```
library(mice)
setwd("C:\\Users\\yiyuh\\Documents\\College\\Fall 2018\\Stat 517 - Machine Learning\\Project
data=read.csv("salary_uk.csv")
table(data$Category)
```

```
##
##        Accounting & Finance Jobs                         Admin Jobs
##                              606                               151
##         Charity & Voluntary Jobs                  Consultancy Jobs
##                               23                                80
##           Creative & Design Jobs            Customer Services Jobs
##                               22                               257
##    Domestic help & Cleaning Jobs            Energy, Oil & Gas Jobs
##                               10                                31
##                 Engineering Jobs                       Graduate Jobs
##                             1152                                19
##        Healthcare & Nursing Jobs       Hospitality & Catering Jobs
##                             3149                               525
##            HR & Recruitment Jobs                           IT Jobs
##                              578                              1414
##                       Legal Jobs        Logistics & Warehouse Jobs
##                               88                               110
##                 Maintenance Jobs                Manufacturing Jobs
##                               20                               106
##               Other/General Jobs PR, Advertising & Marketing Jobs
##                              236                                88
##                    Property Jobs                        Retail Jobs
##                               44                                93
##                       Sales Jobs              Scientific & QA Jobs
##                              426                               129
##                 Social work Jobs                     Teaching Jobs
##                               53                               342
##        Trade & Construction Jobs                       Travel Jobs
##                              148                               100
```

```
str(data)
```

```
## 'data.frame':    10000 obs. of  12 variables:
##  $ Id                : int  12612628 12612830 12612844 12613049 12613647 13179816 1413133
##  $ Title             : Factor w/ 8274 levels "","'SAP PORTAL ARCHITECT'",..: 2297 7464 41
##  $ FullDescription   : Factor w/ 9940 levels " Call Centre/Telesales advisorBased on the
##  $ LocationRaw       : Factor w/ 2399 levels "Abbots Langley",..: 695 895 968 2061 2061 6
##  $ LocationNormalized: Factor w/ 899 levels "Abbots Langley",..: 251 330 353 761 761 251
##  $ ContractType      : Factor w/ 3 levels "","full_time",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ ContractTime      : Factor w/ 3 levels "","contract",..: 3 3 3 3 3 3 3 3 3 3 ...
```

```
