55" "V56"   
## [61] "V57" "V58" "V59"   
## [64] "V60" "V61" "V62"   
## [67] "V63" "V64" "V65"   
## [70] "V66" "V67" "V68"   
## [73] "V69" "V70" "V71"   
## [76] "V72" "V73" "V74"   
## [79] "V75" "V76" "V77"   
## [82] "V78" "V79" "V80"   
## [85] "V81" "V82" "V83"   
## [88] "V84" "V85" "V86"   
## [91] "V87" "V88" "V89"   
## [94] "V90" "V91" "V92"   
## [97] "V93" "V94" "V95"   
## [100] "V96" "V97" "V98"   
## [103] "V99" "V100" "V101"   
## [106] "V102" "V103" "V104"   
## [109] "V105"

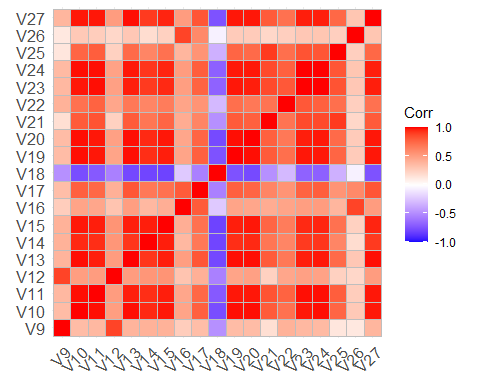
# Create a correlation matrix  
cor\_residen <- cor(Residen)  
corrplot <- ggcorrplot(cor\_residen, type = "lower", colors = c("#6D9EC1", "#FFFFFF", "#E46726"),  
 title = "Correlation Matrix") +  
 theme(axis.text.y = element\_text(angle = 45, vjust = 0.5, hjust=1,size = 8),axis.text.x = element\_text(size = 8))  
  
# Save the plot with a specific width and height  
ggsave("corrplot.png", width = 50, height = 50, dpi = 300, limitsize = FALSE)  
corrplot

Chart, diagram, shape

Description automatically generated

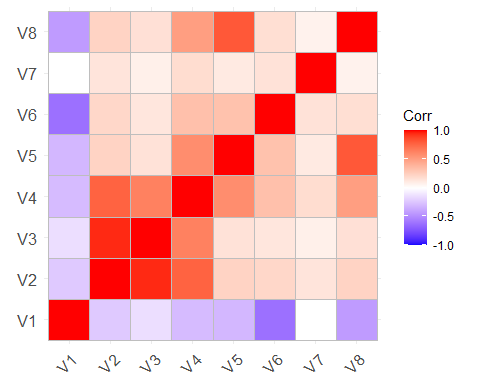
**It appears that there is a pattern in the correlation chart of 50 variables with the variables grouped by project lag into 5 distinct groups. The pattern is consistent across the 5 groups of variables, indicating a similarity in the correlation between variables across different lags. Therefore, the below analysis is based on the variables from project lag 1 and PF VARIABLES.**

# Create a correlation matrix for variables from project lag 1  
cor\_Eco\_var <- cor(Eco\_var)  
  
# Create a correlation matrix plot  
ggcorrplot(cor\_Eco\_var)



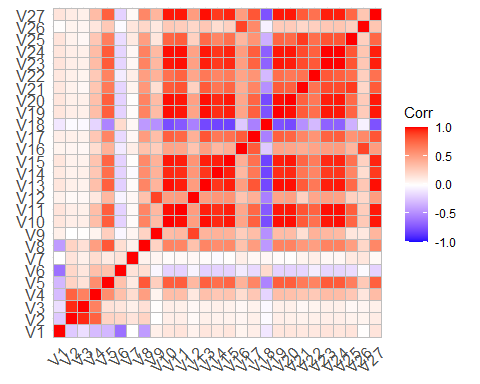
**For variables from project lag 1, the findings are: 1. V18 is negatively correlated with other variables, except V26. 2. All variables are positively corelated with each other, except V18.**

# Create a correlation matrix for PF VARIABLES  
cor\_PF\_var <- cor(PF\_var)  
  
# Create a correlation matrix plot  
ggcorrplot(cor\_PF\_var)



**For variables from PROJECT PHYSICAL AND FINANCIAL VARIABLES, the findings are: 1. V1 is negatively correlated with other variables, except V7. 2. All variables are positively corelated with each other, except V1.**

# Create a correlation matrix for PF VARIABLES and variables from project lag 1  
cor\_allVar <- cor(allVar)  
  
# Create a correlation matrix plot  
ggcorrplot(cor\_allVar)



**(b)**

#Create the Training and Test datasets  
  
# set seed for reproducible results.  
set.seed(2023)  
  
residen\_data <- Residen[-109]  
# Sample the dataset. Returns a list of row indices. 80:20 split.  
row.number <- sample(1:nrow(residen\_data), 0.8\*nrow(residen\_data))  
# create the train and test datasets.  
train <- residen\_data[row.number,]  
test <- residen\_data[-row.number,]  
# show dimensions of the train and test sets.  
dim(train)

## [1] 297 108

dim(test)

## [1] 75 108

#make a lm model  
model\_full <- lm(V104 ~., data=train)  
model\_full\_time <- system.time(lm(V104 ~ ., data = train))  
summary(model\_full)

