

Predicting Job Salary

Vignesh J Muralidharan

October 13 2018

Introduction

This task is known as a regression problem, one where the response variable Y is continuous in nature. This problem interestingly only has categorical variables that are difficult to process. The main focus of this part will be in checking different features and methods of extraction to improve results. Two languages were used and contrasted for this task to create a linear model from the features. Two methods were explored: bag of words and aggregation. The data consists of the following information:

Id - A unique identifier for each job ad

Title - A freetext field supplied to us by the job advertiser as the Title of the job ad.

FullDescription - The full text of the job ad as provided by the job advertiser.

LocationRaw - The freetext location as provided by the job advertiser.

LocationNormalized - Adzuna's normalised location from within our own location tree,

ContractType - full_time or part_time, interpreted by Adzuna from description

ContractTime - permanent or contract, interpreted by Adzuna from description

Company - the name of the employer as supplied to us by the job advertiser.

Category - 30 standard job categories this ad fits into, inferred in a very messy way based on the source SalaryRaw - the freetext salary field we received in the job advert from the advertiser.

SalaryNormalised - the annualised salary interpreted by Adzuna from the raw salary.

SourceName - the name of the website or advertiser from whom we received the job advert.

```
library(plyr) ; library(stringr)
library(caret) ; library(car)
library(class) ; library(knitr)
library(MASS) ; library(e1071)
library(glmnet) ; library(pls) ; library(mice)
```

1. Reading data into R and converting some variables into factors

```
sdata=read.csv("https://raw.githubusercontent.com/vigneshjmurali/Statistical-Predictive-Modelling/master/Datasets/Project%201_Dataset_1_salary_uk.csv")
```

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

2. Data Cleaning

Aggregating Titles into three levels: Senior, Mid-Level, Junior

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

Dividing locations into two levels, London label as 1, others as 0

```
myurl<-'https://docs.google.com/spreadsheets/d/e/2PACX-1vQXzU41Zv3GwB5s_YJQsrLdSnMt2isMWj03ZZ910sLe1_vL9ZtsyROewGegGZDkmwgYYa1FMw2tWzKl/pub?gid=1568496122&single=true&output=csv'
tree1 <- read.csv(url(myurl),header = FALSE)
tree<-as.vector(tree1[, 'V1'])
for (i in 1:nrow(sdata)) {
  # get city name
  loc <- sdata$LocationNormalized[i]
  # find the first line in the tree in which that city name appears
  line.id <- which(grepl(loc, tree))[1]
  # use regular expressions to pull out the broad location
  r <- regexpr("~.+?~", tree[line.id])
  match <- regmatches(tree[line.id], r)
  # store the broad location
  sdata$Location[i] <- gsub("~", "", match) #Error: replacement has length zero
}
## Error in sdata$Location[i] <- gsub("~", "", match): replacement has length zero
sdata$Location <- as.factor(sdata$Location)
table(sdata$Location)
##           East Midlands           Eastern England           London
##                349                598                1928
##      North East England      North West England      Northern Ireland
##                214                561                125
##           Scotland           South East England           South West England
##                443                4500                514
##           Wales           West Midlands Yorkshire And The Humber
##                173                366                229
```

```
# Label London as 1, non-London as 0
sdata$Location <- as.factor(ifelse(sdata$Location == "London", 1, 0))
```

Since there are so many different companies. I have no idea how to aggregate them into levels. I just label Top 50 companies as 1, others as 0.

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

Creating an aggregate category: WhiteCollar=(Accounting, Engineering, Legal, IT, Cosultancy,HR)

WhiteCollar labels 1, others label 0

```
sdata$WhiteCollar <- grepl('IT', sdata$Category) | grepl('Engineer', sdata$Category) |
grepl('Finance', sdata$Category) | grepl('Legal', sdata$Category) | grepl('Consult', sdata$Category) |
grepl('HR', sdata$Category)
sdata$WhiteCollar <- as.factor(ifelse(sdata$WhiteCollar == "TRUE", 1, 0))
```

Dividing 'SourceName' into two levels, Top 5 Source lables 1, others label 0

```
sources.counts <- summary(sdata$SourceName)
top5.sources <- names(sources.counts[order(sources.counts, decreasing= TRUE)][1:5])
sdata$Top5Source <- factor(sdata$Source, levels=top5.sources)
sdata$Top5Source <- as.factor(ifelse(is.na(sdata$Top5Source), 0, 1))
```

Dropping previously modified attributes and attributes that will not be used

```
sdata1<-subset(sdata,select = -c(Id,Title,FullDescription,LocationRaw,LocationNormalized,
Company,Category,SourceName))
```

Randomly dividing the clean data set into two sets of labels 1 (training data) and 2(test data). Here I used mice package to impute missing values to variable 'contractType' and 'contractTime' in these two sets seperately. I tried to impute the missing values for the whole data set but failed as the size of the data set is too large.

