finalproject

Rui Chen

November 28, 2016

SUMMARY 1. multiple linear regression

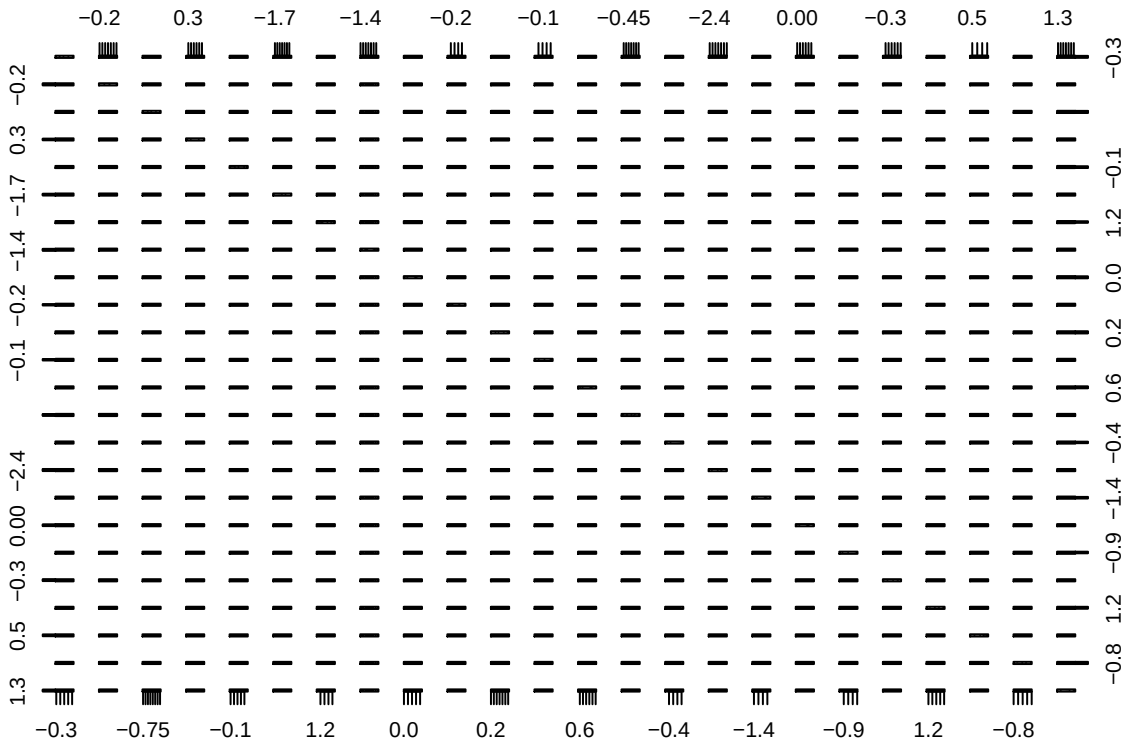
calculate VIF (non linear regression???)

2.All subset selection 3.Ridge regression 4.Lasso regression 5.PCR 6.GAM

```
kidney = read.csv("Kidney_2.csv", header = TRUE)
kidney = t(kidney)
colnames(kidney) = kidney[1,]
kidney= kidney[-1,]
kidney = as.data.frame(kidney)

# change from factor to numeric
for(i in 1:dim(kidney)[2]){
  kidney[,i] = as.numeric(levels(kidney[,i])[kidney[,i]])
}

pairs(kidney)
```



Split training and testing data

```
set.seed(1)
train = sample(1:40, 30)
test = (-train)
```

Linear regression:

```
library(car)
fit.lm = lm(Mapk1~., data = kidney[train,])
pred.lm = predict(fit.lm, newdata = kidney[test,])
mean((pred.lm-kidney[test,]$Mapk1)^2)
```

```
## [1] 0.09013328
```

we see the summary of the fit.lm, the p value of the coefficients are extremely large, thus linear regression including all covariates is not good.

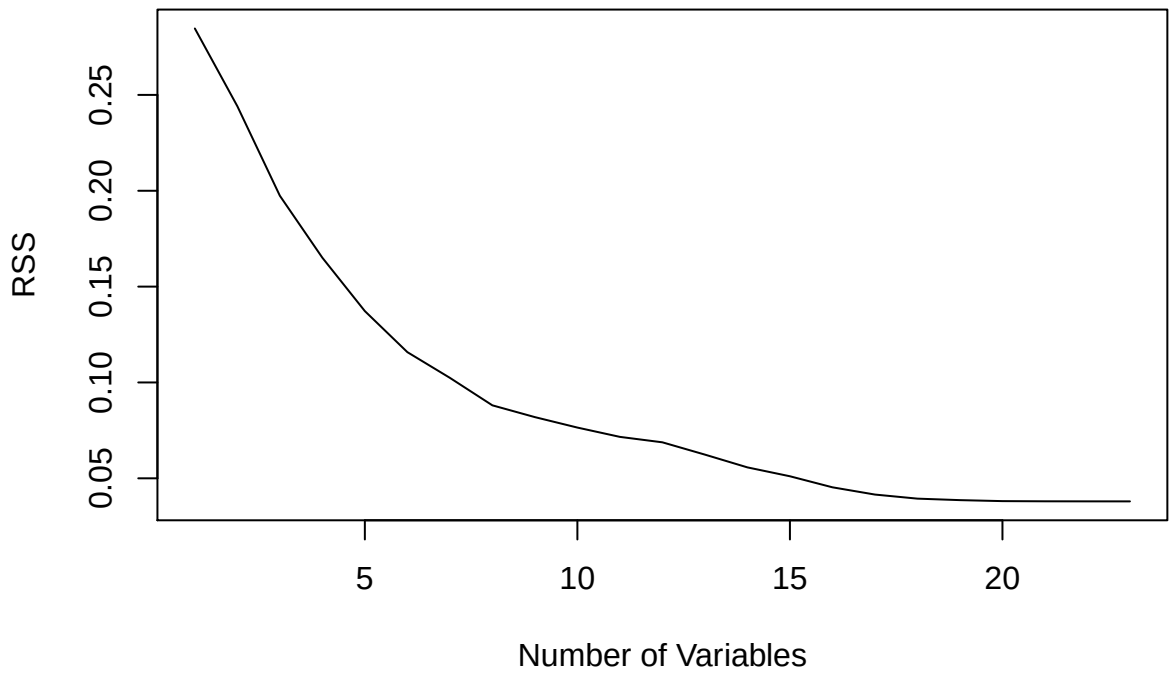
model selection:

All subset:

```
library(leaps)
regfit.full=regsubsets(Mapk1~.,kidney[train,], nvmax = 23)
reg.summary=summary(regfit.full)
reg.summary$rsq
```

```
##  [1] 0.3342386 0.4290990 0.5383187 0.6139098 0.6790434 0.7291189 0.7604858
##  [8] 0.7940114 0.8083868 0.8211605 0.8325085 0.8391377 0.8541462 0.8697444
## [15] 0.8805510 0.8940403 0.9028730 0.9078741 0.9096926 0.9109002 0.9111170
## [22] 0.9111942 0.9112003
```