##   
## Call:  
## lm(formula = V104 ~ ., data = train)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -853.69 -44.96 0.00 41.47 644.36   
##   
## Coefficients: (34 not defined because of singularities)  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) 5.638e+04 7.111e+04 0.793 0.428689   
## `START YEAR` -9.919e+02 1.053e+03 -0.942 0.347247   
## `START QUARTER` -3.181e+02 5.252e+02 -0.606 0.545341   
## `COMPLETION YEAR` 1.561e+02 2.121e+01 7.361 3.48e-12 \*\*\*  
## `COMPLETION QUARTER` 6.222e+01 1.046e+01 5.950 1.03e-08 \*\*\*  
## V1 -4.893e+00 2.565e+00 -1.908 0.057713 .   
## V2 8.378e-02 2.703e-02 3.100 0.002187 \*\*   
## V3 -2.790e-01 8.122e-02 -3.435 0.000706 \*\*\*  
## V4 -1.118e-02 4.034e-02 -0.277 0.781814   
## V5 -7.090e-01 3.789e-01 -1.871 0.062655 .   
## V6 9.205e-02 6.995e-02 1.316 0.189563   
## V7 NA NA NA NA   
## V8 1.202e+00 1.943e-02 61.900 < 2e-16 \*\*\*  
## V9 5.749e-02 2.634e-01 0.218 0.827398   
## V10 -6.401e+01 1.780e+02 -0.360 0.719399   
## V11 -4.793e-01 1.154e+02 -0.004 0.996690   
## V12 -6.232e+01 1.234e+02 -0.505 0.614124   
## V13 -4.494e-03 3.046e-02 -0.148 0.882829   
## V14 1.652e-03 2.757e-01 0.006 0.995224   
## V15 1.263e+00 6.110e+01 0.021 0.983524   
## V16 2.677e-01 3.937e+00 0.068 0.945854   
## V17 -6.795e-03 9.331e-02 -0.073 0.942013   
## V18 1.174e+02 3.359e+02 0.350 0.726970   
## V19 1.725e-01 1.344e+00 0.128 0.897964   
## V20 -2.879e-01 1.572e+00 -0.183 0.854849   
## V21 2.456e-02 4.315e-02 0.569 0.569802   
## V22 5.564e-02 4.355e-01 0.128 0.898454   
## V23 2.525e+01 1.111e+02 0.227 0.820354   
## V24 5.199e+01 2.190e+02 0.237 0.812591   
## V25 -9.982e-02 3.675e-01 -0.272 0.786185   
## V26 -2.338e-03 6.409e-02 -0.036 0.970936   
## V27 -4.048e-04 3.094e-03 -0.131 0.896038   
## V28 6.365e-02 1.524e-01 0.418 0.676588   
## V29 1.581e+02 3.269e+02 0.483 0.629259   
## V30 -3.673e+01 9.114e+01 -0.403 0.687370   
## V31 -2.977e+01 2.208e+02 -0.135 0.892873   
## V32 1.179e-02 2.559e-02 0.461 0.645467   
## V33 -1.549e-02 2.082e-01 -0.074 0.940776   
## V34 7.136e+00 6.796e+01 0.105 0.916462   
## V35 -2.608e-01 3.413e+00 -0.076 0.939167   
## V36 9.787e-03 5.305e-02 0.184 0.853804   
## V37 1.274e+02 1.773e+02 0.718 0.473292   
## V38 -3.047e-01 3.014e+00 -0.101 0.919552   
## V39 2.354e-01 1.135e+00 0.207 0.835885   
## V40 -9.146e-03 2.573e-01 -0.036 0.971678   
## V41 -1.165e-01 2.302e-01 -0.506 0.613349   
## V42 6.808e+01 1.997e+02 0.341 0.733532   
## V43 -9.005e+01 3.421e+02 -0.263 0.792641   
## V44 -5.816e-02 2.540e-01 -0.229 0.819055   
## V45 -2.701e-03 6.042e-02 -0.045 0.964391   
## V46 1.250e-03 2.995e-03 0.417 0.676717   
## V47 9.969e-02 1.283e-01 0.777 0.438100   
## V48 -8.665e+01 2.449e+02 -0.354 0.723831   
## V49 5.625e+01 1.387e+02 0.406 0.685495   
## V50 -1.382e+02 2.169e+02 -0.637 0.524684   
## V51 9.017e-03 2.623e-02 0.344 0.731401   
## V52 -2.018e-01 5.685e-01 -0.355 0.722960   
## V53 -9.319e+00 7.437e+01 -0.125 0.900395   
## V54 3.154e+00 7.253e+00 0.435 0.664072   
## V55 -2.087e-02 4.977e-02 -0.419 0.675410   
## V56 7.116e-01 2.006e+02 0.004 0.997173   
## V57 -5.984e-01 2.406e+00 -0.249 0.803780   
## V58 1.112e+00 3.828e+00 0.291 0.771601   
## V59 -7.529e-02 1.865e-01 -0.404 0.686763   
## V60 1.831e-02 1.385e-01 0.132 0.894974   
## V61 9.585e+01 1.933e+02 0.496 0.620534   
## V62 5.532e+01 3.038e+02 0.182 0.855669   
## V63 7.108e-02 7.915e-02 0.898 0.370135   
## V64 -1.304e-02 6.977e-02 -0.187 0.851893   
## V65 -1.048e-03 1.674e-03 -0.626 0.531997   
## V66 3.394e-02 3.237e-01 0.105 0.916587   
## V67 -6.756e+01 1.795e+02 -0.376 0.706968   
## V68 -4.003e+01 2.072e+02 -0.193 0.847009   
## V69 3.961e+01 1.202e+02 0.330 0.742079   
## V70 -1.838e-02 1.399e-02 -1.314 0.190213   
## V71 NA NA NA NA   
## V72 NA NA NA NA   
## V73 NA NA NA NA   
## V74 NA NA NA NA   
## V75 NA NA NA NA   
## V76 NA NA NA NA   
## V77 NA NA NA NA   
## V78 NA NA NA NA   
## V79 NA NA NA NA   
## V80 NA NA NA NA   
## V81 NA NA NA NA   
## V82 NA NA NA NA   
## V83 NA NA NA NA   
## V84 NA NA NA NA   
## V85 NA NA NA NA   
## V86 NA NA NA NA   
## V87 NA NA NA NA   
## V88 NA NA NA NA   
## V89 NA NA NA NA   
## V90 NA NA NA NA   
## V91 NA NA NA NA   
## V92 NA NA NA NA   
## V93 NA NA NA NA   
## V94 NA NA NA NA   
## V95 NA NA NA NA   
## V96 NA NA NA NA   
## V97 NA NA NA NA   
## V98 NA NA NA NA   
## V99 NA NA NA NA   
## V100 NA NA NA NA   
## V101 NA NA NA NA   
## V102 NA NA NA NA   
## V103 NA NA NA NA   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 151.2 on 223 degrees of freedom  
## Multiple R-squared: 0.9878, Adjusted R-squared: 0.9838   
## F-statistic: 248 on 73 and 223 DF, p-value: < 2.2e-16

**Explain the output:**

1. **The “Residuals” section shows that summary statistics of the residuals, and the minimum residual is -853.69, and the maximum residual is 644.36, and the median residual is 0.00, which suggest the model might not be perfectly fitting the data due to the far distance of max and min from zero.**
2. **The “coefficients” section reports the estimated regression coefficients for each independent variable in the model. But there are 34 coefficients are not defined due to singularities, which might indicate multicollinearity issues in the dataset.**
3. **The “Residual standard error” is 151.2. The “Multiple R-squared” and “Adjusted R-squared” indicate that the model explains a high proportion of the variance in the dependent variable (98.78%) and that the adjusted R-squared is slightly lower due to the inclusion of additional variables.**
4. **The “F-statistic” tests the overall significance of the model, compared to a null model with no independent variables. In this case, the F-statistic is 248, and the associated p-value is very small (< 2.2e-16), indicating strong evidence against the null hypothesis.**

## (c)

### Backwards Selection:

# Fit a linear regression model using backwards selection  
backward\_model <- stepAIC(model\_full, direction = "backward", trace = 0)

### Stepwise Selection:

# first create null model.  
model\_null <- lm(V104 ~ 1, data=train)  
  
# Stepwise selection using AIC with null model as lower scope  
stepwise\_model <- stepAIC(model\_null, direction="both", scope=list(upper = model\_full, lower = model\_null), trace = 0)

### Output:

summary(backward\_model)