##  $ Company           : Factor w/ 1198 levels "","1 1 Recruitment Limited",..: 478 478 478
##  $ Category          : Factor w/ 28 levels "Accounting & Finance Jobs",..: 9 9 9 9 9 9 9
##  $ SalaryRaw         : Factor w/ 5780 levels " 20-30K"," 20 - 25k Basic,  70 - 75K OTE++
##  $ SalaryNormalized  : int  25000 30000 30000 27500 25000 25000 75000 22000 23000 85000
##  $ SourceName        : Factor w/ 74 levels "accountancyagejobs.com",..: 9 9 9 9 9 9 9 9 9
```

```
data$Title<- as.factor(data$Title)
data$FullDescription<- as.factor(data$FullDescription)
data$ContractType[data$ContractType=='']<-NA
data$ContractType <- as.factor(data$ContractType)
data$ContractTime[data$ContractTime=='']<-NA
data$ContractTime <- as.factor(data$ContractTime)
data$Category <- as.factor(data$Category)
data$SourceName <- as.factor(data$SourceName)
data$Company <- as.factor(data$Company)
data$LocationNormalized <- as.factor(data$LocationNormalized)
data<-subset(data,select = -c(SalaryRaw))


#Cleaning the data

data$Tlevel<-"Mid-Level"
for(i in 1:length(data$Title)){
if(grepl('Director', data[i,3],ignore.case=TRUE)|
 grepl('Senior', data[i,2], ignore.case = TRUE)|
 grepl('Chef',data[i,2] , ignore.case = TRUE) |
 grepl('Lead',data[i,2] , ignore.case = TRUE)){
data$Tlevel [i]<- "Senior"
} else if (grepl("data$Junior",data[i,2] ,ignore.case = TRUE) | grepl("Entry",data[i,2] , ig
data$Tlevel[i]<- "Junior"
} else {
data$Tlevel[i]<- "Mid-Level"
}
}


# Aggregate company variable
company.counts <- summary(data$Company)
top.company <- names(company.counts[order(company.counts, decreasing= TRUE)][1:50])
data$TopCom <- factor(data$Company, levels=top.company)
data$TopCom[data$TopCom == ""] <-NA
data$TopCom <- as.factor(ifelse(is.na(data$TopCom), 0, 1))

# White Collar jobs are 1, else 0
data$WhiteCollar <- grepl('IT', data$Category) | grepl('Engineer', data$Category) |
grepl('Finance', data$Category) | grepl('Legal', data$Category) | grepl('Consult', data$Cate
grepl('HR', data$Category)
```

```
data$WhiteCollar <- as.factor(ifelse(data$WhiteCollar == "TRUE", 1, 0))

#SourceName is separated as top 5 being 1, and else 0
sources.counts <- summary(data$SourceName)
top5.sources <- names(sources.counts[order(sources.counts, decreasing= TRUE)][1:5])
data$Top5Source <- factor(data$Source, levels=top5.sources)
data$Top5Source <- as.factor(ifelse(is.na(data$Top5Source), 0, 1))

#Deleting features
data1<-subset(data,select = -c(Id,Title,FullDescription,LocationRaw,LocationNormalized,
Company,Category,SourceName))


#Train Test data split
set.seed(2344)
n=10000
idx=sample(1:2,n,repl=T)
ss1<-data1[idx==1,]
ss_mod1=mice(ss1[, !names(ss1) %in% "SalaryNormalized"],
method = c("polyreg", "polyreg", "", "" , "", ""))

##
##  iter imp variable
##   1   1  ContractType  ContractTime
##   1   2  ContractType  ContractTime
##   1   3  ContractType  ContractTime
##   1   4  ContractType  ContractTime
##   1   5  ContractType  ContractTime
##   2   1  ContractType  ContractTime
##   2   2  ContractType  ContractTime
##   2   3  ContractType  ContractTime
##   2   4  ContractType  ContractTime
##   2   5  ContractType  ContractTime
##   3   1  ContractType  ContractTime
##   3   2  ContractType  ContractTime
##   3   3  ContractType  ContractTime
##   3   4  ContractType  ContractTime
##   3   5  ContractType  ContractTime
##   4   1  ContractType  ContractTime
##   4   2  ContractType  ContractTime
##   4   3  ContractType  ContractTime
##   4   4  ContractType  ContractTime
##   4   5  ContractType  ContractTime
##   5   1  ContractType  ContractTime
##   5   2  ContractType  ContractTime
##   5   3  ContractType  ContractTime
```

