Variable Selection in Regression

Simulate data from the following settings

Let the **true model** be

$$y_i = 10 + 0.5x_{1i} - 5x_{2i} + \epsilon_i,$$

where $\epsilon_i \sim N(0, 0.49)$ and $i = 1, \ldots, 25$. Let the predictors be simulated from

$$x_{1i} \sim U(-2, 2),$$

$$x_{2i} \sim U(-1, 4).$$

We do have other variables:

$$x_{3i} = 1 + 0.8x_{1i} + e_i,$$

$$x_{4i} = 2 + 0.2x_{1i} + e_i$$

$$x_{5i} = -0.5x_{1i} + e_i,$$

$$x_{6i} = 2 + e_i$$

where $e_i \sim N(0, 0.7^2)$.

Note that variables x_1 and x_2 affects the response y. The other variables x_3, x_4, x_5 are correlated to x_1.

```r
set.seed(233300)
n.sample <- 25
error <- rnorm(n.sample, 0, 0.7)

x1 <- runif(n.sample, -2, 2)
x2 <- runif(n.sample, -1, 4)
x3 <- 1 + 0.8*x1 + rnorm(n.sample, 1, 0.5)
x4 <- 2 + 0.2*x1 + rnorm(n.sample, 2, 0.5)
x5 <- -0.5*x1 + rnorm(n.sample, 0, 0.5)
x6 <- rnorm(n.sample, 2, 0.5)
X <- matrix(NA, n.sample, 6)
X[,1] <- x1
X[,2] <- x2
X[,3] <- x3
X[,4] <- x4
X[,5] <- x5
X[,6] <- x6
colnames(X) <- c("x1", "x2", "x3", "x4", "x5", "x6")

### True model ###
y <- 10 + 0.5*x1 - 5*x2 + error

### Training set and testing set ###
data.train <- data.frame(X[1:20,])
data.train$y <- y[1:20]
data.test <- data.frame(X[21:25,])
data.test$y <- y[21:25]
```

Fit regression models by using the training set

(a) Fit the "true" regression model by x_1 and x_2.

```r
library(car)
```

```
##     carData
```

```r
fit0 <- lm(y~x1+x2, data = data.train)
summary(fit0)
```

```
##
## Call:
## lm(formula = y ~ x1 + x2, data = data.train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -1.7047 -0.4383  0.1868  0.3711  1.5271
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   9.6235     0.2886  33.351  < 2e-16 ***
## x1            0.5066     0.1514   3.347  0.00382 **
## x2           -4.9129     0.1376 -35.695  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.8201 on 17 degrees of freedom
## Multiple R-squared:  0.9882, Adjusted R-squared:  0.9868
## F-statistic: 710.1 on 2 and 17 DF,  p-value: < 2.2e-16
```

```r
vif(fit0)
```

```
##       x1       x2
## 1.058586 1.058586
```

(b) Fit a regression model by $x_1, \ldots, x_6$

```r
fit1 <- lm(y~., data = data.train)
summary(fit1)
```

```
##
## Call:
## lm(formula = y ~ ., data = data.train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -1.7952 -0.3342  0.1027  0.4271  1.0735
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
```

```
## (Intercept) 10.27462    2.23489   4.597   0.0005 ***
## x1           0.26921    0.42436   0.634   0.5368
## x2          -4.95677    0.17434 -28.432 4.32e-13 ***
## x3           0.03651    0.41926   0.087   0.9319
## x4           0.14436    0.34447   0.419   0.6820
## x5          -0.16624    0.64417  -0.258   0.8004
## x6          -0.60585    0.51290  -1.181   0.2587
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.8716 on 13 degrees of freedom
## Multiple R-squared:  0.9898, Adjusted R-squared:  0.9851
## F-statistic: 209.9 on 6 and 13 DF,  p-value: 3.602e-12
```

```r
vif(fit1)
```

```
##       x1       x2       x3       x4       x5       x6
## 7.364237 1.503585 4.992045 1.143809 5.567599 1.939979
```

The result shows that variable x_1 is not significant in the model if all predictors are in the model.