##   
## Call:  
## lm(formula = V104 ~ `START YEAR` + `START QUARTER` + `COMPLETION YEAR` +   
## `COMPLETION QUARTER` + V1 + V2 + V3 + V8 + V9 + V16 + V17 +   
## V18 + V24 + V25 + V29 + V30 + V32 + V37 + V38 + V39 + V42 +   
## V43 + V47 + V49 + V50 + V52 + V53 + V54 + V55 + V57 + V58 +   
## V59 + V62 + V63 + V64 + V66 + V67 + V68 + V70, data = train)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -958.41 -43.96 -0.23 42.11 595.06   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) 4.112e+04 9.302e+03 4.421 1.45e-05 \*\*\*  
## `START YEAR` -7.714e+02 1.391e+02 -5.546 7.25e-08 \*\*\*  
## `START QUARTER` -2.032e+02 3.653e+01 -5.562 6.69e-08 \*\*\*  
## `COMPLETION YEAR` 1.622e+02 1.839e+01 8.821 < 2e-16 \*\*\*  
## `COMPLETION QUARTER` 5.959e+01 9.158e+00 6.508 3.97e-10 \*\*\*  
## V1 -2.945e+00 1.899e+00 -1.551 0.122230   
## V2 7.125e-02 1.683e-02 4.233 3.22e-05 \*\*\*  
## V3 -2.432e-01 6.175e-02 -3.939 0.000106 \*\*\*  
## V8 1.183e+00 1.618e-02 73.132 < 2e-16 \*\*\*  
## V9 6.669e-02 2.494e-02 2.674 0.007986 \*\*   
## V16 9.008e-01 3.664e-01 2.459 0.014607 \*   
## V17 -1.179e-02 4.673e-03 -2.522 0.012271 \*   
## V18 8.448e+01 3.462e+01 2.440 0.015371 \*   
## V24 6.845e+01 2.243e+01 3.052 0.002513 \*\*   
## V25 -1.183e-01 2.892e-02 -4.090 5.77e-05 \*\*\*  
## V29 2.211e+01 1.622e+01 1.363 0.174135   
## V30 -1.679e+01 7.382e+00 -2.275 0.023735 \*   
## V32 8.041e-03 1.657e-03 4.854 2.10e-06 \*\*\*  
## V37 6.896e+01 2.922e+01 2.360 0.019014 \*   
## V38 -6.467e-01 2.277e-01 -2.840 0.004872 \*\*   
## V39 5.525e-01 1.646e-01 3.358 0.000905 \*\*\*  
## V42 9.372e+01 2.229e+01 4.204 3.62e-05 \*\*\*  
## V43 -1.016e+02 2.279e+01 -4.457 1.24e-05 \*\*\*  
## V47 6.335e-02 2.395e-02 2.646 0.008658 \*\*   
## V49 3.328e+01 9.700e+00 3.431 0.000699 \*\*\*  
## V50 -9.766e+01 2.995e+01 -3.261 0.001261 \*\*   
## V52 -1.520e-01 3.396e-02 -4.476 1.15e-05 \*\*\*  
## V53 -9.154e+00 4.414e+00 -2.074 0.039094 \*   
## V54 2.220e+00 7.524e-01 2.950 0.003467 \*\*   
## V55 -1.055e-02 3.993e-03 -2.642 0.008750 \*\*   
## V57 -5.704e-01 1.885e-01 -3.027 0.002724 \*\*   
## V58 1.310e+00 2.813e-01 4.658 5.12e-06 \*\*\*  
## V59 -3.945e-02 2.051e-02 -1.923 0.055588 .   
## V62 8.778e+01 2.157e+01 4.070 6.27e-05 \*\*\*  
## V63 4.911e-02 2.061e-02 2.383 0.017903 \*   
## V64 -9.482e-03 3.611e-03 -2.626 0.009162 \*\*   
## V66 5.505e-02 2.137e-02 2.576 0.010544 \*   
## V67 -6.639e+01 1.612e+01 -4.118 5.15e-05 \*\*\*  
## V68 -3.020e+01 6.932e+00 -4.357 1.91e-05 \*\*\*  
## V70 -1.044e-02 2.438e-03 -4.281 2.62e-05 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 145.2 on 257 degrees of freedom  
## Multiple R-squared: 0.9871, Adjusted R-squared: 0.9851   
## F-statistic: 502.8 on 39 and 257 DF, p-value: < 2.2e-16

summary(stepwise\_model)

##   
## Call:  
## lm(formula = V104 ~ V8 + V7 + V21 + V72 + V1 + V24 + V51 + V20 +   
## V37 + V34 + V4 + `COMPLETION QUARTER` + V5 + V46, data = train)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -963.86 -49.20 -3.23 43.55 653.60   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) -6.903e+02 2.017e+02 -3.422 0.000714 \*\*\*  
## V8 1.192e+00 1.628e-02 73.182 < 2e-16 \*\*\*  
## V7 4.150e+01 4.365e+00 9.506 < 2e-16 \*\*\*  
## V21 9.683e-03 5.844e-03 1.657 0.098661 .   
## V72 -1.342e+01 9.594e-01 -13.986 < 2e-16 \*\*\*  
## V1 -5.905e+00 2.067e+00 -2.856 0.004608 \*\*   
## V24 6.237e+00 1.676e+00 3.722 0.000238 \*\*\*  
## V51 1.436e-03 2.214e-04 6.489 3.88e-10 \*\*\*  
## V20 -3.225e-01 8.249e-02 -3.910 0.000116 \*\*\*  
## V37 4.857e+01 1.464e+01 3.317 0.001029 \*\*   
## V34 2.005e+00 1.016e+00 1.974 0.049415 \*   
## V4 4.869e-02 1.975e-02 2.466 0.014273 \*   
## `COMPLETION QUARTER` 1.488e+01 7.696e+00 1.933 0.054266 .   
## V5 -3.234e-01 2.156e-01 -1.500 0.134856   
## V46 -1.376e-04 9.915e-05 -1.388 0.166341   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 145.3 on 282 degrees of freedom  
## Multiple R-squared: 0.9858, Adjusted R-squared: 0.9851   
## F-statistic: 1396 on 14 and 282 DF, p-value: < 2.2e-16

**For the output, the findings in backward model are:**

* 1. **the formula is lm(formula = V104 ~ START YEAR + START QUARTER + COMPLETION YEAR + COMPLETION QUARTER + V1 + V2 + V3 + V8 + V9 + V16 + V17 + V18 + V24 + V25 + V29 + V30 + V32 + V37 + V38 + V39 + V42 + V43 + V47 + V49 + V50 + V52 + V53 + V54 + V55 + V57 + V58 + V59 + V62 + V63 + V64 + V66 + V67 + V68 + V70, data = train)**
  2. **min residual is -958.41, max residual is 595.06, and the median residual is -0.23. Compared with stepward model, it has a smaller range of residuals.**
  3. **Residual standard error is 145.2, which is slightly lower than stepward model.**
  4. **Multiple R-squared: 0.9871, Adjusted R-squared: 0.9851. After adjusting, the R-squared is the same as stepward model. 5. both F-statistics have very low p-values, indicating strong evidence against the null hypothesis.**

**The findings in stepward model are:**

* 1. **the formula is lm(formula = V104 ~ V8 + V7 + V21 + V72 + V1 + V24 + V51 + V20 + V37 + V34 + V4 + COMPLETION QUARTER + V5 + V46, data = train)**
  2. **min residual is -963.86, max residual is 653.60, and the median residual is -3.23.**
  3. **Residual standard error is 145.3.**
  4. **Multiple R-squared: 0.9858, Adjusted R-squared: 0.9851.**

### Computational Time:

backward\_time <- system.time(step(model\_full, direction = "backward", trace = 0))  
stepwise\_time <- system.time(stepAIC(model\_null, direction="both", scope=list(upper=model\_full, lower=model\_null), trace = 0))  
cat("Stepwise selection computational time:", stepwise\_time[[3]], "\n")

## Stepwise selection computational time: 1.21

cat("Backward selection computational time:", backward\_time[[3]], "\n")

## Backward selection computational time: 12.97

**Stepwise selection performs better. Stepwise selection is computationally faster than backward selection. Backward selection evaluates all possible subsets of features, which can be computationally expensive. However, stepwise selection evaluates only a subset of features at each step, which can be more efficient.**

### Holdout Mean Square Error:

# Prediction try predictions from different models  
# Evaluate backward\_model on test data  
pred\_backward\_test <- predict(backward\_model, newdata=test)  
mse\_backward\_test <- mean((test$V104 - pred\_backward\_test)^2)  
rmse\_backward\_test <- sqrt(mean((pred\_backward\_test - test$V104)^2))  
  
# Evaluate stepwise\_model on test data  
pred\_stepwise\_test <- predict(stepwise\_model, newdata=test)  
mse\_stepwise\_test <- mean((test$V104 - pred\_stepwise\_test)^2)  
rmse\_stepwise\_test <- sqrt(mean((pred\_stepwise\_test - test$V104)^2))  
  
# Evaluate backward\_model on train data  
pred\_backward\_train <- predict(backward\_model, newdata=train)  
mse\_backward\_train <- mean((train$V104 - pred\_backward\_train)^2)  
rmse\_backward\_train<- sqrt(mean((pred\_backward\_train - train$V104)^2))  
  
# Evaluate stepwise\_model on train data  
pred\_stepwise\_train <- predict(stepwise\_model, newdata=train)  
mse\_stepwise\_train <- mean((train$V104 - pred\_stepwise\_train)^2)  
rmse\_stepwise\_train <- sqrt(mean((pred\_stepwise\_train - train$V104)^2))  
  