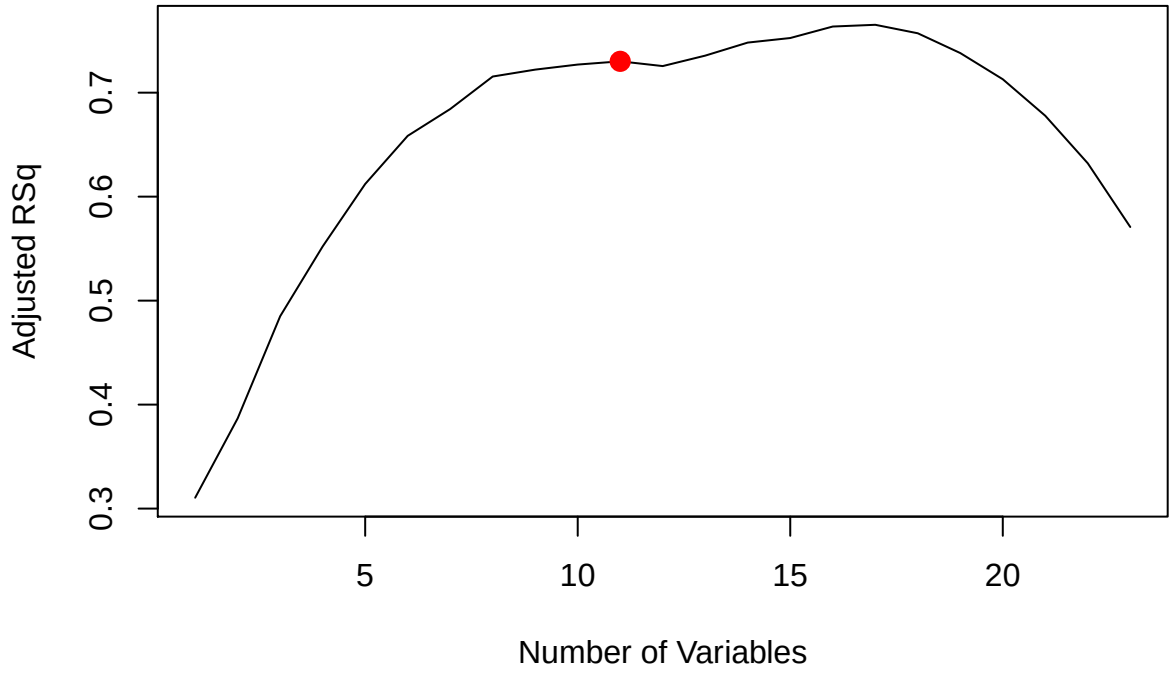
```
plot(reg.summary$rss,xlab="Number of Variables",ylab="RSS",type="l")
```




```
plot(reg.summary$adjr2,xlab="Number of Variables",ylab="Adjusted RSq",type="l")
which.max(reg.summary$adjr2)
```

```
## [1] 17
```

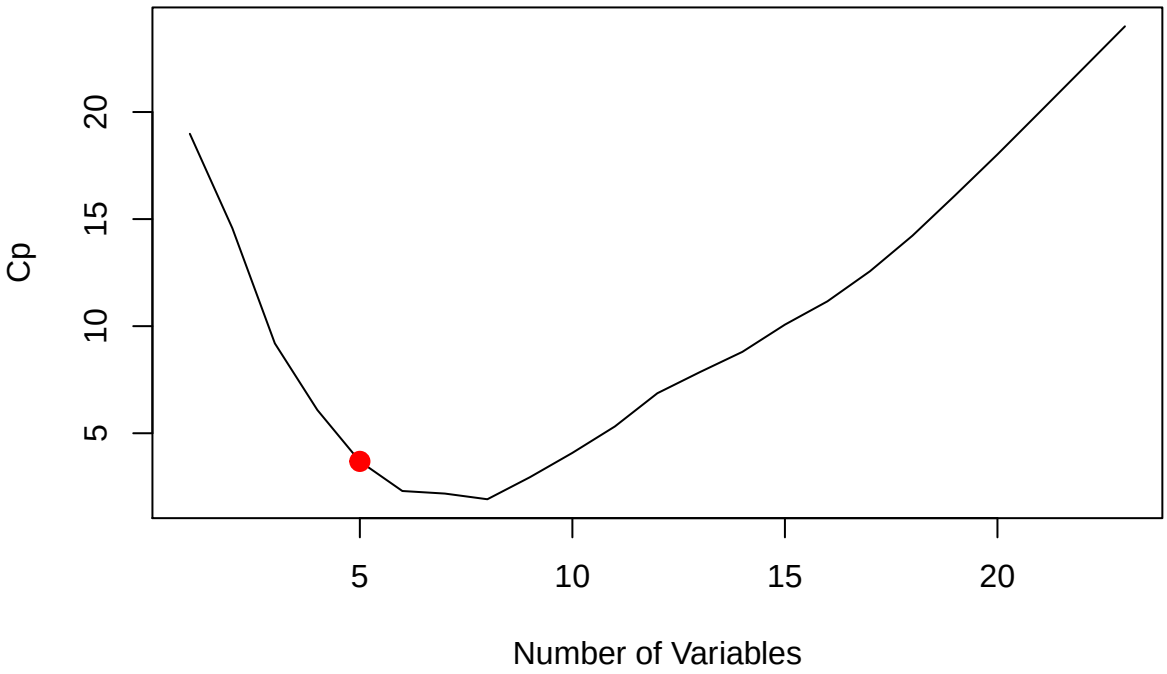
```
points(11,reg.summary$adjr2[11], col="red",cex=2,pch=20)
```




```
plot(reg.summary$cp,xlab="Number of Variables",ylab="Cp",type='l')
which.min(reg.summary$cp)
```

```
## [1] 8
```