> **What is the reason?**
>
> Variables X are correlated!!!
>
> Note that the correlation coefficients of x_1, x_3, and x_5 are large. Hence, the multicollinearity of predictors may affect the result when we fit a model including all variables.

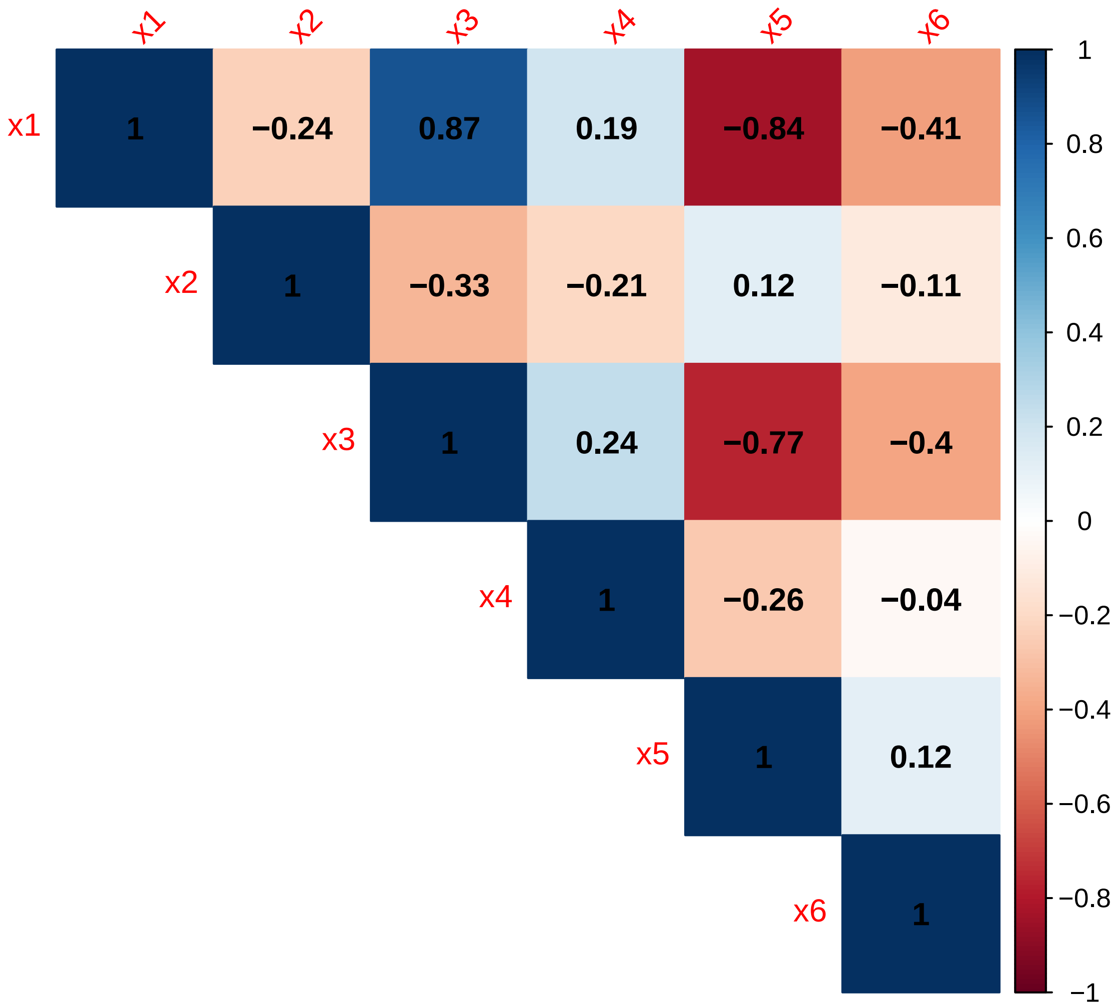
```r
library(corrplot)
```

```
## corrplot 0.92 loaded
```

```r
cor(data.train[,1:6])
```

```
##            x1         x2         x3          x4         x5          x6
## x1  1.0000000 -0.2352514  0.8675893  0.19462042 -0.8397783 -0.41221934
## x2 -0.2352514  1.0000000 -0.3346983 -0.20586999  0.1202139 -0.11080916
## x3  0.8675893 -0.3346983  1.0000000  0.24057052 -0.7666987 -0.39583847
## x4  0.1946204 -0.2058700  0.2405705  1.00000000 -0.2649732 -0.03810505
## x5 -0.8397783  0.1202139 -0.7666987 -0.26497321  1.0000000  0.11757858
## x6 -0.4122193 -0.1108092 -0.3958385 -0.03810505  0.1175786  1.00000000
```

```r
corrplot(cor(data.train[,1:6]), method = 'color', addCoef.col = 'black', type = 'upper', tl.srt = 45)
```




```r
#pairs(data.train[,1:6], pch = 19)
```