```
##   5   4  ContractType  ContractTime
##   5   5  ContractType  ContractTime

## Warning: Number of logged events: 51

ss11<-cbind(complete(ss_mod1),SalaryNormalized=ss1[,'SalaryNormalized'])
ss2<-data1[idx==2,]
ss_mod2=mice(ss2[, !names(ss2) %in% "SalaryNormalized"],
method = c("polyreg", "polyreg", "", "" , "", ""))

##
##  iter imp variable
##   1   1  ContractType  ContractTime
##   1   2  ContractType  ContractTime
##   1   3  ContractType  ContractTime
##   1   4  ContractType  ContractTime
##   1   5  ContractType  ContractTime
##   2   1  ContractType  ContractTime
##   2   2  ContractType  ContractTime
##   2   3  ContractType  ContractTime
##   2   4  ContractType  ContractTime
##   2   5  ContractType  ContractTime
##   3   1  ContractType  ContractTime
##   3   2  ContractType  ContractTime
##   3   3  ContractType  ContractTime
##   3   4  ContractType  ContractTime
##   3   5  ContractType  ContractTime
##   4   1  ContractType  ContractTime
##   4   2  ContractType  ContractTime
##   4   3  ContractType  ContractTime
##   4   4  ContractType  ContractTime
##   4   5  ContractType  ContractTime
##   5   1  ContractType  ContractTime
##   5   2  ContractType  ContractTime
##   5   3  ContractType  ContractTime
##   5   4  ContractType  ContractTime
##   5   5  ContractType  ContractTime

## Warning: Number of logged events: 51

ss22<-cbind(complete(ss_mod2),SalaryNormalized=ss2[,'SalaryNormalized'])
set.seed(1234)
n=10000
idx2=sample(1:2,n,repl=T)
```

```
data2=rbind(ss11,ss22)
data1.train<-data2[idx2==1,] #training set
data1.test<-data2[idx2==2,] #testing set

####Linear Regression####
data.lm = lm(formula = SalaryNormalized ~ ., data = data1.train)
summary(data.lm)

##
## Call:
## lm(formula = SalaryNormalized ~ ., data = data1.train)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -30537  -9883  -3324   5836 142389
##
## Coefficients:
##                        Estimate Std. Error t value Pr(>|t|)
## (Intercept)             23385.3     4744.1   4.929 8.52e-07 ***
## ContractTypepart_time  -2951.0      705.5  -4.183 2.93e-05 ***
## ContractTimepermanent  -3256.5      644.9  -5.049 4.59e-07 ***
## TlevelMid-Level         9976.7     4689.5   2.127 0.033430 *
## TlevelSenior           14084.3     4717.5   2.986 0.002844 **
## TopCom1                -5762.2      513.3 -11.226  < 2e-16 ***
## WhiteCollar1            9058.8      479.3  18.900  < 2e-16 ***
## Top5Source1            -1781.6      489.2  -3.642 0.000274 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 15520 on 4982 degrees of freedom
## Multiple R-squared:  0.1087, Adjusted R-squared:  0.1075
## F-statistic: 86.81 on 7 and 4982 DF,  p-value: < 2.2e-16

lm_full <- data.lm # full model is the model just fitted
lm_null <- lm(SalaryNormalized ~ 1, data = data1.train)