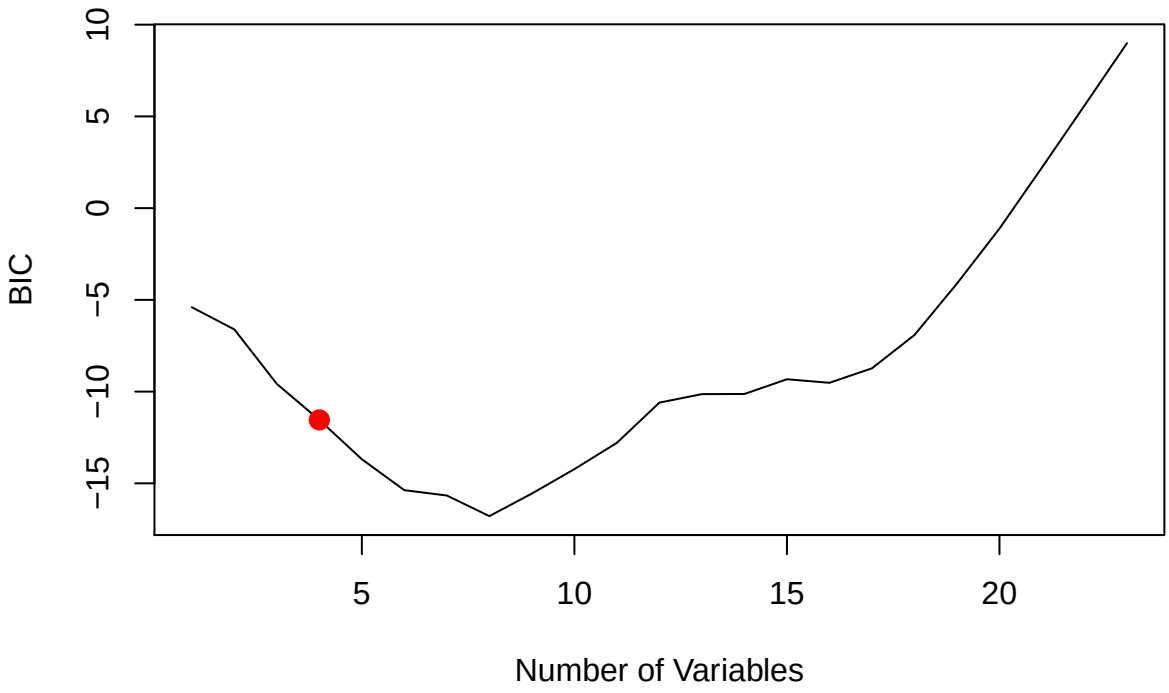
```
points(5,reg.summary$cp[5],col="red",cex=2,pch=20)
```



```
which.min(reg.summary$bic)
```

```
## [1] 8
```

```
plot(reg.summary$bic,xlab="Number of Variables",ylab="BIC",type='l')
points(4,reg.summary$bic[4],col="red",cex=2,pch=20)
```



By criteria of adjusted r-square, we choose model with 11 variables; by criteria of Mallow CP, we choose model with 5 variables, by criteria of BIC, we choose model with 4 variables.

Because the observations and full model variables are relatively small, so I think doing all subset selection is resonable and backward/forward selection is not needed.

12 variables: Cdc42; Pla2g6; Akt2; Plcg2; Rac2;Rik;Pla2g5;Sphk2;Map2k1;Ptk2;Nos3;Rac1;

```
library(boot)
```

```
##
## Attaching package: 'boot'
```

```
## The following object is masked from 'package:car':
##
##     logit
```

```
fit.lm12=glm(Mapk1~Cdc42+Pla2g6+Akt2+Plcg2+Rac2+Rik+Pla2g5+Sphk2+Map2k1+Ptk2+Nos3+Rac1, data = kidney)
summary(fit.lm12)
```

```
##
## Call:
## glm(formula = Mapk1 ~ Cdc42 + Pla2g6 + Akt2 + Plcg2 + Rac2 +
```

```
##     Rik + Pla2g5 + Sphk2 + Map2k1 + Ptk2 + Nos3 + Rac1, data = kidney)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -0.15115  -0.04645   0.01172   0.04583   0.11955
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.19801    0.28063   0.706  0.48647
## Cdc42        0.38193    0.18538   2.060  0.04914 *
## Pla2g6      -0.16999    0.08384  -2.028  0.05258 .
## Akt2        -0.31975    0.16327  -1.958  0.06059 .
## Plcg2        0.28072    0.12162   2.308  0.02889 *
## Rac2        -0.15978    0.08113  -1.969  0.05925 .
## Rik          0.12778    0.09560   1.337  0.19250
## Pla2g5      -0.13544    0.08890  -1.523  0.13927
## Sphk2       -0.13278    0.07708  -1.723  0.09639 .
## Map2k1       0.21906    0.14863   1.474  0.15209
## Ptk2         0.30159    0.23734   1.271  0.21468
## Nos3         0.19545    0.07343   2.662  0.01293 *
## Rac1         0.39902    0.11674   3.418  0.00201 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for gaussian family taken to be 0.005602636)
##
##     Null deviance: 0.61566  on 39  degrees of freedom
## Residual deviance: 0.15127  on 27  degrees of freedom
## AIC: -81.587
##
## Number of Fisher Scoring iterations: 2
```

```
cv.out = cv.glm(kidney, fit.lm12)
cv.out$delta[1]
```

```
## [1] 0.008570011
```

4 variables: Akt2; Rik; Pik3r3;Rac1

```
fit.lm4 = glm(Mapk1~Akt2+Rik+Pik3r3+Rac1, data = kidney)
summary(fit.lm4)
```

```
##
## Call:
## glm(formula = Mapk1 ~ Akt2 + Rik + Pik3r3 + Rac1, data = kidney)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -0.16348  -0.06566   0.00381   0.05495   0.17092
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.44461    0.15706  -2.831 0.007643 **
```

```
## Akt2        -0.40548    0.16922  -2.396 0.022047 *
## Rik          0.22072    0.10178   2.169 0.036994 *
## Pik3r3       0.24399    0.11350   2.150 0.038565 *
## Rac1         0.31716    0.08464   3.747 0.000644 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for gaussian family taken to be 0.006857599)
##
##     Null deviance: 0.61566  on 39  degrees of freedom
## Residual deviance: 0.24002  on 35  degrees of freedom
## AIC: -79.122
##
## Number of Fisher Scoring iterations: 2
```

```
cv.out2 = cv.glm(kidney, fit.lm4)
cv.out2$delta[1]
```

```
## [1] 0.007906196
```