How to handle this issue?

1. Choose only one variable among all highly correlated variables with a meaningful or practice reason.
2. Use the stepwise regression to choose variables.
3. Use the PCA technique to summarize the variables and then fit a regrssion model by the PC scores.
4. Use the shrinkage method to select variables.
5. Use the partial least square (PLS) method.

The strategies are as follows:

1. **Choosing only one variable is sometimes too subjective.**

2. Stepwise regression

Here, use the stepwise regression with forward and backward scheme on p-value to choose variables. (You can use other types of stepwise regression.)

```r
library(olsrr)
```

Both ways with p-values:

The default of p-values are $pent = 0.1, prem = 0.3$.

```r
summary(fit1)
```

```
##
## Call:
## lm(formula = y ~ ., data = data.train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -1.7952 -0.3342  0.1027  0.4271  1.0735
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 10.27462    2.23489   4.597   0.0005 ***
## x1           0.26921    0.42436   0.634   0.5368
## x2          -4.95677    0.17434 -28.432 4.32e-13 ***
## x3           0.03651    0.41926   0.087   0.9319
## x4           0.14436    0.34447   0.419   0.6820
## x5          -0.16624    0.64417  -0.258   0.8004
## x6          -0.60585    0.51290  -1.181   0.2587
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.8716 on 13 degrees of freedom
## Multiple R-squared:  0.9898, Adjusted R-squared:  0.9851
## F-statistic: 209.9 on 6 and 13 DF,  p-value: 3.602e-12
```

```r
stepwise0 <- ols_step_both_p(fit1)
stepwise0
```

```
##
##                              Stepwise Selection Summary
## ----------------------------------------------------------------------------------
##                      Added/                   Adj.
## Step    Variable    Removed     R-Square    R-Square     C(p)       AIC        RMSE
## ----------------------------------------------------------------------------------
##    1       x2       addition       0.980       0.979    8.9660    61.6943    1.0264
##    2       x1       addition       0.988       0.987    1.0500    53.5717    0.8201
## ----------------------------------------------------------------------------------
```

Use different p-values: p=0.15.

```r
stepwise1 <- ols_step_both_p(fit1, pent = 0.15, prem = 0.15)
stepwise1
```

```
##
##                              Stepwise Selection Summary
## -------------------------------------------------------------------------------------
##                     Added/                   Adj.
## Step    Variable   Removed     R-Square    R-Square     C(p)        AIC        RMSE
## -------------------------------------------------------------------------------------
##    1       x2      addition       0.980       0.979    8.9660    61.6943    1.0264
##    2       x1      addition       0.988       0.987    1.0500    53.5717    0.8201
## -------------------------------------------------------------------------------------
```

Use different p-values: p=0.3.

```r
stepwise2 <- ols_step_both_p(fit1, pent = 0.3, prem = 0.3)
stepwise2
```