#c(Model = "Backwards Selection (test)" , MSE = round(mse\_backward\_test, 2), RMSE =round(rmse\_backward\_test, 2), R2 = round(summary(backward\_model)$r.squared,2))  
#c(Model = "Stepwise Selection (test)" , MSE = round(mse\_stepwise\_test, 2), RMSE =round(rmse\_stepwise\_test, 2),R2 = round(summary(stepwise\_model)$r.squared,2))  
results <- data.frame(  
 Model = c("Backwards Selection", "Stepwise Selection"),  
 MSE = round(c(mse\_backward\_test, mse\_stepwise\_test), 2),  
 RMSE = round(c(rmse\_backward\_test, rmse\_stepwise\_test), 2),  
 R2 = round(c(summary(backward\_model)$r.squared, summary(stepwise\_model)$r.squared), 2),  
 Time = c(stepwise\_time[[3]], backward\_time[[3]])  
)

print(results)

## Model MSE RMSE R2 Time  
## 1 Backwards Selection 23356.80 152.83 0.99 1.21  
## 2 Stepwise Selection 19953.32 141.26 0.99 12.97

**Based on the above information of MSE, RMSE and R^2, the stepwise selection performs better, because stepwise has less value in all three figures.**

### 

### Cross Validation Mean Square Error

# set the number of folds  
k <- 10  
  
# set seed  
set.seed(2023)  
  
# assign every row in residen\_data a random fold number  
folds <- sample(1:k, nrow(residen\_data), replace = TRUE)  
   
# create empty vectors to store the mean squared error for each fold  
stepwise\_mse <- vector(mode = "numeric", length = k)  
backward\_mse <- vector(mode = "numeric", length = k)  
   
# loop through each fold  
for (i in 1:k) {  
   
 # split the data into training and testing sets for this fold  
 cv\_test\_data <- residen\_data[folds == i, ]  
 cv\_train\_data <- residen\_data[folds != i, ]  
   
 # train the stepwise regression model on the training data  
 model\_full\_1 <- lm(V104 ~ ., data = cv\_train\_data)  
 model\_null\_1 <- lm(V104 ~ 1, data=cv\_train\_data)  
   
 stepwise\_model2 <- stepAIC(model\_null\_1, direction="both", scope=list(upper=model\_full\_1, lower=model\_null\_1), trace = 0)  
   
 # make predictions on the testing data using the stepwise model  
 stepwise\_predictions <- predict(stepwise\_model2, newdata = cv\_test\_data)  
   
 # calculate the mean squared error of the stepwise model for this fold  
 stepwise\_mse[i] <- mean((cv\_test\_data$V104 - stepwise\_predictions)^2)  
   
 # train the backward regression model on the training data  
 backward\_model2 <- stepAIC(model\_full\_1, direction = "backward", trace = 0)  
   
 # make predictions on the testing data using the backward model  
 backward\_predictions <- predict(backward\_model2, newdata = cv\_test\_data)  
   
 # calculate the mean squared error of the backward model for this fold  
 backward\_mse[i] <- mean((cv\_test\_data$V104 - backward\_predictions)^2)  
}  
   
# calculate the cross-validated mean squared error for each model  
stepwise\_cv\_mse <- mean(stepwise\_mse)  
backward\_cv\_mse <- mean(backward\_mse)  
cat("Stepwise selection CV MSE:", stepwise\_cv\_mse, "\n")

## Stepwise selection CV MSE: 36190.24

cat("Backward selection CV MSE:", backward\_cv\_mse, "\n")

## Backward selection CV MSE: 39200.89

**Conclusion:**

**Stepwise has a better performance in this case because its CV MSE is less than Backward.**

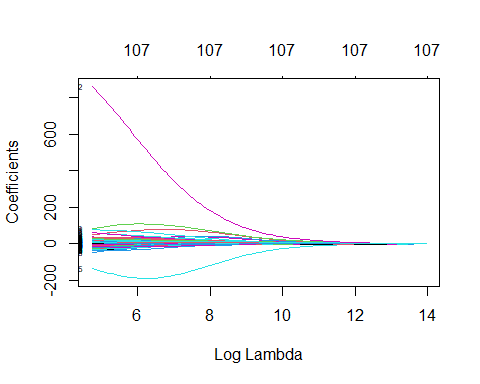
## 

## (d)

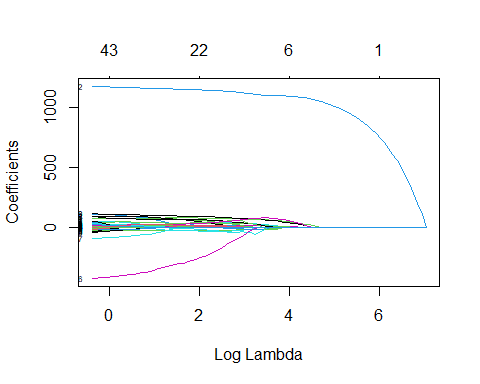
# Standardize predictor variables  
x\_train <- as.matrix(scale(train[-108]))  
y\_train <- train$V104  
x\_test <- as.matrix(scale(test[-108]))  
y\_test <- test$V104

# Fit Ridge and LASSO regression model  
set.seed(2023)  
ridge\_model <- glmnet(x = x\_train, y = y\_train, alpha = 0 )  
lasso\_model <- glmnet(x = x\_train, y = y\_train, alpha = 1)

plot(ridge\_model, xvar = "lambda", label = TRUE)



plot(lasso\_model, xvar = "lambda", label = TRUE)

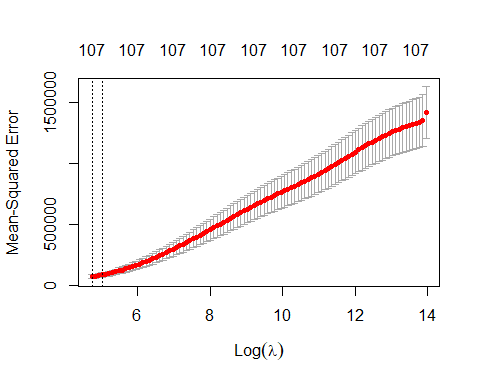


# Cross-validation for Ridge  
set.seed(2023)  
cv\_ridge <- cv.glmnet(x = x\_train, y = y\_train, alpha = 0, nfolds = 10)  
ridge\_mean\_cvmse <- cv\_ridge$cvm[cv\_ridge$lambda == cv\_ridge$lambda.min]  
  
# Cross-validation for LASSO  
set.seed(2023)  
cv\_lasso <- cv.glmnet(x = x\_train, y = y\_train, alpha = 1, nfolds = 10)  
lasso\_mean\_cvmse <- cv\_lasso$cvm[cv\_lasso$lambda == cv\_lasso$lambda.min]

cat("Ridge min lambda:", cv\_ridge$lambda.min, "\n")

## Ridge min lambda: 115.4997

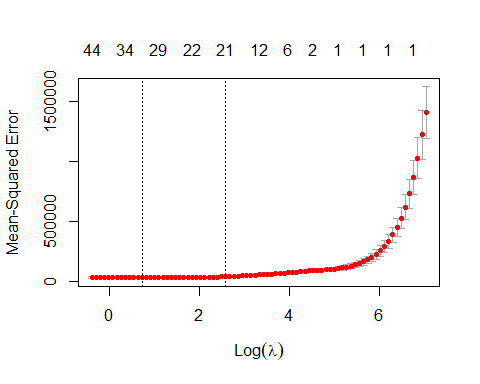
plot(cv\_ridge)



cat("LASSO min lambda:", cv\_lasso$lambda.min, "\n")

## LASSO min lambda: 2.065885

plot(cv\_lasso)



# Fit Ridge and LASSO regression model  
set.seed(2023)  
ridge\_model\_best <- glmnet(x = x\_train, y = y\_train, alpha = 0 , lambda = cv\_ridge$lambda.min)  
lasso\_model\_best <- glmnet(x = x\_train, y = y\_train, alpha = 1, lambda = cv\_lasso$lambda.min)