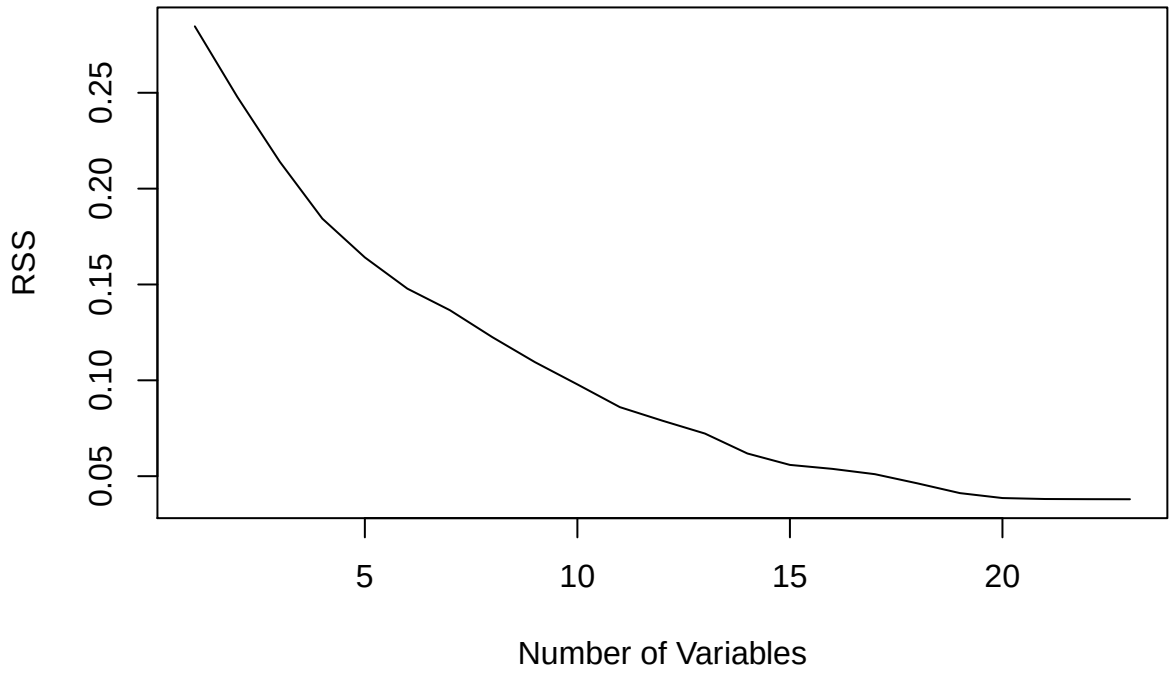
5 variables: Akt2; Rik; Pik3r3;Pik3r1; Rac1

```
fit.lm5 = glm(Mapk1~Akt2+Rik+Pik3r3+Pik3r1+Rac1, data = kidney)
cv.out3 = cv.glm(kidney, fit.lm5)
cv.out3$delta[1]
```

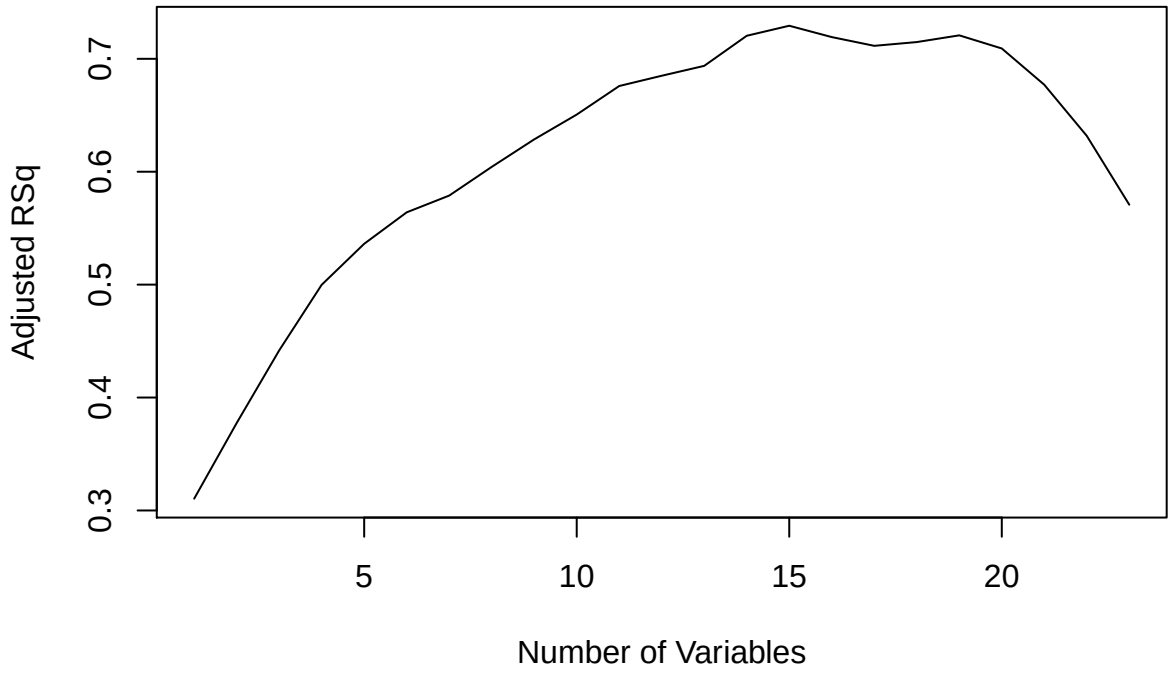
```
## [1] 0.00757514
```

forward

```
regfit.fwd=regsubsets(Mapk1~.,data=kidney[train,],nvmax=23,method="forward")
fwd.summary = summary(regfit.fwd)
plot(fwd.summary$rss,xlab="Number of Variables",ylab="RSS",type="l")
```



```
plot(fwd.summary$adjr2,xlab="Number of Variables",ylab="Adjusted RSq",type="l")
```




```
which.max(fwd.summary$adjr2)  #15
```

```
## [1] 15
```

```
which.min(fwd.summary$cp)#9
```

```
## [1] 9
```

```
which.min(fwd.summary$bic)#5
```

```
## [1] 5
```

```
subvar.15=names(which(fwd.summary$which[which.max(fwd.summary$adjr2),] == "TRUE")[-1])
subvar.15full = c("Mapk1", subvar.15)
```

with 15 variables:

```
fwd.lm15 = lm(Mapk1~., data = kidney[train,subvar.15full])
pred.fwd.lm15 = predict(fwd.lm15, newdata =kidney[test,subvar.15])
mean((pred.fwd.lm15-kidney[test,]$Mapk1)^2)
```

```
## [1] 0.044479
```

with 9 variables:

```
subvar.9=names(which(fwd.summary$which[which.min(fwd.summary$cp),] == "TRUE")[-1])
subvar.9full = c("Mapk1", subvar.9)

fwd.lm9 = lm(Mapk1~., data = kidney[train, subvar.9full])
pred.fwd.lm9 = predict(fwd.lm9, newdata =kidney[test,subvar.9])
mean((pred.fwd.lm9-kidney[test,]$Mapk1)^2)
```

```
## [1] 0.01306337
```

with 5 variables:

```
subvar.5=names(which(fwd.summary$which[which.min(fwd.summary$bic),] == "TRUE")[-1])
subvar.5full = c("Mapk1", subvar.5)
fwd.lm5 = lm(Mapk1~., data = kidney[train, subvar.5full])
pred.fwd.lm5 = predict(fwd.lm5, newdata =kidney[test,subvar.5])
mean((pred.fwd.lm5-kidney[test,]$Mapk1)^2)
```

```
## [1] 0.007990794
```

backward

```
regfit.bwd=regsubsets(Mapk1~.,data=kidney,nvmax=23,method="backward")
bwd.summary = summary(regfit.bwd)

which.max(bwd.summary$adjr2)
```

```
## [1] 13
```

```
which.min(bwd.summary$cp)
```

```
## [1] 7
```

```
which.min(bwd.summary$bic)
```

```
## [1] 7
```

with 13 variables:

```
bwd.lm13 = glm(Mapk1~Cdc42+Pla2g6+Akt2+Plcg2+Rac2+Pla2g5+Sphk2+Map2k1+Ptk2+Nos3+Pik3ca+Ppp3cb+Rac1, data
cv.bwd13 = cv.glm(kidney, bwd.lm13)
cv.bwd13$delta[1]
```

```
## [1] 0.008602781
```

with 7 variables:

```
bwd.lm7 = glm(Mapk1~Cdc42+Akt2+Plcg2+Rac2+Sphk2+Ppp3cb+Rac1, data = kidney)
cv.bwd7 = cv.glm(kidney, bwd.lm7)
cv.bwd7$delta[1]
```

```
## [1] 0.008279871
```