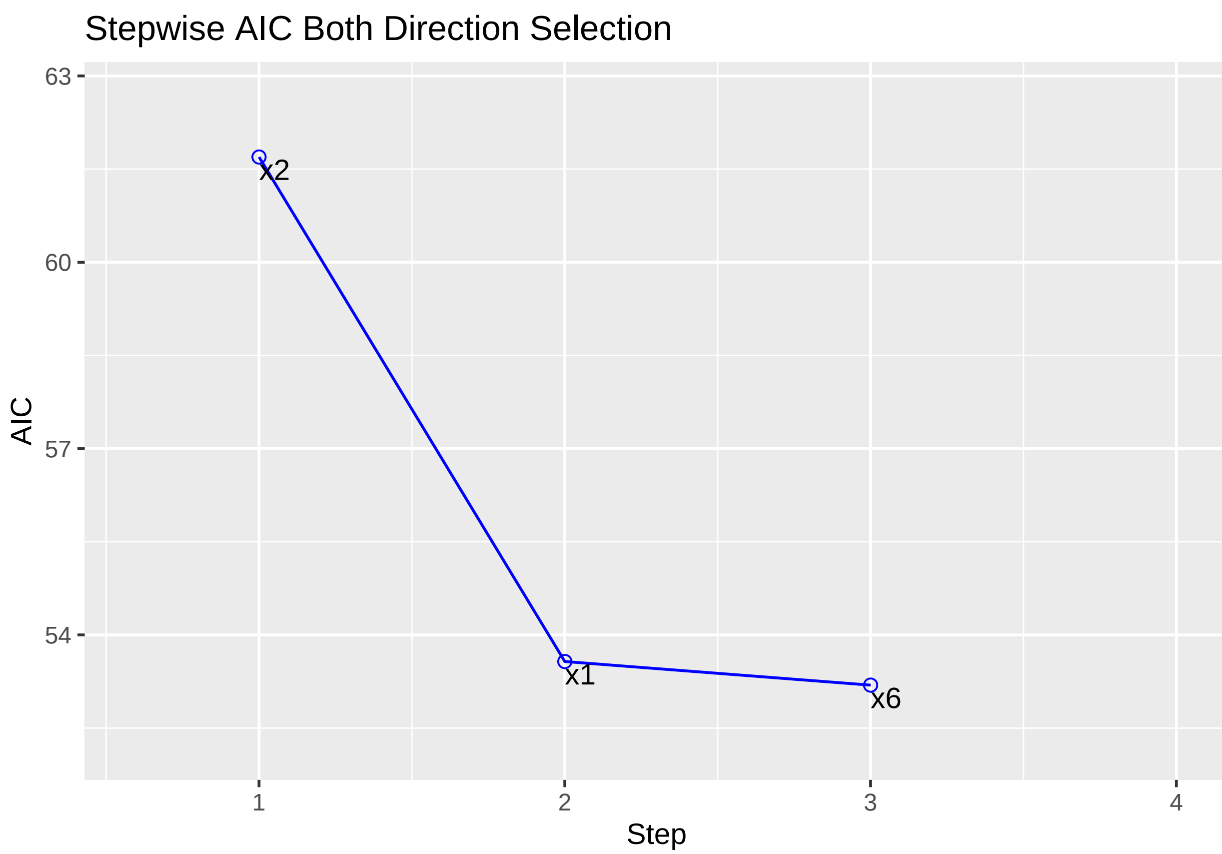
```
##
##                              Stepwise Selection Summary
## -------------------------------------------------------------------------------------
##                     Added/                   Adj.
## Step    Variable   Removed     R-Square    R-Square     C(p)        AIC        RMSE
## -------------------------------------------------------------------------------------
##    1       x2      addition       0.980       0.979    8.9660    61.6943    1.0264
##    2       x1      addition       0.988       0.987    1.0500    53.5717    0.8201
##    3       x6      addition       0.989       0.988    1.3610    53.1916    0.7965
## -------------------------------------------------------------------------------------
```

Both ways with AIC:

```r
stepwise3 <- ols_step_both_aic(fit1)
stepwise3
```

```
##
##
##                               Stepwise Summary
## -----------------------------------------------------------------------------
## Variable     Method       AIC       RSS      Sum Sq      R-Sq      Adj. R-Sq
## -----------------------------------------------------------------------------
## x2          addition    61.694    18.965    947.483    0.98038      0.97929
## x1          addition    53.572    11.432    955.015    0.98817      0.98678
## x6          addition    53.192    10.150    956.298    0.98950      0.98753
## -----------------------------------------------------------------------------
```

```r
plot(stepwise3)
```



Different criteria give slightly different results, and you can choose one with similarly selected variables. Fortunately, the results show that the significant variables are x_1 and x_2. But, it is not always a lucky case like this.