# backward selection
step(lm_full, trace = F, scope = list(lower=formula(lm_null), upper=formula(lm_full)),
direction = 'backward')

##
## Call:
## lm(formula = SalaryNormalized ~ ContractType + ContractTime +
##     Tlevel + TopCom + WhiteCollar + Top5Source, data = data1.train)
##
## Coefficients:
```

```
##           (Intercept)  ContractTypepart_time  ContractTimepermanent
##                 23385                  -2951                  -3257
##       TlevelMid-Level           TlevelSenior                TopCom1
##                  9977                  14084                  -5762
##          WhiteCollar1            Top5Source1
##                  9059                  -1782
```

```
# forward selection
step(lm_null, trace = F, scope = list(lower=formula(lm_null), upper=formula(lm_full)),
direction = 'forward')
```

```
##
## Call:
## lm(formula = SalaryNormalized ~ WhiteCollar + TopCom + Tlevel +
##     Top5Source + ContractTime + ContractType, data = data1.train)
##
## Coefficients:
##           (Intercept)           WhiteCollar1                TopCom1
##                 23385                   9059                  -5762
##       TlevelMid-Level           TlevelSenior            Top5Source1
##                  9977                  14084                  -1782
## ContractTimepermanent  ContractTypepart_time
##                 -3257                  -2951
```

```
##Predict using the model
lm.pred <- predict(data.lm , newdata = data1.test)
lm.RMSE<-sqrt(mean((lm.pred - data1.test$SalaryNormalized)^2)) #RMSE value, the smaller the
lm.RMSE
```

```
## [1] 15035.21
```

```
###Log transformation###
log.lm <- lm(log(SalaryNormalized) ~., data=data1.train)
summary(log.lm)
```

```
##
## Call:
## lm(formula = log(SalaryNormalized) ~ ., data = data1.train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -1.6533 -0.2937 -0.0154  0.2650  1.9336
##
## Coefficients:
```

```
##                         Estimate Std. Error t value Pr(>|t|)
## (Intercept)              9.89217    0.13075  75.657  < 2e-16 ***
## ContractTypepart_time -0.12837    0.01944  -6.602 4.47e-11 ***
## ContractTimepermanent -0.03987    0.01777  -2.243 0.024934 *
## TlevelMid-Level         0.36253    0.12925   2.805 0.005051 **
## TlevelSenior            0.47000    0.13002   3.615 0.000303 ***
## TopCom1                -0.17500    0.01415 -12.370  < 2e-16 ***
## WhiteCollar1            0.27351    0.01321  20.705  < 2e-16 ***
## Top5Source1            -0.04127    0.01348  -3.061 0.002221 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.4277 on 4982 degrees of freedom
## Multiple R-squared:  0.1289, Adjusted R-squared:  0.1277
## F-statistic: 105.3 on 7 and 4982 DF,  p-value: < 2.2e-16
```

```
log.pred <- predict(log.lm , newdata = data1.test)
log.RMSE<-sqrt(mean((exp(log.pred) - data1.test$SalaryNormalized)^2)) #RMSE value, the small
log.RMSE
```

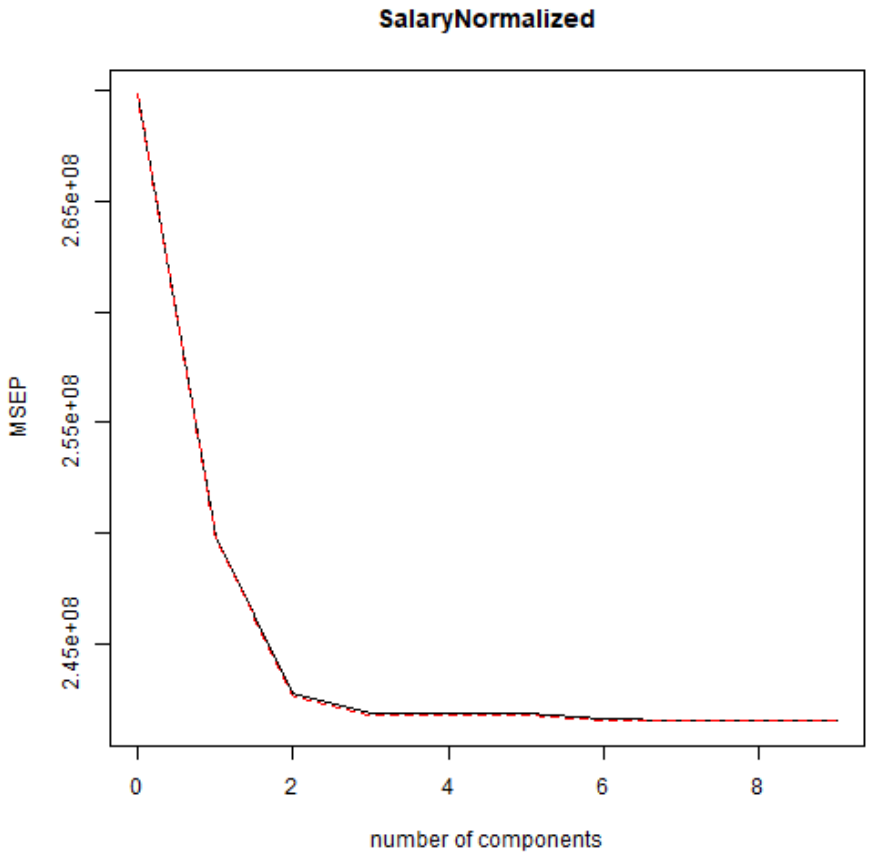
```
## [1] 15289.8
```

```
####Partial Least Squares Regression####
library(pls)
set.seed(1)
pls.fit=plsr(SalaryNormalized~., data=data1.train,scale=TRUE, validation="CV")
summary(pls.fit)
```