Ridge:

```
x =model.matrix(Mapk1~.,kidney[train,])[,-1]
y = kidney[train,]$Mapk1
library(glmnet)
```

```
## Warning: package 'glmnet' was built under R version 3.2.4
```

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

```
## Warning: package 'Matrix' was built under R version 3.2.5
```

```
## Loading required package: foreach
```

```
## Loaded glmnet 2.0-5
```

```
grid=10^seq(10,-2,length=100)   #create a grid for \lambda
ridge.mod=glmnet(x,y,alpha=0,lambda=grid) #alpha=0 is the ridge penalty, alpha=1 is the lasso penalty
set.seed(1)
cv.out=cv.glmnet(x,y,alpha=0, nfold = 5)  # 5 fold cross validation
bestlam.ridge=cv.out$lambda.min
bestlam.ridge
```

```
## [1] 0.3435062
```

```
ridge.mod=glmnet(x,y,alpha=0,lambda=bestlam.ridge)
xtest = model.matrix(Mapk1~., kidney[test,])[,-1]
ytest = kidney[test,]$Mapk1
mean((predict(ridge.mod, s = bestlam.ridge, newx = xtest)-ytest)^2)
```

```
## [1] 0.01114473
```

Lasso:

```
set.seed(1)
cv.out=cv.glmnet(x,y,alpha=1, nfold = 5)  #5 fold cross validation
bestlam.lasso=cv.out$lambda.min
bestlam.lasso #0.004707
```

```
## [1] 0.04334557
```

```
lasso.mod=glmnet(x,y,alpha=1,lambda=bestlam.lasso)
pred.lasso = predict(lasso.mod, s = bestlam.lasso, newx = xtest)
mean((pred.lasso-ytest)^2) #0.01175695
```

```
## [1] 0.01575946
```

```
lasso.coef = coef(lasso.mod)
lasso.coef[lasso.coef!=0]
```

```
## <sparse>[ <logic> ] : .M.sub.i.logical() maybe inefficient
```

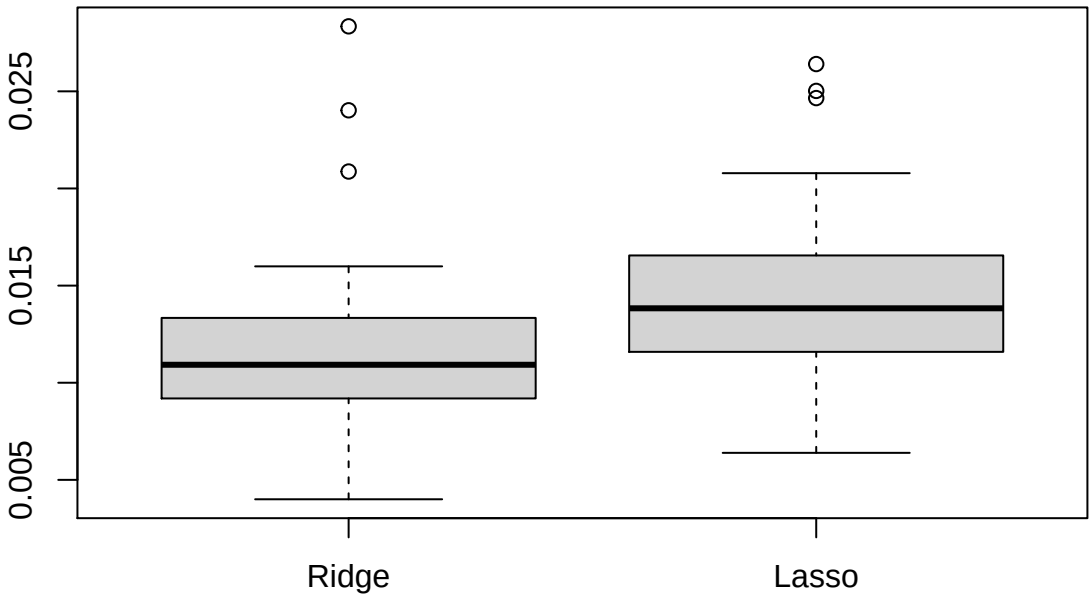
```
## [1] 0.001082049 0.075613453 0.091374208 0.043440760
```

Loop and boxplot

```
ridge.err = matrix(0,nrow = 50, ncol = 1)
lasso.err = matrix(0,nrow = 50, ncol = 1)
for(i in 1:50){
  set.seed(i)
  train = sample(1:40, 30)
  test = (-train)

  x =model.matrix(Mapk1~.,kidney[train,])[,-1]
  y = kidney[train,]$Mapk1
  cv.out=cv.glmnet(x,y,alpha=0, nfold = 5)  # 5 fold cross validation
  bestlam.ridge=cv.out$lambda.min
  ridge.mod=glmnet(x,y,alpha=0,lambda=bestlam.ridge)
  xtest = model.matrix(Mapk1~., kidney[test,])[,-1]
  ytest = kidney[test,]$Mapk1
  ridge.err[i,1] = mean((predict(ridge.mod, s = bestlam.ridge, newx = xtest)-ytest)^2)

  cv.out=cv.glmnet(x,y,alpha=1, nfold = 5)  #5 fold cross validation
  bestlam.lasso=cv.out$lambda.min
  bestlam.lasso
  lasso.mod=glmnet(x,y,alpha=1,lambda=bestlam.lasso)
  pred.lasso = predict(lasso.mod, s = bestlam.lasso, newx = xtest)
  lasso.err[i,1] = mean((pred.lasso-ytest)^2) #0.0117569
}
boxplot(cbind(ridge.err,lasso.err),names = c("Ridge","Lasso"), main = "MSE Boxplot vs Ridge & Lasso", c
```