3. PCA technique on training set

What is the difference using the covariance matrix of X and the correlation matrix of X?

```
pca0 <- princomp(data.train[,1:6]) # by covariance matrix
summary(pca0)
```

```
## Importance of components:
##                           Comp.1    Comp.2     Comp.3     Comp.4     Comp.5
## Standard deviation     1.787443 1.2780830 0.58966492 0.52650581 0.39434059
## Proportion of Variance 0.563817 0.2882649 0.06135984 0.04891927 0.02744205
## Cumulative Proportion  0.563817 0.8520818 0.91344167 0.96236094 0.98980299
##                            Comp.6
## Standard deviation     0.24038067
## Proportion of Variance 0.01019701
## Cumulative Proportion  1.00000000
```

```
pca0$loadings
```

```
##
## Loadings:
```

```
##    Comp.1 Comp.2 Comp.3 Comp.4 Comp.5 Comp.6
## x1  0.641  0.329  0.125         0.563  0.375
## x2 -0.426  0.890                       0.104
## x3  0.538  0.169        -0.180 -0.791  0.146
## x4  0.101        -0.927 -0.330         0.108
## x5 -0.317 -0.213  0.233 -0.478         0.756
## x6        -0.153 -0.247  0.789 -0.201  0.494
##
##                 Comp.1 Comp.2 Comp.3 Comp.4 Comp.5 Comp.6
## SS loadings      1.000  1.000  1.000  1.000  1.000  1.000
## Proportion Var  0.167  0.167  0.167  0.167  0.167  0.167
## Cumulative Var  0.167  0.333  0.500  0.667  0.833  1.000
```

```
pca <- princomp(data.train[,1:6], cor = TRUE) # by correlation matrix
summary(pca)
```

```
## Importance of components:
##                           Comp.1    Comp.2    Comp.3    Comp.4    Comp.5
## Standard deviation     1.7241883 1.0903576 0.9149928 0.8805264 0.3748760
## Proportion of Variance 0.4954709 0.1981466 0.1395353 0.1292211 0.0234220
## Cumulative Proportion  0.4954709 0.6936175 0.8331528 0.9623739 0.9857959
##                            Comp.6
## Standard deviation     0.29193243
## Proportion of Variance 0.01420409
## Cumulative Proportion  1.00000000
```

```
pca$loadings
```

```
##
## Loadings:
##    Comp.1 Comp.2 Comp.3 Comp.4 Comp.5 Comp.6
## x1  0.551  0.119  0.142         0.331  0.741
## x2 -0.197  0.649 -0.292  0.626 -0.209  0.136
## x3  0.547         0.121        -0.823
## x4  0.214 -0.446 -0.859  0.110
## x5 -0.502        -0.138 -0.488 -0.359  0.601
## x6 -0.248 -0.603  0.352  0.590 -0.199  0.250
##
##                 Comp.1 Comp.2 Comp.3 Comp.4 Comp.5 Comp.6
## SS loadings      1.000  1.000  1.000  1.000  1.000  1.000
## Proportion Var  0.167  0.167  0.167  0.167  0.167  0.167
## Cumulative Var  0.167  0.333  0.500  0.667  0.833  1.000
```