```
## Data:    X dimension: 4990 9
##  Y dimension: 4990 1
## Fit method: kernelpls
## Number of components considered: 9
##
## VALIDATION: RMSEP
## Cross-validated using 10 random segments.
##        (Intercept)  1 comps  2 comps  3 comps  4 comps  5 comps  6 comps
## CV           16427    15806    15580    15552    15551    15551    15544
## adjCV        16427    15804    15578    15550    15549    15549    15542
##        7 comps  8 comps  9 comps
## CV       15542    15542    15542
## adjCV    15540    15540    15540
##
## TRAINING: % variance explained
##                   1 comps  2 comps  3 comps  4 comps  5 comps  6 comps
## X                   24.72    42.62    60.65    77.62    90.31    92.58
```

```
## SalaryNormalized     7.82    10.47    10.76    10.76    10.76    10.85
##                   7 comps  8 comps  9 comps
## X                  100.00   100.30   100.60
## SalaryNormalized    10.87    10.87    10.87
```

```
validationplot(pls.fit,val.type="MSEP")
```

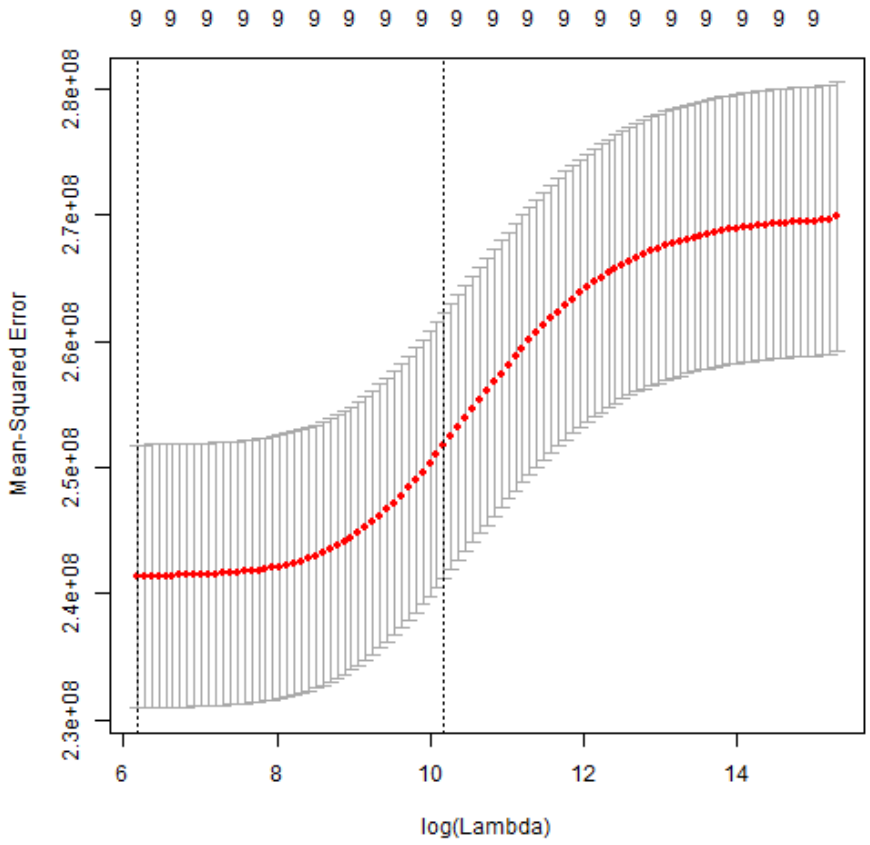



```
pls.pred=predict(pls.fit,x.test,ncomp=7 )