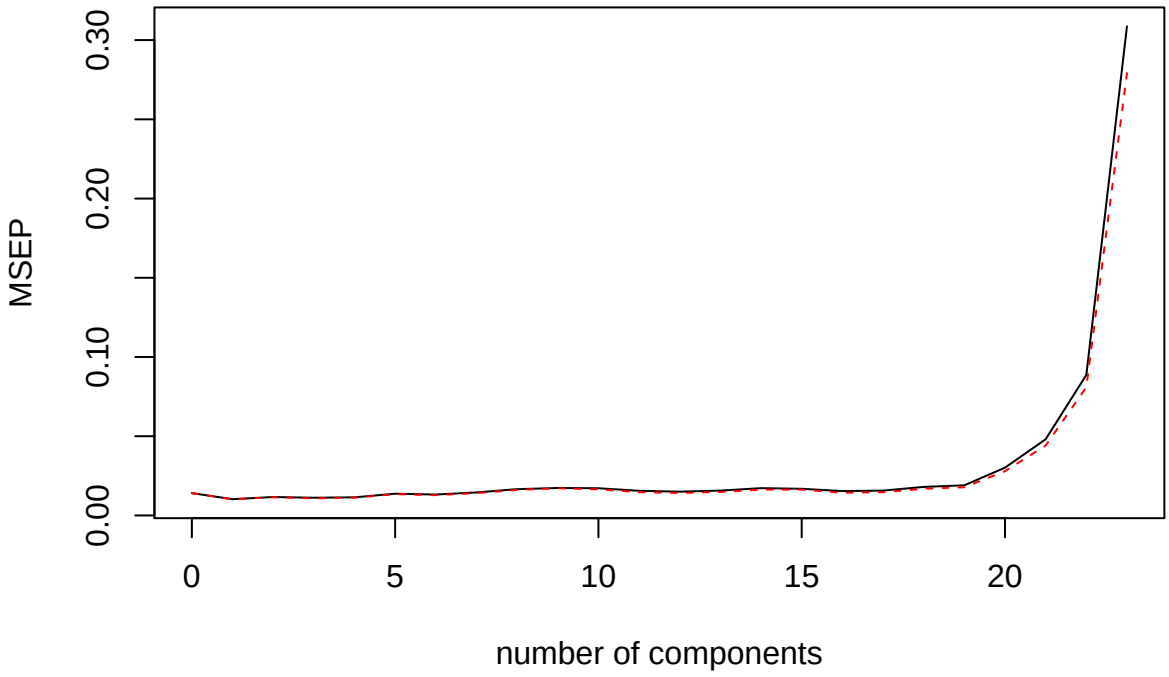
PCR:

```
library(pls)
```

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

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

```
set.seed(1)
pcr.fit = pcr(Mapk1~., data = kidney[train,],scale = TRUE, validation = "CV")
validationplot(pcr.fit, val.type = "MSEP")
```



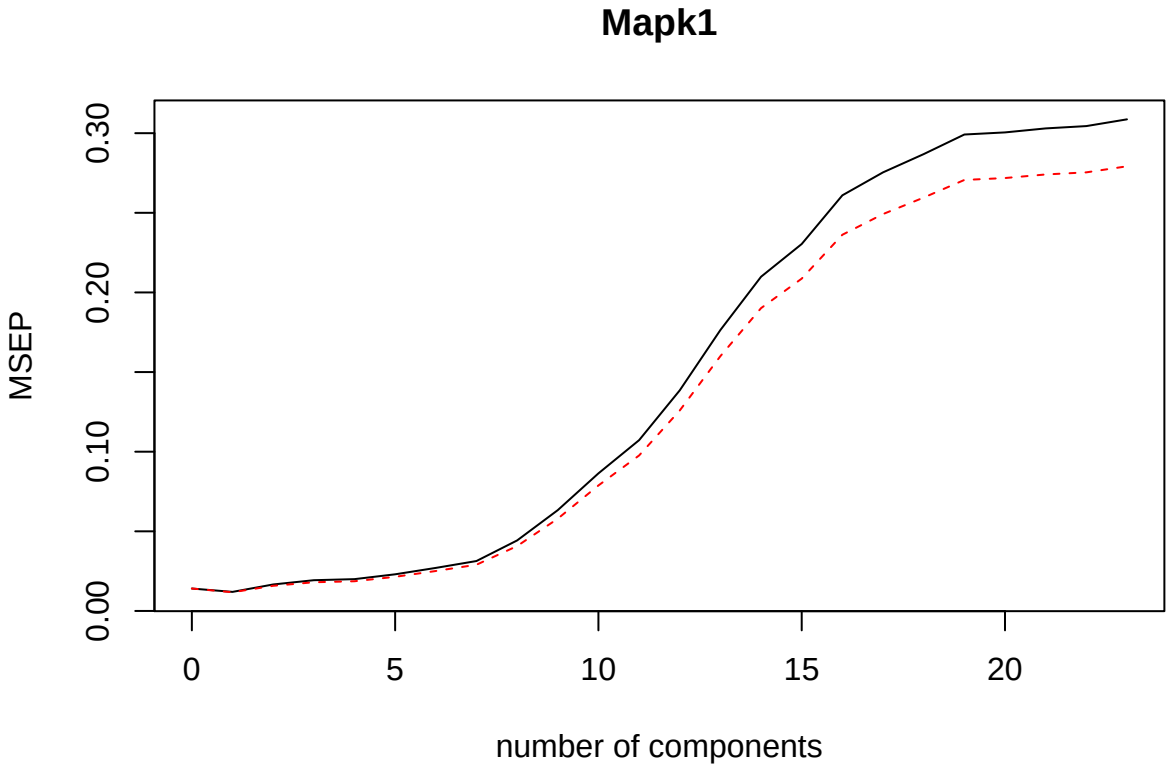
As we can see in the plot, when M = 4 yield smallest MSEP So we compute MSE as follow:

```
pcr.pred = predict(pcr.fit, kidney[test,], ncomp = 4)
mean((pcr.pred[1]-ytest)^2)
```

```
## [1] 0.03057632
```

PLS

```
set.seed(1)
pls.fit = plsr(Mapk1~., data = kidney[train,], scale = TRUE, validation = "CV")
validationplot(pls.fit,val.type = "MSEP")
```




```
pls.pred = predict(pls.fit, kidney[test,], ncomp = 1)
mean((pls.pred-kidney[test,]$Mapk1)^2)
```

```
## [1] 0.009064731
```