The results show that the cumulative proportion of PC1 to PC3 are around 91%. We can choose the first two or first three PCs to be the predictors.

```
z01 <- pca0$scores[,1]
z02 <- pca0$scores[,2]
z03 <- pca0$scores[,3]
z04 <- pca0$scores[,4]
z05 <- pca0$scores[,5]
z06 <- pca0$scores[,6]
```

```r
fit.by.pca0 <- lm(y~z01+z02+z03+z04+z05+z06, data=data.train)
summary(fit.by.pca0)
```

```
##
## Call:
## lm(formula = y ~ z01 + z02 + z03 + z04 + z05 + z06, data = data.train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -1.7952 -0.3342  0.1027  0.4271  1.0735
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   1.6028     0.1949   8.224 1.65e-06 ***
## z01           2.4274     0.1090  22.263 9.78e-12 ***
## z02          -4.1962     0.1525 -27.519 6.56e-13 ***
## z03           0.4358     0.3305   1.319    0.210
## z04          -0.4675     0.3702  -1.263    0.229
## z05           0.6914     0.4942   1.399    0.185
## z06          -0.8195     0.8107  -1.011    0.331
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.8716 on 13 degrees of freedom
## Multiple R-squared:  0.9898, Adjusted R-squared:  0.9851
## F-statistic: 209.9 on 6 and 13 DF,  p-value: 3.602e-12
```

```r
z11 <- pca$scores[,1]
z12 <- pca$scores[,2]
z13 <- pca$scores[,3]
z14 <- pca$scores[,4]
z15 <- pca$scores[,5]
z16 <- pca$scores[,6]

fit.by.pca1 <- lm(y~z11+z12+z13+z14+z15+z16, data=data.train)
summary(fit.by.pca1)
```

```
##
## Call:
## lm(formula = y ~ z11 + z12 + z13 + z14 + z15 + z16, data = data.train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -1.7952 -0.3342  0.1027  0.4271  1.0735
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   1.6028     0.1949   8.224 1.65e-06 ***
## z11           1.7042     0.1130  15.077 1.30e-09 ***
## z12          -4.2177     0.1787 -23.597 4.67e-12 ***
## z13           1.8618     0.2130   8.741 8.37e-07 ***
## z14          -4.3579     0.2213 -19.689 4.63e-11 ***
```

```
## z15           1.6042     0.5199   3.086  0.00868 **
## z16          -0.8248     0.6676  -1.235  0.23852
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.8716 on 13 degrees of freedom
## Multiple R-squared:  0.9898, Adjusted R-squared:  0.9851
## F-statistic: 209.9 on 6 and 13 DF,  p-value: 3.602e-12
```

```r
fit.pca0 <- lm(y~z01+z02, data=data.train)
fit.pca1 <- lm(y~z11+z12+z13+z14+z15, data=data.train)
summary(fit.pca0)
```

```
##
## Call:
## lm(formula = y ~ z01 + z02, data = data.train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -1.6525 -0.7018  0.1642  0.6628  1.1906
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   1.6028     0.2077   7.716 5.95e-07 ***
## z01           2.4274     0.1162  20.888 1.47e-13 ***
## z02          -4.1962     0.1625 -25.819 4.45e-15 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.929 on 17 degrees of freedom
## Multiple R-squared:  0.9848, Adjusted R-squared:  0.983
## F-statistic: 551.4 on 2 and 17 DF,  p-value: 3.474e-16
```

```r
summary(fit.pca1)
```

```
##
## Call:
## lm(formula = y ~ z11 + z12 + z13 + z14 + z15, data = data.train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -1.5897 -0.3197  0.1495  0.4349  1.3927
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   1.6028     0.1985   8.074 1.23e-06 ***
## z11           1.7042     0.1151  14.801 6.07e-10 ***
## z12          -4.2177     0.1821 -23.166 1.45e-12 ***
## z13           1.8618     0.2170   8.581 6.00e-07 ***
## z14          -4.3579     0.2255 -19.329 1.71e-11 ***
## z15           1.6042     0.5296   3.029  0.00901 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
##
## Residual standard error: 0.8878 on 14 degrees of freedom
## Multiple R-squared:  0.9886, Adjusted R-squared:  0.9845
## F-statistic: 242.4 on 5 and 14 DF,  p-value: 4.437e-13
```

From the above results, we use the model with z_1, z_2, z_3, z_4, and z_5 to be a possible model by select the significant variables with the largest adj.R^2.