Baseline

Having a baseline and a method of classifying success is equally as important in a regression model as with classification. In this case, it was decided that the root mean squared error (RMSE) would provide the most meaningful insight into the quality of the model. The works by calculating the difference between the expected outcome and the predicted outcome, squaring it, averaging that quantity and taking the square root.

$$RMSE = \sqrt{\frac{1}{n} \sum_{i=1}^n (\hat{Y}_i - Y_i)^2}$$

```
set.seed(2344) ; n=10000
idx=sample(1:2,n,repl=T)
ss1<-sdata1[idx==1,]
ss_mod1=mice(ss1[, !names(ss1) %in% "SalaryNormalized"],
method = c("polyreg", "polyreg", "", "", "", "", ""))
## Warning: Number of logged events: 51
ss11<-cbind(complete(ss_mod1),SalaryNormalized=ss1[, 'SalaryNormalized'])
ss2<-sdata1[idx==2,]
ss_mod2=mice(ss2[, !names(ss2) %in% "SalaryNormalized"],
method = c("polyreg", "polyreg", "", "", "", "", ""))
## 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)
sdata2=rbind(ss11,ss22)
sdata1.train<-sdata2[idx2==1,] #training set
sdata1.test<-sdata2[idx2==2,] #testing set
```

3.Linear regression

The primary method for developing this model hinges on linear regression and shaping the features such that the linear regression model can best fit them

```
# Load function
sdata.lm = lm(formula = SalaryNormalized ~ ., data = sdata1.train)
summary(sdata.lm)
## Call:
## lm(formula = SalaryNormalized ~ ., data = sdata1.train)
## Residuals:
##      Min       1Q   Median       3Q      Max
## -32921  -9192  -2743   4846 142092
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)    22736.0     2729.8   8.329 < 2e-16 ***
## ContractTypepart_time -2219.5       675.5  -3.286  0.00102 **
## ContractTimepermanent -3969.3       630.3  -6.298 3.28e-10 ***
## TlevelMid-Level      7972.5     2646.3   3.013  0.00260 **
## TlevelSenior     15502.9     2657.0   5.835 5.73e-09 ***
## Location1         3532.0       547.5   6.451 1.22e-10 ***
## TopCom1        -5580.9       495.9 -11.253 < 2e-16 ***
## WhiteCollar1      9872.8       470.1  20.999 < 2e-16 ***
## Top5Source1     -1368.7       476.7  -2.871  0.00411 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## Residual standard error: 15110 on 4981 degrees of freedom
## Multiple R-squared:  0.1554, Adjusted R-squared:  0.154
## F-statistic: 114.5 on 8 and 4981 DF,  p-value: < 2.2e-16
lm_full <- sdata.lm # full model is the model just fitted
lm_null <- lm(SalaryNormalized ~ 1, data = sdata1.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 + Location + TopCom + WhiteCollar + Top5Source, data = sdata1.train)
## Coefficients:
##              (Intercept)  ContractTypepart_time  ContractTimepermanent
##                22736                -2220                -3969
##      TlevelMid-Level      TlevelSenior      Location1
##                7972                15503                3532
##      TopCom1      WhiteCollar1      Top5Source1
##        -5581                9873        -1369
# forward selection
step(lm_null, trace = F, scope = list(lower=formula(lm_null), upper=formula(lm_full)),
     direction = 'forward')
## Call:
## lm(formula = SalaryNormalized ~ WhiteCollar + Tlevel + TopCom +
##      Location + ContractTime + ContractType + Top5Source, data = sdata1.train)
## Coefficients:
##              (Intercept)      WhiteCollar1      TlevelMid-Level
##                22736                9873                7972
##      TlevelSenior      TopCom1      Location1
##                15503        -5581                3532
## ContractTimepermanent  ContractTypepart_time      Top5Source1
##        -3969                -2220        -1369
##Predict using the model
lm.pred <- predict(sdata.lm , newdata = sdata1.test)
lm.RMSE<-sqrt(mean((lm.pred - sdata1.test$SalaryNormalized)^2)) #RMSE value, the smaller the better
```

er

```
lm.RMSE
```

```
## [1] 14644.16
```

Both backward and forward selection shows that no variables was dropped, so I used the full model to make the prediction. The result above shows that RMSE=14644.16.