pls.RMSE<-sqrt(mean((pls.pred - y.test)^2))
```

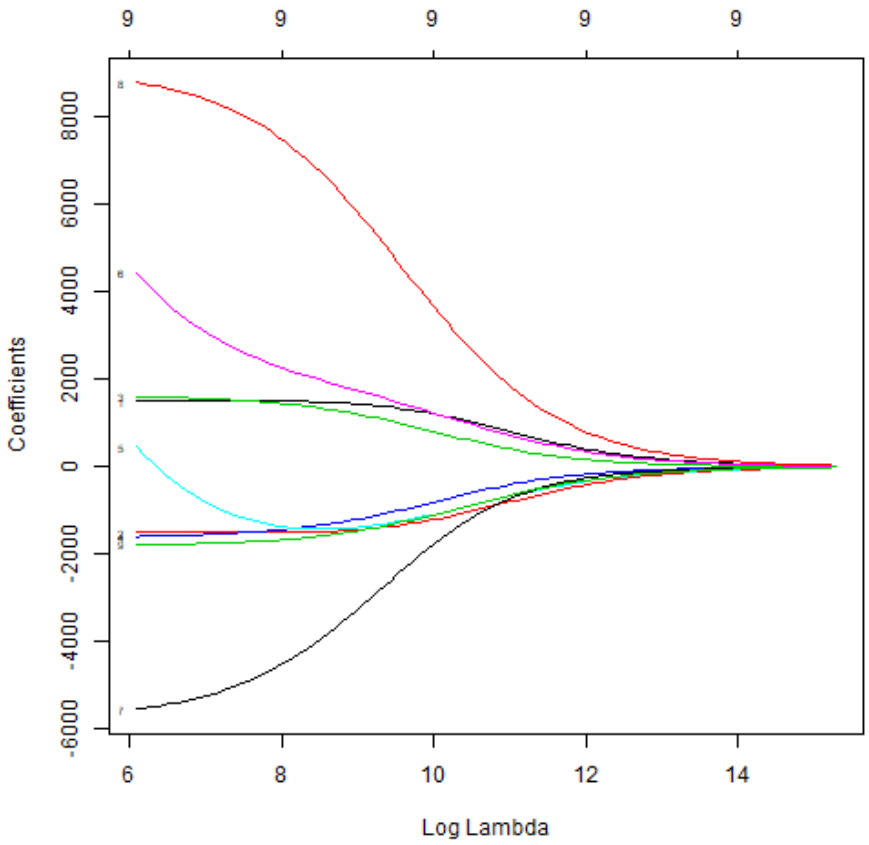
```
####Ridge Regression####
install.packages("glmnet")
```

```
## Error in install.packages : Updating loaded packages
```

```
library("glmnet")
x.train <- model.matrix(SalaryNormalized ~., data = data1.train)[, -1]
y.train <- data1.train$SalaryNormalized
# test set
x.test <- model.matrix(SalaryNormalized ~., data = data1.test)[, -1]
y.test <- data1.test$SalaryNormalized
# obtain best lambda
set.seed(1)
ri.lambda<- cv.glmnet(x.train, y.train, alpha = 0)
plot(ri.lambda)
```




```
# predict test set using best lambda and calculate RMSE
ridge.fit <- glmnet(x.train, y.train, alpha = 0)
plot(ridge.fit, xvar = "lambda", label = TRUE)
```



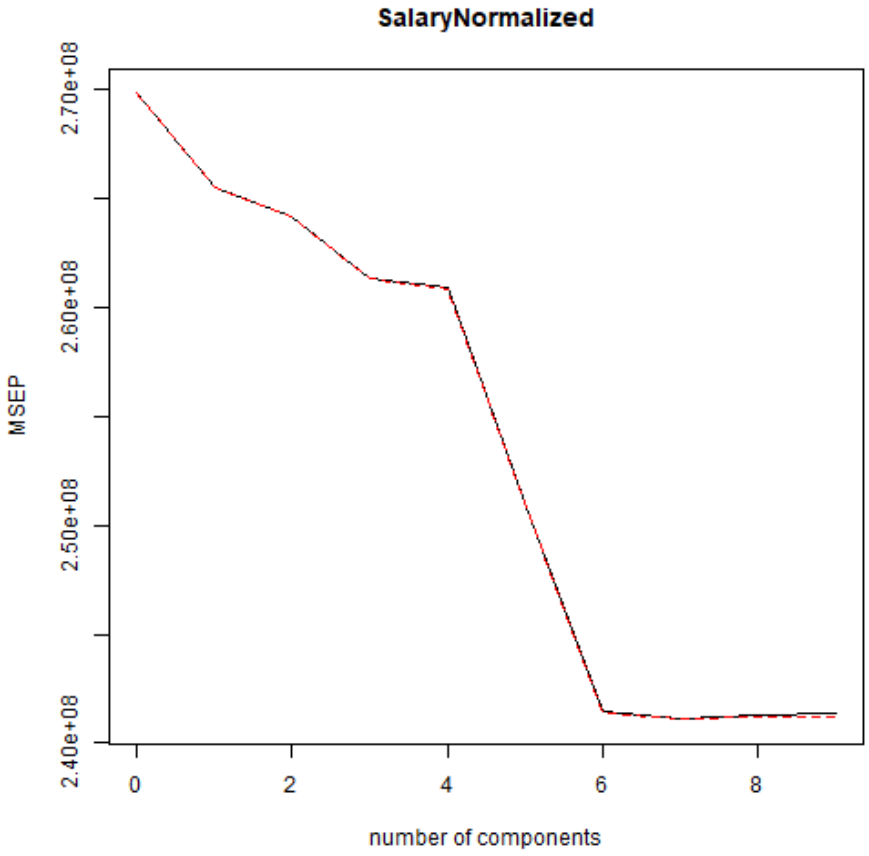

```
ridge.pred <- predict(ridge.fit, s = ri.lambda$lambda.min, newx = x.test)
ridge.RMSE<-sqrt(mean((ridge.pred - y.test)^2))

#### Principal components regression ####
set.seed(2)
pcr.fit=pcr(SalaryNormalized~., data=data1.train,scale=TRUE, validation="CV")
summary(pcr.fit)