# Predict on test set  
ridge\_pred <- predict(ridge\_model\_best, newx = x\_test)  
lasso\_pred <- predict(lasso\_model\_best, newx = x\_test)  
  
# Compute test MSE for each model  
ridge\_mse <- mean((test$V104 - ridge\_pred)^2)  
lasso\_mse <- mean((test$V104 - lasso\_pred)^2)  
  
# Compute computational time for each model  
ridge\_time <- system.time(ridge\_model\_best)  
lasso\_time <- system.time(lasso\_model\_best)

ridge\_time

## user system elapsed   
## 0 0 0

lasso\_time

## user system elapsed   
## 0 0 0

**Both computational time is not expensive.**

# Print results  
cat("Ridge MSE:", ridge\_mse, "\n")

## Ridge MSE: 66721.59

cat("LASSO MSE:", lasso\_mse, "\n")

## LASSO MSE: 29940.84

cat("Ridge time:", ridge\_time[[3]], "\n")

## Ridge time: 0.361

cat("LASSO time:", lasso\_time[[3]], "\n")

## LASSO time: 0.142

cat("Ridge CV MSE:", ridge\_mean\_cvmse, "\n")

## Ridge CV MSE: 69787.75

cat("LASSO CV MSE:", lasso\_mean\_cvmse, "\n")

## LASSO CV MSE: 32459.32

cat("Ridge min lambda:", cv\_ridge$lambda.min, "\n")

## Ridge min lambda: 115.4997

cat("LASSO min lambda:", cv\_lasso$lambda.min, "\n")

## LASSO min lambda: 2.065885

**Based on the above, LASSO with 2.066 lambda has a lower CV MSE (32,459.32) and holdout MSE (29,940.84). And the computational time for both is not expensive. Therefore, LASSO is better than Ridge.**

# Q2

## (a)

parkinsons <- read.csv("parkinsons.csv", header = TRUE)  
names(parkinsons)

## [1] "X" "X1" "X2" "X3" "X4" "X5" "X6" "X7" "X8"   
## [10] "X9" "X10" "X11" "X12" "X13" "X14" "X15" "X16" "X17"   
## [19] "X18" "X19" "X20" "X21" "X22" "X23" "X24" "X25" "X26"   
## [28] "X27" "X28" "X29" "X30" "X31" "X32" "X33" "X34" "X35"   
## [37] "X36" "X37" "X38" "X39" "X40" "X41" "X42" "X43" "X44"   
## [46] "X45" "X46" "X47" "X48" "X49" "X50" "X51" "X52" "X53"   
## [55] "X54" "X55" "X56" "X57" "X58" "X59" "X60" "X61" "X62"   
## [64] "X63" "X64" "X65" "X66" "X67" "X68" "X69" "X70" "X71"   
## [73] "X72" "X73" "X74" "X75" "X76" "X77" "X78" "X79" "X80"   
## [82] "X81" "X82" "X83" "X84" "X85" "X86" "X87" "X88" "X89"   
## [91] "X90" "X91" "X92" "X93" "X94" "X95" "X96" "X97" "UPDRS"

dim(parkinsons)

## [1] 42 99

sum(is.na(parkinsons))

## [1] 0

#create lists for x features and y  
parkinsons <- parkinsons[, -1]  
parkinsons\_x <- parkinsons[1:97]  
parkinsons\_y <- parkinsons$UPDRS

# Split the data into a training set (30 patients) and a test set (12 patients)  
set.seed(2023) # Set the RNG seed for reproducibility  
train\_idx <- sample(1:nrow(parkinsons), 30, replace = FALSE)  
#pk.train <- parkinsons[train\_idx, ]  
#pk.test <- parkinsons[-train\_idx, ]  
  
# Standardize the predictor variables  
parkinsons\_x <- scale(parkinsons\_x)  
  
pk.x\_train <- as.matrix(parkinsons\_x[train\_idx, ])  
pk.x\_test <- as.matrix(parkinsons\_x[-train\_idx, ])  
pk.y\_train <- parkinsons\_y[train\_idx]  
pk.y\_test <- parkinsons\_y[-train\_idx]  
  
pk.train <- data.frame(pk.x\_train, pk.y\_train)  
pk.test <- data.frame(pk.x\_test, pk.y\_test)  
  
#check the number of patients for train and test  
dim(pk.x\_train)

## [1] 30 97

dim(pk.x\_test)

## [1] 12 97

#head(pk.train)

parkinsons.lm <- lm(pk.y\_train~., pk.train)  
# check the summary output  
summary(parkinsons.lm)

##   
## Call:  
## lm(formula = pk.y\_train ~ ., data = pk.train)  
##   
## Residuals:  
## ALL 30 residuals are 0: no residual degrees of freedom!  
##   
## Coefficients: (68 not defined because of singularities)  
## Estimate Std. Error t value Pr(>|t|)  
## (Intercept) 18.52 NaN NaN NaN  
## X1 -219.44 NaN NaN NaN  
## X2 -84.30 NaN NaN NaN  
## X3 187.07 NaN NaN NaN  
## X4 67.09 NaN NaN NaN  
## X5 -40.65 NaN NaN NaN  
## X6 92.09 NaN NaN NaN  
## X7 142.98 NaN NaN NaN  
## X8 24.21 NaN NaN NaN  
## X9 -74.35 NaN NaN NaN  
## X10 76.97 NaN NaN NaN  
## X11 135.30 NaN NaN NaN  
## X12 -144.51 NaN NaN NaN  
## X13 25424.73 NaN NaN NaN  
## X14 -4253.07 NaN NaN NaN  
## X15 -9210.54 NaN NaN NaN  
## X16 -8589.50 NaN NaN NaN  
## X17 12783.03 NaN NaN NaN  
## X18 -10.70 NaN NaN NaN  
## X19 35.73 NaN NaN NaN  
## X20 61.38 NaN NaN NaN  
## X21 -29.10 NaN NaN NaN  
## X22 -6.35 NaN NaN NaN  
## X23 -262.47 NaN NaN NaN  
## X24 58.86 NaN NaN NaN  
## X25 -25461.00 NaN NaN NaN  
## X26 4263.46 NaN NaN NaN  
## X27 9152.03 NaN NaN NaN  
## X28 8460.06 NaN NaN NaN  
## X29 -12714.79 NaN NaN NaN  
## X30 NA NA NA NA  
## X31 NA NA NA NA  
## X32 NA NA NA NA  
## X33 NA NA NA NA  
## X34 NA NA NA NA  
## X35 NA NA NA NA  
## X36 NA NA NA NA  
## X37 NA NA NA NA  
## X38 NA NA NA NA  
## X39 NA NA NA NA  
## X40 NA NA NA NA  
## X41 NA NA NA NA  
## X42 NA NA NA NA  
## X43 NA NA NA NA  
## X44 NA NA NA NA  
## X45 NA NA NA NA  
## X46 NA NA NA NA  
## X47 NA NA NA NA  
## X48 NA NA NA NA  
## X49 NA NA NA NA  
## X50 NA NA NA NA  
## X51 NA NA NA NA  
## X52 NA NA NA NA  
## X53 NA NA NA NA  
## X54 NA NA NA NA  
## X55 NA NA NA NA  
## X56 NA NA NA NA  
## X57 NA NA NA NA  
## X58 NA NA NA NA  
## X59 NA NA NA NA  
## X60 NA NA NA NA  
## X61 NA NA NA NA  
## X62 NA NA NA NA  
## X63 NA NA NA NA  
## X64 NA NA NA NA  
## X65 NA NA NA NA  
## X66 NA NA NA NA  
## X67 NA NA NA NA  
## X68 NA NA NA NA  
## X69 NA NA NA NA  
## X70 NA NA NA NA  
## X71 NA NA NA NA  
## X72 NA NA NA NA  
## X73 NA NA NA NA  
## X74 NA NA NA NA  
## X75 NA NA NA NA  
## X76 NA NA NA NA  
## X77 NA NA NA NA  
## X78 NA NA NA NA  
## X79 NA NA NA NA  
## X80 NA NA NA NA  
## X81 NA NA NA NA  
## X82 NA NA NA NA  
## X83 NA NA NA NA  
## X84 NA NA NA NA  
## X85 NA NA NA NA  
## X86 NA NA NA NA  
## X87 NA NA NA NA  
## X88 NA NA NA NA  
## X89 NA NA NA NA  
## X90 NA NA NA NA  
## X91 NA NA NA NA  
## X92 NA NA NA NA  
## X93 NA NA NA NA  
## X94 NA NA NA NA  
## X95 NA NA NA NA  
## X96 NA NA NA NA  
## X97 NA NA NA NA  
##   
## Residual standard error: NaN on 0 degrees of freedom  
## Multiple R-squared: 1, Adjusted R-squared: NaN   
## F-statistic: NaN on 29 and 0 DF, p-value: NA