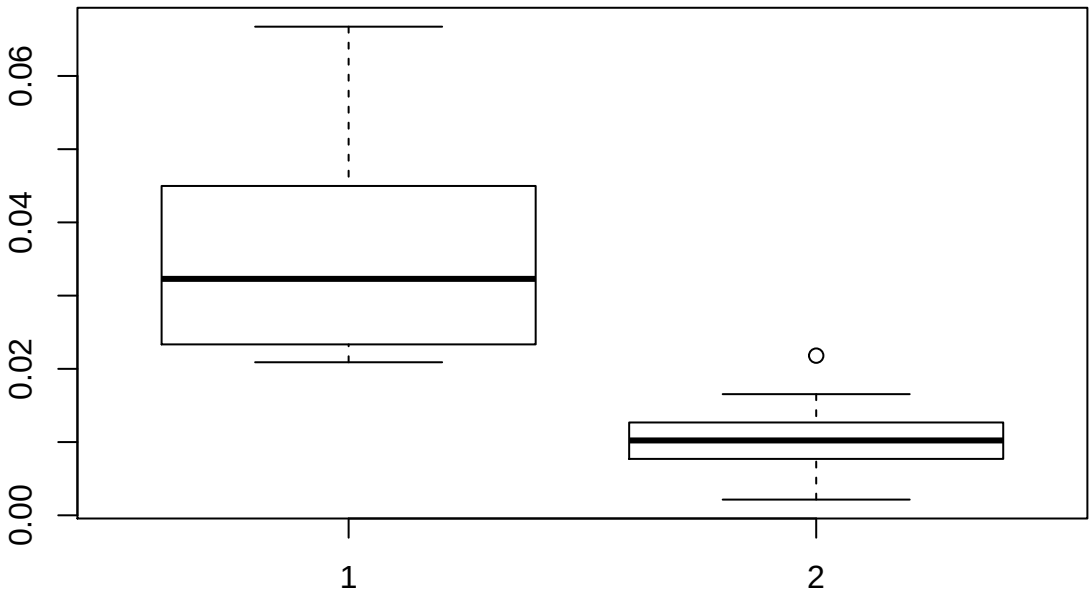
for loop with multiple seeds dimension reduction:

```
pcr.err = matrix(0, nrow = 50, ncol = 1)
pls.err = matrix(0, nrow = 50, ncol = 1)
for(i in 1:50){
  set.seed(i)
  train = sample(1:40, 30)
  test = (-train)
  pcr.fit = pcr(Mapk1~., data = kidney[train,],scale = TRUE, validation = "CV")
  pcr.pred = predict(pcr.fit, kidney[test,], ncomp = 4)
  pcr.err[i,1]=mean((pcr.pred[1]-ytest)^2)

  pls.fit = plsr(Mapk1~., data = kidney[train,], scale = TRUE, validation = "CV")
  pls.pred = predict(pls.fit, kidney[test,], ncomp = 1)
  pls.err[i,1]=mean((pls.pred-kidney[test,]$Mapk1)^2)

}

boxplot(cbind(pcr.err, pls.err))
```



Random Forest:

```
library (randomForest)
```

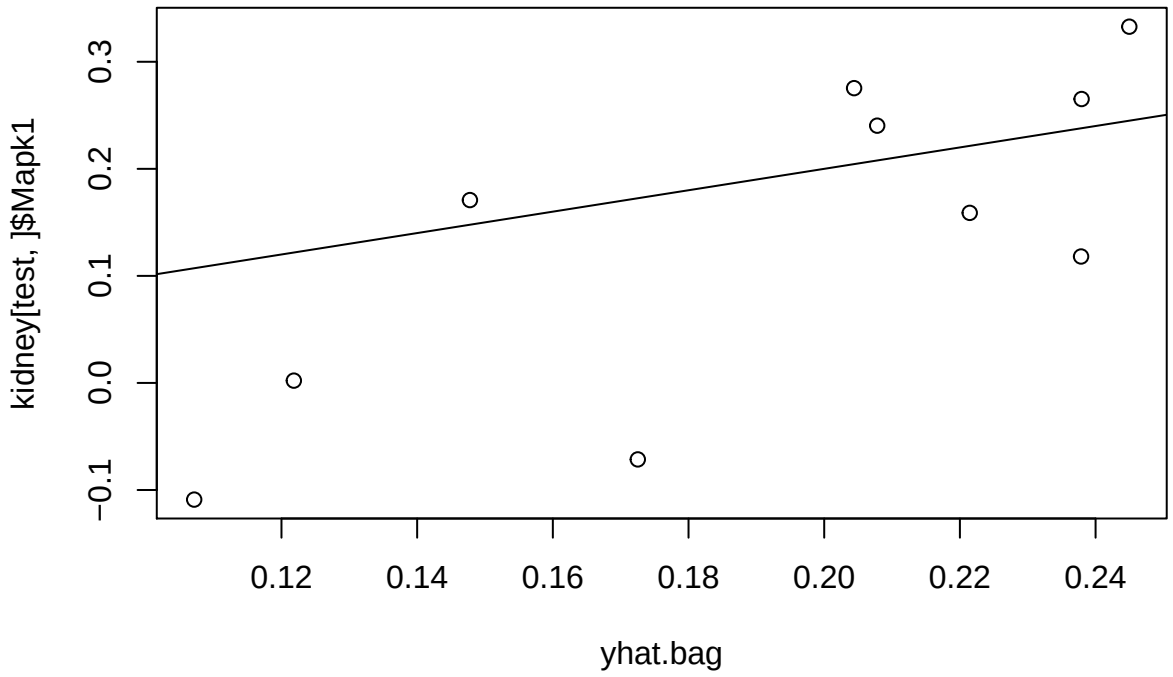
```
## randomForest 4.6-12

## Type rfNews() to see new features/changes/bug fixes.
```

```
set.seed (1)
bag.kidney = randomForest(Mapk1 ~. ,data=kidney ,subset =train, mtry=23, importance =TRUE)
bag.kidney
```

```
##
## Call:
##  randomForest(formula = Mapk1 ~ ., data = kidney, mtry = 23, importance = TRUE,      subset = train)
##                Type of random forest: regression
##                      Number of trees: 500
## No. of variables tried at each split: 23
##
##           Mean of squared residuals: 0.01220412
##                     % Var explained: 7.26
```

```
## Bagging
yhat.bag = predict (bag.kidney ,newdata =kidney[test,])
plot(yhat.bag, kidney[test,]$Mapk1)
abline (0,1)
```