## Data:    X dimension: 4990 9
##  Y dimension: 4990 1
## Fit method: svdpc
## Number of components considered: 9
##
## VALIDATION: RMSEP
## Cross-validated using 10 random segments.
```

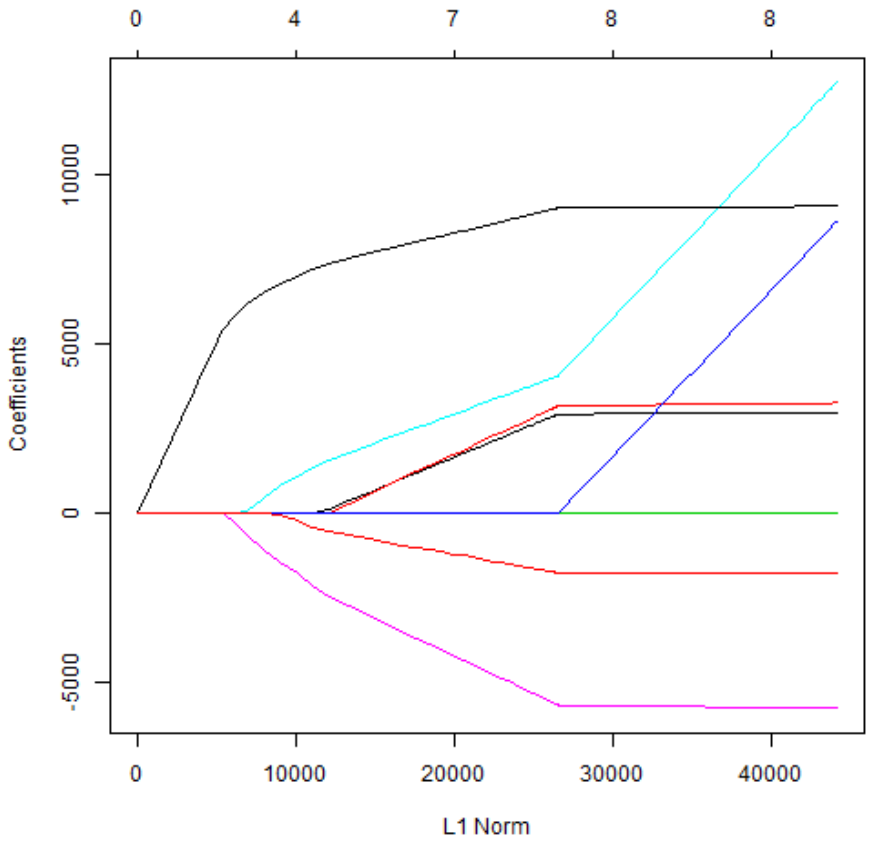
```
##        (Intercept)  1 comps  2 comps  3 comps  4 comps  5 comps  6 comps
## CV           16427    16296    16255    16167    16152    15845    15538
## adjCV        16427    16296    16254    16166    16151    15844    15537
##        7 comps  8 comps  9 comps
## CV       15529    15534    15536
## adjCV    15527    15531    15531
##
## TRAINING: % variance explained
##                   1 comps  2 comps  3 comps  4 comps  5 comps  6 comps
## X                  32.097   52.870   72.061   82.502   91.358    99.92
## SalaryNormalized    1.626    2.232    3.335    3.525    7.211    10.75
##                   7 comps  8 comps  9 comps
## X                  100.00   100.00   100.00
## SalaryNormalized    10.87    10.87    10.89

validationplot(pcr.fit,val.type="MSEP")
```

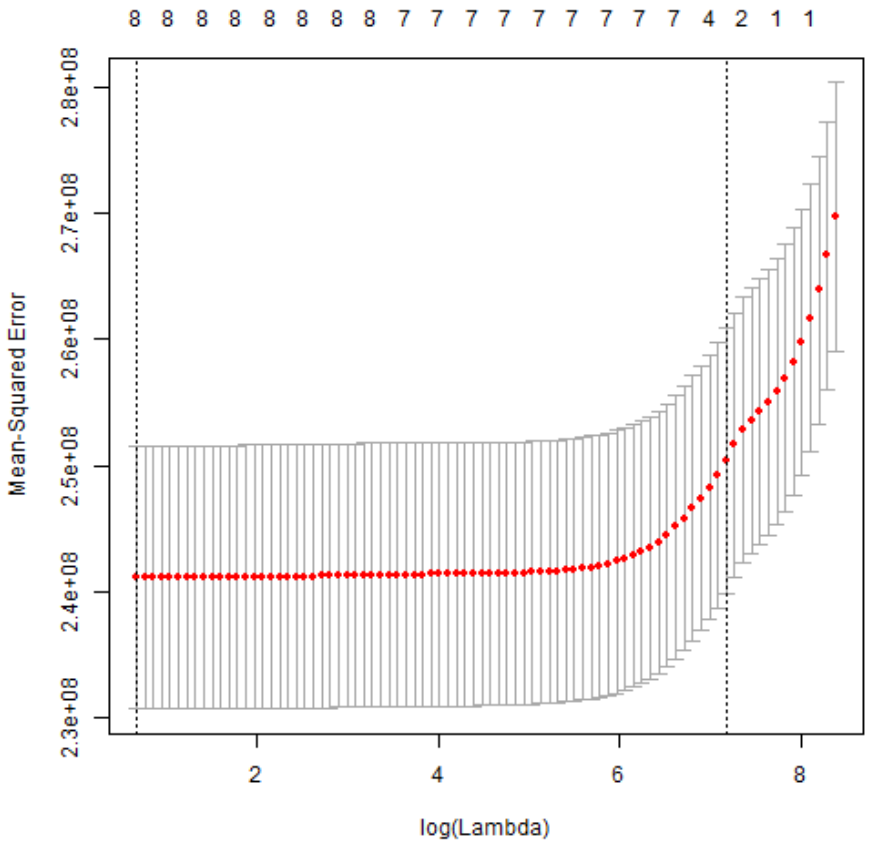


```
pcr.pred=predict(pcr.fit,x.test,ncomp=7)
pcr.RMSE<-sqrt(mean((pcr.pred - y.test)^2))