summary(parkinsons.lm)$r.squared

## [1] 1

**The R-squared value is 1, which means the linear model fits the training data perfectly and explains the variation in the training data. However, this model is not going to be useful. The reason is that a model that is overfit to the training data may not generalize well to new data. Overfitting occurs when a model is too complex and captures noise in the training data, rather than the underlying relationships between the variables. This can lead to poor performance on new data, which is called “high variance”.**

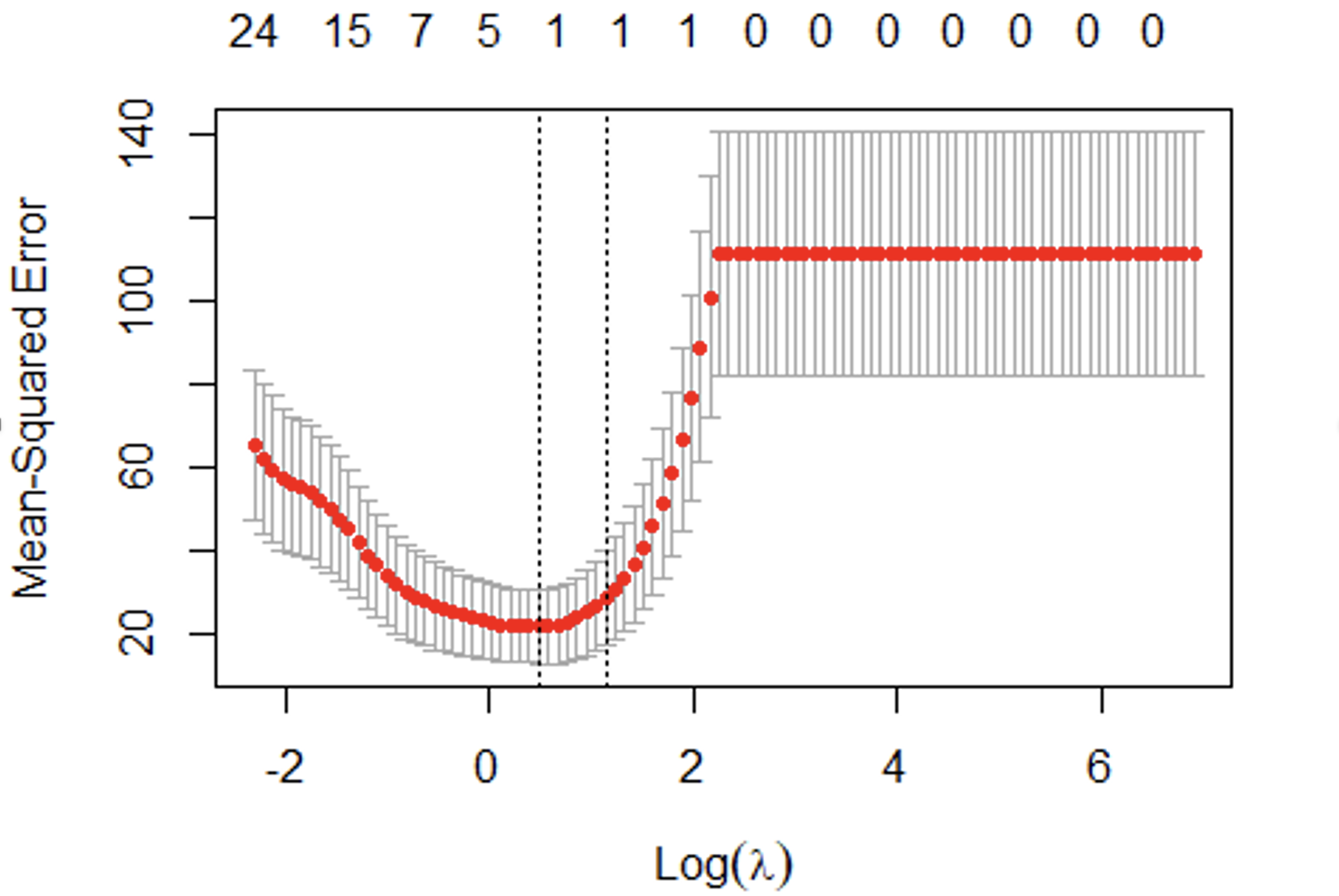
## (b)

#create a grid   
grid = 10^seq(3, -1, length = 100)

# Fit LASSO regression with cross-validation  
lasso\_cvfit <- cv.glmnet(x = pk.x\_train, y = pk.y\_train, alpha = 1, nfolds = 30, type.measure = "mse", thresh = 1e-10, lambda = grid)

## Warning: Option grouped=FALSE enforced in cv.glmnet, since < 3 observations per  
## fold

plot(lasso\_cvfit)



coef(lasso\_cvfit)

# Fit the LASSO model using the optimal lambda  
opt\_lambda <- lasso\_cvfit$lambda.min  
# Identify the selected features  
coef.lasso\_cvfit <- coef(lasso\_cvfit, s = "lambda.min")  
selected\_features <- row.names(coef.lasso\_cvfit)[which(coef.lasso\_cvfit != 0)]  
num\_features <- length(selected\_features)

# Calculate the test error  
test\_pred <- predict(lasso\_cvfit, newx=pk.x\_test)  
test\_mse <- mean((pk.y\_test - test\_pred)^2)

cat("Optimal lambda:", opt\_lambda, "\n")

## Optimal lambda: 1.629751

cat("CV MSE:", lasso\_cvfit$cvm[lasso\_cvfit$lambda == opt\_lambda], "\n")

## CV MSE: 22.08775

cat("Test MSE:", test\_mse, "\n")

## Test MSE: 20.66078

**The optimal lambda is 1.629751, and the test MSE is 20.66078.**

## (c)

selected\_features <- row.names(coef.lasso\_cvfit)[which(coef.lasso\_cvfit != 0)]  
num\_features <- length(selected\_features)

cat("Number of selected features:", length(selected\_features[-1]), "\n")

## Number of selected features: 1

cat("Selected features:", selected\_features[-1], "\n")

## Selected features: X97

**Conclusion: Based on the above (seed), there is only 1 feature selected in the model for the UPDRS. it means that only one predictor variable X97 was selected by the Lasso regression model as being important for predicting the response variable UPDRS. This could be since the other predictor variables have low correlation with the response variable or that they are highly correlated with other predictor variables in the model. As stated in the question, feature X97 is already known to be informative for UPDRS.**

**Having only one predictor variable in the model could be desirable in some situations, as it can simplify the interpretation of the model and reduce the risk of overfitting. However, it is worth exploring other modeling techniques to identify a set of predictor variables that are most important for predicting the response variable.**

## (d)

set.seed(20) # Set the RNG seed for reproducibility  
train\_idx <- sample(1:nrow(parkinsons), 30, replace = FALSE)  
  