4. Trying modeling log-transformation of the response: log(SalaryNormalized)

```
log.lm <- lm(log(SalaryNormalized) ~., data=sdata1.train)
```

```
summary(log.lm)
```

```
## Call:
```

```
## lm(formula = log(SalaryNormalized) ~ ., data = sdata1.train)
```

```
## Residuals:
```

```
##      Min       1Q   Median       3Q      Max   
## -1.60888 -0.27385 -0.01005  0.23389  1.89592
```

```
## Coefficients:
```

```
##              Estimate Std. Error t value Pr(>|t|)      
## (Intercept)    9.92264    0.07432  133.518 < 2e-16 ***   
## ContractTypepart_time -0.10134    0.01839   -5.511 3.75e-08 ***   
## ContractTimepermanent -0.06286    0.01716   -3.663 0.000252 ***   
## TlevelMid-Level      0.24133    0.07204    3.350 0.000815 ***   
## TlevelSenior         0.48701    0.07233    6.733 1.85e-11 ***   
## Location1            0.10195    0.01491    6.839 8.93e-12 ***   
## TopCom1              -0.17090    0.01350  -12.657 < 2e-16 ***   
## WhiteCollar1         0.29900    0.01280   23.360 < 2e-16 ***   
## Top5Source1          -0.02449    0.01298   -1.887 0.059239 .
```

```
## ---
```

```
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
##
```

```
## Residual standard error: 0.4113 on 4981 degrees of freedom
```

```
## Multiple R-squared:  0.1945, Adjusted R-squared:  0.1932
```

```
## F-statistic: 150.4 on 8 and 4981 DF,  p-value: < 2.2e-16
```

```
log.pred <- predict(log.lm , newdata = sdata1.test)
```

```
log.RMSE<-sqrt(mean((exp(log.pred) - sdata1.test$SalaryNormalized)^2)) #RMSE value, the smaller t  
he better
```

```
log.RMSE
```

```
## [1] 14857.66
```

After log transformation to the response, the RMSE=14857.66 has increased compared to the model without transformation. This means that log transformation didn't help.

5. Ridge Regression

```
library(glmnet)
```

```
#training set
```

```
x.train <- model.matrix(SalaryNormalized ~., data = sdata1.train)[, -1]
```

```
y.train <- sdata1.train$SalaryNormalized
```

```
# test set
```

```
x.test <- model.matrix(SalaryNormalized ~., data = sdata1.test)[, -1]
```

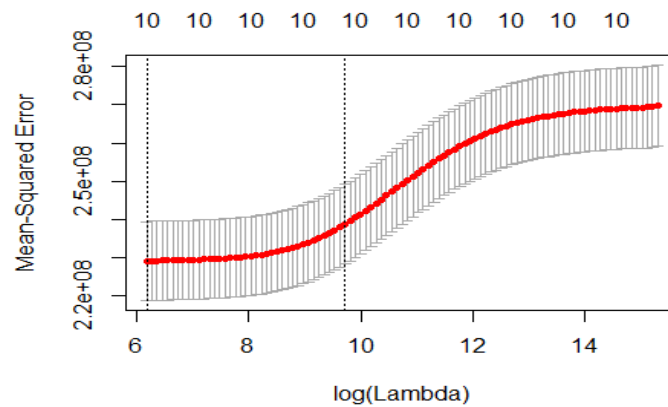
```
y.test <- sdata1.test$SalaryNormalized
```

```
# obtain best lambda
```

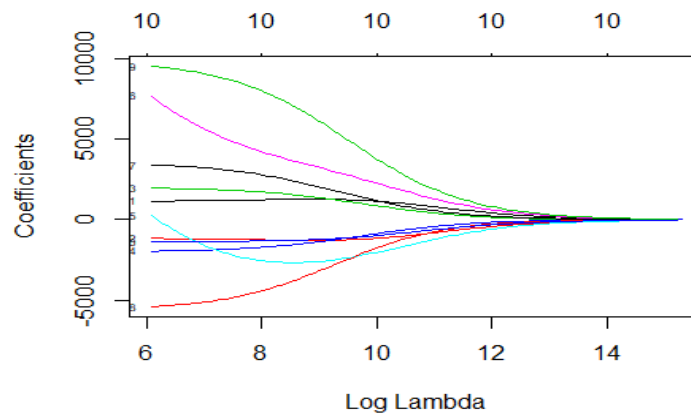
```
set.seed(1)
```

```
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)
```