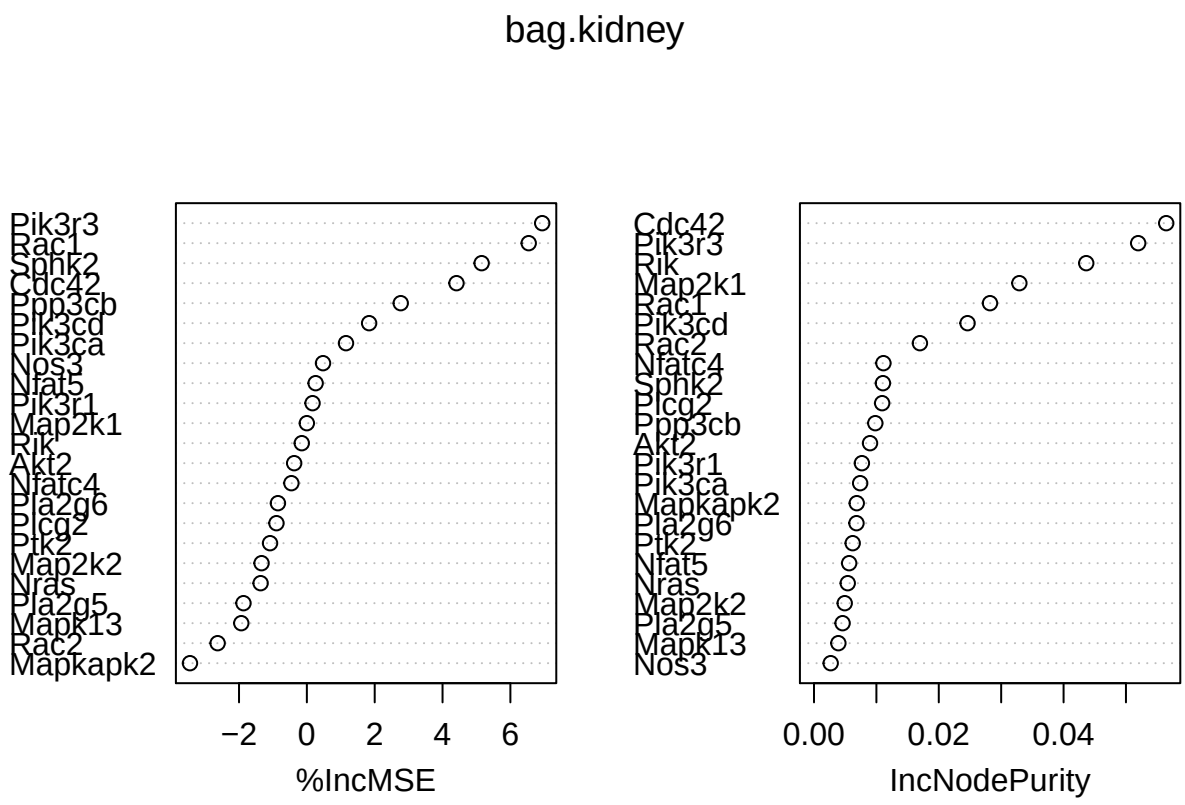

```
mean((yhat.bag - kidney[test,]$Mapk1)^2)
```

```
## [1] 0.01538727
```

```
importance(bag.kidney)
```

```
##                %IncMSE IncNodePurity
## Cdc42     4.4113763666   0.056450936
## Pla2g6   -0.8524981435   0.006810763
## Akt2     -0.3739238191   0.008981560
## Plcg2    -0.8962996598   0.010923329
## Rac2     -2.6298001751   0.016976224
## Rik      -0.1496868372   0.043632689
## Mapkapk2 -3.4454597987   0.006853431
## Pik3cd    1.8361304158   0.024611232
## Pla2g5   -1.8686575150   0.004583953
## Sphk2     5.1526233463   0.011050401
## Map2k1   -0.0006555316   0.032917069
## Pik3r3    6.9374316105   0.051960332
## Ptk2     -1.0851456775   0.006200674
## Nras     -1.3628557255   0.005407841
## Nos3      0.4765219090   0.002677512
## Pik3r1    0.1676667355   0.007676851
## Pik3ca    1.1538453499   0.007397432
## Ppp3cb    2.7674432337   0.009824797
## Map2k2   -1.3350649103   0.004924585
## Nfatc4   -0.4561129269   0.011125076
## Mapk13   -1.9327275166   0.003905593
## Rac1      6.5402063013   0.028210517
## Nfat5     0.2591210757   0.005636071
```

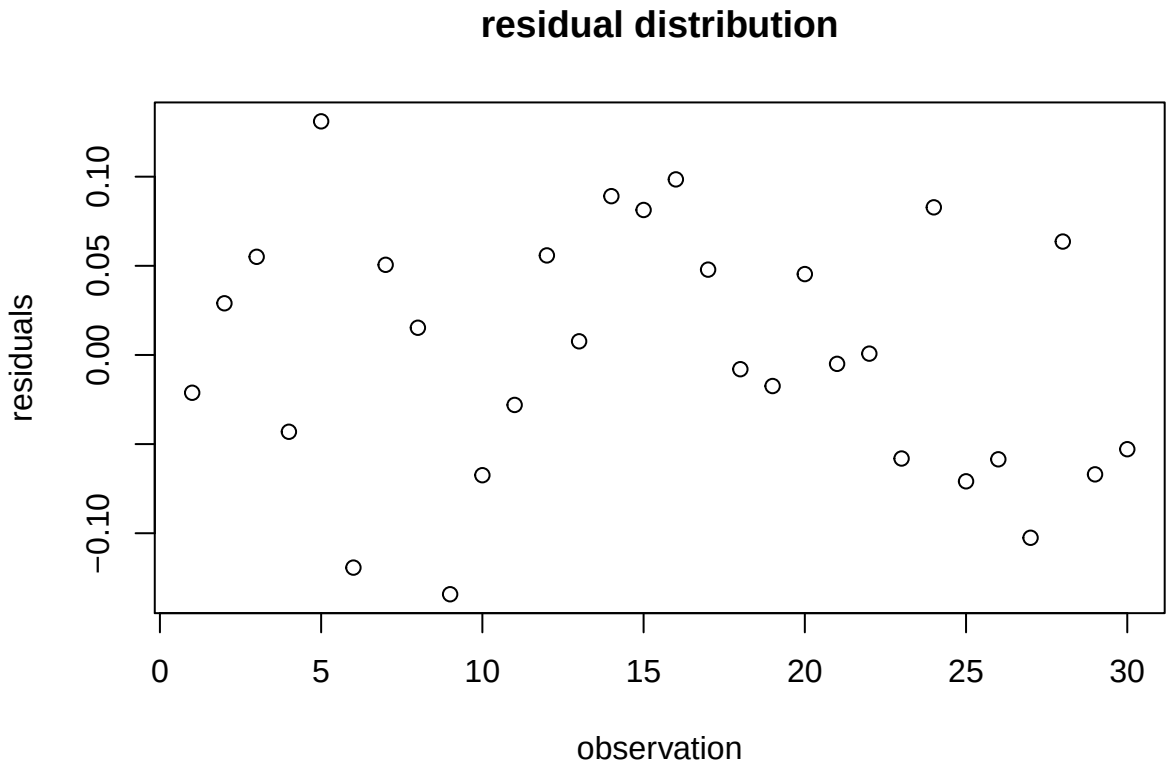
```
varImpPlot(bag.kidney)
```




```
bag.kidney =randomForest(Mapk1 ~., data=kidney, subset =train,mtry=8, importance = TRUE)
yhat.bag = predict(bag.kidney,newdata =kidney[test,])
mean((yhat.bag-kidney[test,]$Mapk1)^2)
```

```
## [1] 0.01539891
```

```
fit.best = lm(Mapk1~Akt2+Rik+Sphk2+Pik3r1+Rac1, data = kidney[train,])
# check the model violation.
plot(fit.best$residuals, ylab = "residuals",xlab = "observation",main = "residual distribution")
```




```
# check correlation between the predictors:
pairs(kidney[,c("Mapk1","Akt2","Rik","Sphk2","Pik3r1","Rac1")])
```