# Standardize the predictor variables  
parkinsons\_x <- scale(parkinsons\_x)  
  
pk.x\_train2 <- as.matrix(parkinsons\_x[train\_idx, ])  
pk.y\_train2 <- parkinsons\_y[train\_idx]  
  
# Fit LASSO regression with cross-validation  
lasso\_cvfit2 <- cv.glmnet(pk.x\_train2, pk.y\_train2, alpha = 1, nfolds = 30, type.measure = "mse",   
 standardize = FALSE, intercept = FALSE, thresh = 1e-10,   
 keep = TRUE, parallel = FALSE,   
 lambda = grid)

## Warning: Option grouped=FALSE enforced in cv.glmnet, since < 3 observations per  
## fold

# Identify the selected features  
lasso\_cvfit\_coef <- coef(lasso\_cvfit2, s = "lambda.min")  
selected\_features2 <- row.names(lasso\_cvfit\_coef)[which(lasso\_cvfit\_coef != 0)]

# Print the optimal lambda and test error  
cat("Optimal lambda:", lasso\_cvfit2$lambda.min, "\n")

## Optimal lambda: 2.848036

cat("CV MSE:", lasso\_cvfit2$cvm[lasso\_cvfit2$lambda == lasso\_cvfit2$lambda.min],"\n")

## CV MSE: 694.4499

cat("Number of selected features:", length(selected\_features2), "\n")

## Number of selected features: 6

cat("Selected features:", selected\_features2, "\n")

## Selected features: X44 X50 X62 X89 X90 X97

**Due to the different random split, the features selection is also different. The split can affect the distribution of the data between the training and test datasets, and therefore affect the model’s ability to learn from the data.**

# Q3

insurance <- read.csv("insurance.csv")  
names(insurance)

## [1] "age" "sex" "bmi" "children" "smoker" "region" "charges"

sum(is.na(insurance))

## [1] 0

str(insurance)

## 'data.frame': 1338 obs. of 7 variables:  
## $ age : int 19 18 28 33 32 31 46 37 37 60 ...  
## $ sex : chr "female" "male" "male" "male" ...  
## $ bmi : num 27.9 33.8 33 22.7 28.9 ...  
## $ children: int 0 1 3 0 0 0 1 3 2 0 ...  
## $ smoker : chr "yes" "no" "no" "no" ...  
## $ region : chr "southwest" "southeast" "southeast" "northwest" ...  
## $ charges : num 16885 1726 4449 21984 3867 ...

#Convert the categorical variables (sex, smoker, region) into factors.  
insurance$sex <- factor(insurance$sex)  
insurance$smoker <- factor(insurance$smoker)  
insurance$region <- factor(insurance$region)  
str(insurance)

## 'data.frame': 1338 obs. of 7 variables:  
## $ age : int 19 18 28 33 32 31 46 37 37 60 ...  
## $ sex : Factor w/ 2 levels "female","male": 1 2 2 2 2 1 1 1 2 1 ...  
## $ bmi : num 27.9 33.8 33 22.7 28.9 ...  
## $ children: int 0 1 3 0 0 0 1 3 2 0 ...  
## $ smoker : Factor w/ 2 levels "no","yes": 2 1 1 1 1 1 1 1 1 1 ...  
## $ region : Factor w/ 4 levels "northeast","northwest",..: 4 3 3 2 2 3 3 2 1 2 ...  
## $ charges : num 16885 1726 4449 21984 3867 ...

set.seed(2023) # set the seed for reproducibility  
# Sample the dataset. Returns a list of row indices. 80:20 split.  
row.index <- sample(1:nrow(insurance), 0.8\*nrow(insurance))  
# create the train and test datasets.  
insurance\_train <- insurance[row.index,]  
insurance\_test <- insurance[-row.index,]  
names(insurance\_train)

## [1] "age" "sex" "bmi" "children" "smoker" "region" "charges"

dim(insurance\_train)

## [1] 1070 7

dim(insurance\_test)

## [1] 268 7

# Split the dataset into predictors and response variable  
insurance\_y\_train <- insurance\_train$charges  
insurance\_x\_train <- model.matrix(charges~.,data=insurance\_train)[,-1]  
  
insurance\_y\_test <- insurance\_test$charges  
insurance\_x\_test <- model.matrix(charges~.,data=insurance\_test)[,-1]  
  
# Define the range of alpha values to test  
alpha\_range <- seq(0, 1, by = 0.001)  
  
# Create an empty vector to store the cross-validation error for each alpha value  
elentcv\_error <- rep(NA, length(alpha\_range))  
  
# Loop over the alpha values and calculate the cross-validation error for each one  
for (i in 1:length(alpha\_range)) {  
 elent.cvfit <- cv.glmnet(insurance\_x\_train, insurance\_y\_train, alpha = alpha\_range[i], nfolds = 10)  
 elentcv\_error[i] <- elent.cvfit$cvm[elent.cvfit$lambda == elent.cvfit$lambda.min]  
}  
  
# Find the alpha value with the lowest cross-validation error  
best\_alpha <- alpha\_range[which.min(elentcv\_error)]  
opt\_lambda\_min <- elent.cvfit$lambda.min  
opt\_lambda\_1se <- elent.cvfit$lambda.1se

Get the optimal alpha, and two options of lambda\_min and lambda\_1se:

best\_alpha

## [1] 0.816

opt\_lambda\_min

## [1] 70.22765

opt\_lambda\_1se

## [1] 865.7993

# Train an ElasticNet model with lambda.min  
elent.fit <- glmnet(insurance\_x\_train, insurance\_y\_train, alpha = best\_alpha, lambda = opt\_lambda\_min)  
elent.pred <- predict(elent.fit, newx = insurance\_x\_test)  
elent.mse <- mean((insurance\_y\_test - elent.pred)^2)  
elent.rmse <- sqrt(mean((insurance\_y\_test - elent.pred)^2))

## ElasticNet Model

# Train an ElasticNet model with lambda.1se  
elent.fit2 <- glmnet(insurance\_x\_train, insurance\_y\_train, alpha = best\_alpha, lambda = opt\_lambda\_1se)  
elent.pred2 <- predict(elent.fit2, newx = insurance\_x\_test)  
elent.mse2 <- mean((insurance\_y\_test - elent.pred2)^2)  
elent.rmse2 <- sqrt(mean((insurance\_y\_test - elent.pred2)^2))

# Identify the selected features  
elent.cvfit.coef <- coef(elent.cvfit, s = "lambda.min")  
elent.cvfit.coef2 <- coef(elent.cvfit, s = "lambda.1se")  
selected\_predictors <- row.names(elent.cvfit.coef)[which(elent.cvfit.coef != 0)]  
selected\_predictors2 <- row.names(elent.cvfit.coef2)[which(elent.cvfit.coef2 != 0)]

cat("The optimal alpha:", best\_alpha, "\n")

## The optimal alpha: 0.816

cat("Lambda.min:", opt\_lambda\_min, "\n")

## Lambda.min: 70.22765

cat("Lambda.1se:", opt\_lambda\_1se, "\n")

## Lambda.1se: 865.7993

cat("Number of selected predictors(lambda.min):", length(selected\_predictors[-1]), "\n")

## Number of selected predictors(lambda.min): 7

cat("Number of selected predictors(lambda.1se):", length(selected\_predictors2[-1]), "\n")

## Number of selected predictors(lambda.1se): 3

cat("Selected predictors(lambda.min):", selected\_predictors[-1], "\n")

## Selected predictors(lambda.min): age sexmale bmi children smokeryes regionsoutheast regionsouthwest

cat("Selected predictors(lambda.1se):", selected\_predictors2[-1], "\n")

## Selected predictors(lambda.1se): age bmi smokeryes

cat("\n")

cat("Test MSE (lambda.min):", elent.mse, "\n")