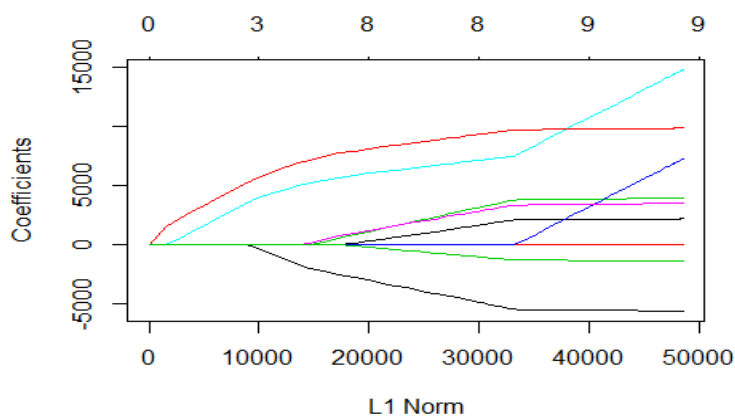


```
ridge.pred <- predict(ridge.fit, s = ri.lambda$lambda.min, newx = x.test)
ridge.RMSE <- sqrt(mean((ridge.pred - y.test)^2))
```

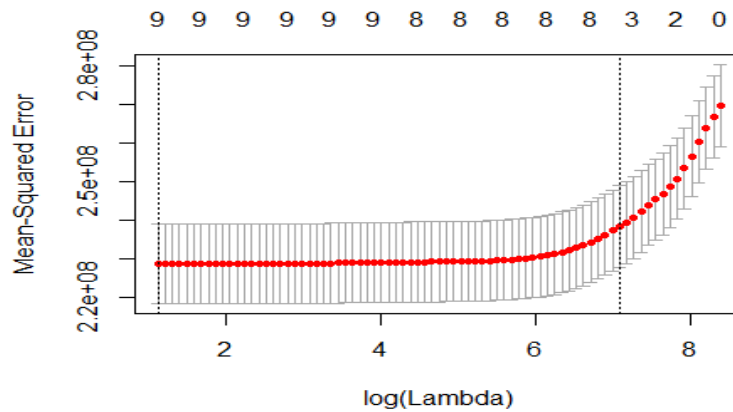
After using Ridge Regression to fit the data, we can see that RMSE= 14643.47 decreased a little bit compared to that of the linear regression.

6.The Lasso Regression

```
set.seed(1) ; lasso.fit=glmnet(x.train,y.train,alpha=1) ; plot(lasso.fit)
```



```
# obtain best Lambda
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))
```

the result above showed us that RMSE= 14642.78 is very close to that of Ridge Regression.

7.Principal Components Regression

```
set.seed(2) ; pcr.fit=pcr(SalaryNormalized~., data=sdata1.train,scale=TRUE, validation="CV")
```

```
summary(pcr.fit)
```

```
## Data:      X dimension: 4990 10
```

```
## Y dimension: 4990 1
```

```
## Fit method: svdpc
```

```
## Number of components considered: 10
```

```
## VALIDATION: RMSEP
```

```
## Cross-validated using 10 random segments.
```

	(Intercept)	1 comps	2 comps	3 comps	4 comps	5 comps	6 comps
## CV	16427	16180	16172	15913	15914	15912	15792
## adjCV	16427	16179	16173	15912	15913	15911	15811

```
##      7 comps  8 comps  9 comps 10 comps
```

```
## CV      15148  15119  15122  15126
```

```
## adjCV   15147  15118  15119  15121
```

```
## TRAINING: % variance explained
```