### Lasso Regression ###
ptm<-proc.time()
set.seed(1)
lasso.fit=glmnet(x.train,y.train,alpha=1)
plot(lasso.fit)
```




```
la.lambda=cv.glmnet(x.train,y.train,alpha=1)
plot(la.lambda)
```




```
# predict test set using best lambda and calculate RMSE
lasso.pred=predict(lasso.fit,s=la.lambda$lambda.min,newx=x.test)
lasso.RMSE<-sqrt(mean((lasso.pred - y.test)^2))

# RMSE summary
RMSE <- rbind(lm.RMSE,log.RMSE,ridge.RMSE,lasso.RMSE,pcr.RMSE,pls.RMSE)
rownames(RMSE) <- (c('Linear Regression', 'Linear Regression(log transform)','Ridge Regressi
'The Lasso','Principal Components Regression','Partial Least Squares'))
colnames(RMSE) <- 'RMSE'
round(RMSE, 4)

##                                     RMSE
## Linear Regression                15035.21
## Linear Regression(log transform) 15289.80
## Ridge Regression                 15031.23
```

```
## The Lasso                          15033.79
## Principal Components Regression  15035.21
## Partial Least Squares             15035.21
```

The Linear Regression model with the log transformed performed marginally better than the other models. The fact that the other models have a very similar score is intriguing.