## Test MSE (lambda.min): 36619051

cat("Test MSE (lambda.1se):", elent.mse2, "\n")

## Test MSE (lambda.1se): 37823721

cat("Test RMSE (lambda.min):", elent.rmse, "\n")

## Test RMSE (lambda.min): 6051.368

cat("Test RMSE (lambda.1se):", elent.rmse2, "\n")

## Test RMSE (lambda.1se): 6150.099

**By using 10-fold CV, the optimized value for alpha is 0.816. Lambda.min and 1se have been obtained. The test MSE and RMSE have been calculated:**

* 1. **By using Lambda.min 70.23, the test MSE is 36619051, and the Test RMSE is 6051.368.**
  2. **By using Lambda.1se 865.7993, the test MSE is 37823721, and the Test RMSE is 6150.099.**

**For the ElasticNet Model, lambda.min would be chosen, because it has lower test MSE and RMSE compared with lambda.1se.**

## Ridge and LASSO Regression Models

# Fit LASSO regression with cross-validation  
lasso.cvfit <- cv.glmnet(insurance\_x\_train, insurance\_y\_train, alpha = 1, nfolds = 10, type.measure = "mse")  
ridge.cvfit <- cv.glmnet(insurance\_x\_train, insurance\_y\_train, alpha = 0, nfolds = 10, type.measure = "mse")  
opt\_lambda\_ridge <- ridge.cvfit$lambda.min  
opt\_lambda\_lasso <- lasso.cvfit$lambda.min

lasso.insurance <- glmnet(insurance\_x\_train, insurance\_y\_train, alpha = 1, lambda = opt\_lambda\_lasso)  
ridge.insurance <- glmnet(insurance\_x\_train, insurance\_y\_train, alpha = 0, lambda = opt\_lambda\_ridge)  
lasso.pred <- predict(lasso.insurance, newx = insurance\_x\_test)  
lasso.mse <- mean((insurance\_y\_test - lasso.pred)^2)  
ridge.pred <- predict(ridge.insurance, newx = insurance\_x\_test)  
ridge.mse <- mean((insurance\_y\_test - ridge.pred)^2)  
lasso.rmse <- sqrt(mean((insurance\_y\_test - lasso.pred)^2))  
ridge.rmse <- sqrt(mean((insurance\_y\_test - ridge.pred)^2))

# Identify the selected features  
lasso.cvfit.coef <- coef(lasso.cvfit, s = "lambda.min")  
ridge.cvfit.coef <- coef(ridge.cvfit, s = "lambda.min")  
selected\_lasso <- row.names(lasso.cvfit.coef)[which(lasso.cvfit.coef != 0)]  
selected\_ridge <- row.names(ridge.cvfit.coef)[which(ridge.cvfit.coef != 0)]

cat("LASSO Optimal Lambda:", opt\_lambda\_lasso, "\n")

## LASSO Optimal Lambda: 58.30423

cat("Ridge Optimal Lambda:", opt\_lambda\_ridge, "\n")

## Ridge Optimal Lambda: 972.5731

cat("Number of selected predictors(LASSO):", length(selected\_lasso[-1]), "\n")

## Number of selected predictors(LASSO): 7

cat("Number of selected predictors(Ridge):", length(selected\_ridge[-1]), "\n")

## Number of selected predictors(Ridge): 8

cat("Selected predictors(LASSO):", selected\_lasso[-1], "\n")

## Selected predictors(LASSO): age sexmale bmi children smokeryes regionsoutheast regionsouthwest

cat("Selected predictors(Ridge):", selected\_ridge[-1], "\n")

## Selected predictors(Ridge): age sexmale bmi children smokeryes regionnorthwest regionsoutheast regionsouthwest

cat("\n")

cat("Test MSE (LASSO):", lasso.mse, "\n")

## Test MSE (LASSO): 36619334

cat("Test MSE (Ridge):", ridge.mse, "\n")

## Test MSE (Ridge): 36981364

cat("Test RMSE (LASSO):", lasso.rmse, "\n")

## Test RMSE (LASSO): 6051.391

cat("Test RMSE (Ridge):", ridge.rmse, "\n")

## Test RMSE (Ridge): 6081.23

**For Ridge,**

**Test MSE: 36981364**

**Test RMSE: 6081.23**

**Number of selected predictors: 8**

**For LASSO,**

**Test MSE: 36619334**

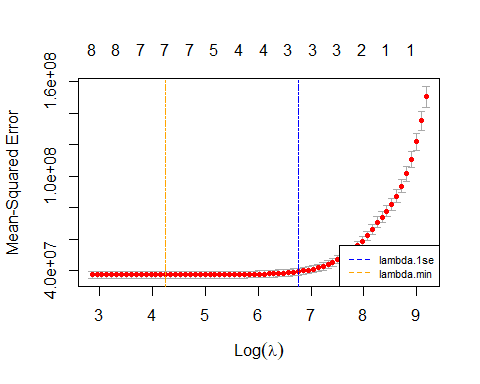
**Test RMSE: 6051. 391**

**Number of selected predictors: 7**

## Plot the CV results for each model

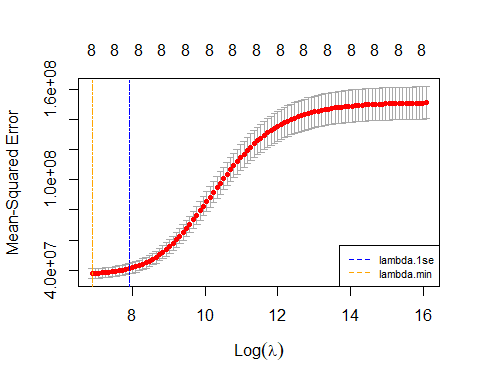
### Plot for ElasticNet

plot(elent.cvfit)  
abline(v = log(elent.cvfit$lambda.min), col = "orange", lty = 6)  
abline(v = log(elent.cvfit$lambda.1se), col = "blue", lty = 6)  
legend("bottomright", legend = c("lambda.1se", "lambda.min"),  
 col = c("blue", "orange"), lty = c(2, 2), cex = 0.7)



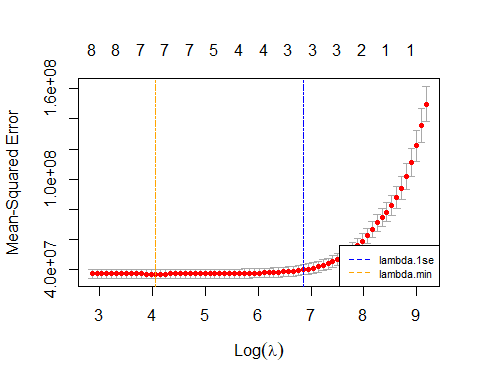
**Plot for Ridge**

plot(ridge.cvfit)  
abline(v = log(ridge.cvfit$lambda.min), col = "orange", lty = 6)  
abline(v = log(ridge.cvfit$lambda.1se), col = "blue", lty = 6)  
legend("bottomright", legend = c("lambda.1se", "lambda.min"),  
 col = c("blue", "orange"), lty = c(2, 2), cex = 0.7)



**Plot for LASSO**

plot(lasso.cvfit)  
abline(v = log(lasso.cvfit$lambda.min), col = "orange", lty = 6)  
abline(v = log(lasso.cvfit$lambda.1se), col = "blue", lty = 6)  
legend("bottomright", legend = c("lambda.1se", "lambda.min"),  
 col = c("blue", "orange"), lty = c(2, 2), cex = 0.7)



**The best performed model is LASSO, the reasons are:**

* 1. **By comparing LASSO and Ridge, LASSO has smaller MSE and RMSE.**
  2. **As for ElasticNet, it generates the similar range of MSE and RMSE as LASSO. However, it is more expansive than LASSO, because it combines L1 and L2 regularization, while LASSO only uses L1 regularization.**