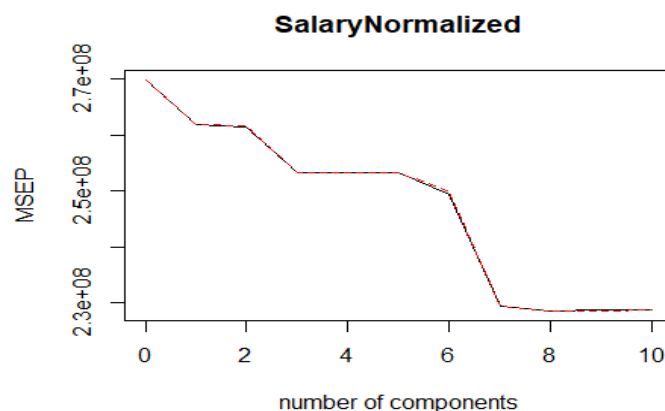
	1 comps	2 comps	3 comps	4 comps	5 comps	6 comps
## X	27.726	46.819	64.776	75.334	84.617	92.347
## SalaryNormalized	3.006	3.171	6.326	6.384	6.423	7.748

```
##      7 comps  8 comps  9 comps 10 comps
```

```
## X      99.86  100.00  100.00  100.00
```

```
## SalaryNormalized  15.20  15.54  15.55  15.55
```

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



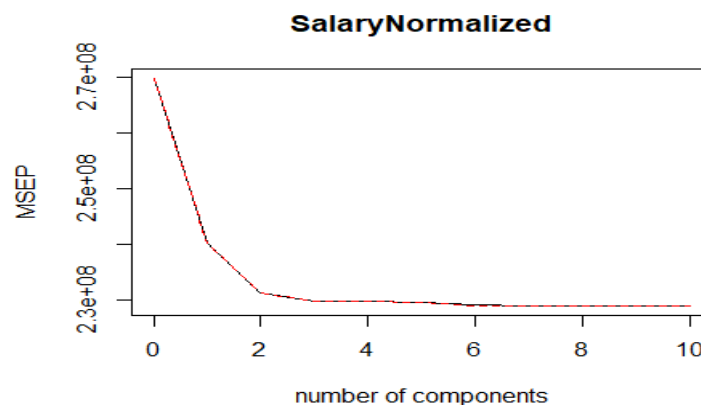
```
set.seed(1) ; pcr.pred=predict(pcr.fit,x.test,ncomp=8) ; pcr.RMSE<-sqrt(mean((pcr.pred - y.test)^2))
```

predict test set using M=8 and calculate RMSE

The lowest crossvalidation error occurs when there are $M = 8$ components; $RMSE=14644.16$, which means Principal Components Regression performed just like linear regression.

8. Partial Least Squares

```
set.seed(1)
pls.fit=plsr(SalaryNormalized~., data=sdata1.train,scale=TRUE, validation="CV") ;summary(pls.fit)
## Data:      X dimension: 4990 10
## Y dimension: 4990 1
## Fit method: kernelpls
## Number of components considered: 10
## VALIDATION: RMSEP
## Cross-validated using 10 random segments.
##      (Intercept) 1 comps 2 comps 3 comps 4 comps 5 comps 6 comps
## CV      16427    15502   15214   15164   15162   15155   15137
## adjCV    16427    15500   15211   15162   15160   15154   15135
##      7 comps 8 comps 9 comps 10 comps
## CV      15137   15137   15137   15137
## adjCV    15134   15134   15134   15134
## TRAINING: % variance explained
##      1 comps 2 comps 3 comps 4 comps 5 comps 6 comps
## X      22.41  37.73  53.29  65.90  78.15  82.77
## SalaryNormalized 11.36  14.73  15.22  15.24  15.33  15.53
##      7 comps 8 comps 9 comps 10 comps
## X      91.09 100.00 101.03 102.06
## SalaryNormalized 15.54 15.54 15.54 15.54
validationplot(pls.fit,val.type="MSEP")
```



#The Lowest cross-validation error occurs when $n = 7$ partial least squares directions are used
`pls.pred=predict(pls.fit,x.test,ncomp=7) ;pls.RMSE<-sqrt(mean((pls.pred - y.test)^2))`

For Partial Least Squares, the lowest cross-validation error occurs when $n = 7$ partial least squares directions are used. $RMSE=14644.56$, which is approximately equal to that of linear regression.

9. Summary

```
# 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 Regression',
                    'The Lasso', 'Principal Components Regression', 'Partial Least Squares'))
colnames(RMSE) <- 'RMSE' ; round(RMSE, 4)
##      RMSE
## Linear Regression      14644.16
## Linear Regression(log transform) 14857.66
## Ridge Regression      14643.47
## The Lasso             14642.78
## Principal Components Regression 14644.16
## Partial Least Squares 14644.56
```

From the output above, we can see that of all the methods that I used, Linear Regression with log transformation performed the worse, while The Lasso performed the best of all. And all the RMSE are pretty close. I couldn't find easy to do the random forest regression in the R I am getting error in that