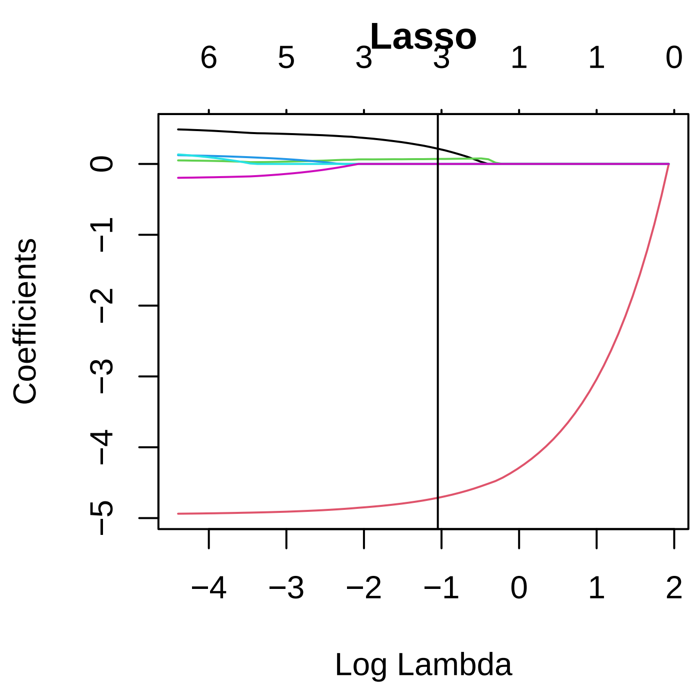
4. Shrinkage mothod (LASSO and Ridge regression)

LASSO Regression

```r
library(glmnet)
```

```r
fit.lasso <- glmnet(X, y, family = "gaussian", alpha = 1)
```

```r
cv.lasso = cv.glmnet(x = X, y = y, alpha = 1, nfolds = 6, family = "gaussian")
lambda.lasso = cv.lasso$lambda.1se
plot(fit.lasso, xvar='lambda', main="Lasso")
abline(v = log(lambda.lasso))
```




```r
coef(cv.lasso)
```

```
## 7 x 1 sparse Matrix of class "dgCMatrix"
##                      s1
## (Intercept)  9.24342117
```

```
## x1          0.21525697
## x2         -4.71353378
## x3          0.07081643
## x4          .
## x5          .
## x6          .
```

From the result, variables x_1, x_2, and x_3 are more important. Then, we can fit a model by these three variables.

5. Partial least squares (PLS) regression

```r
library(pls)
```

```
## Warning:   'pls'   R   4.3.2
```

```
##
##    'pls'
```

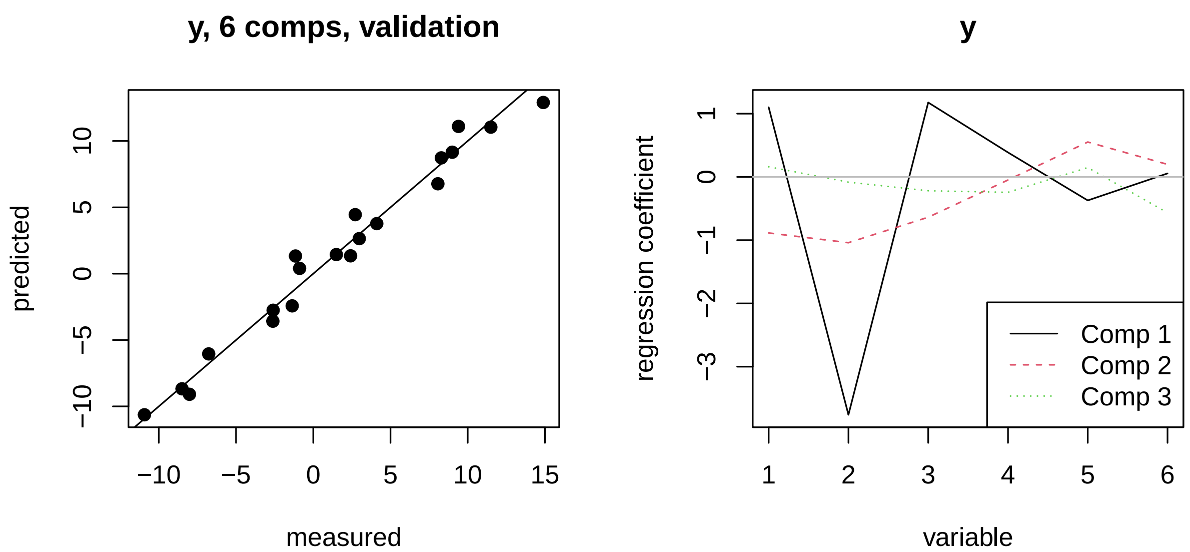
```
##      'package:corrplot':
##
##     corrplot
```

```
##      'package:stats':
##
##     loadings
```

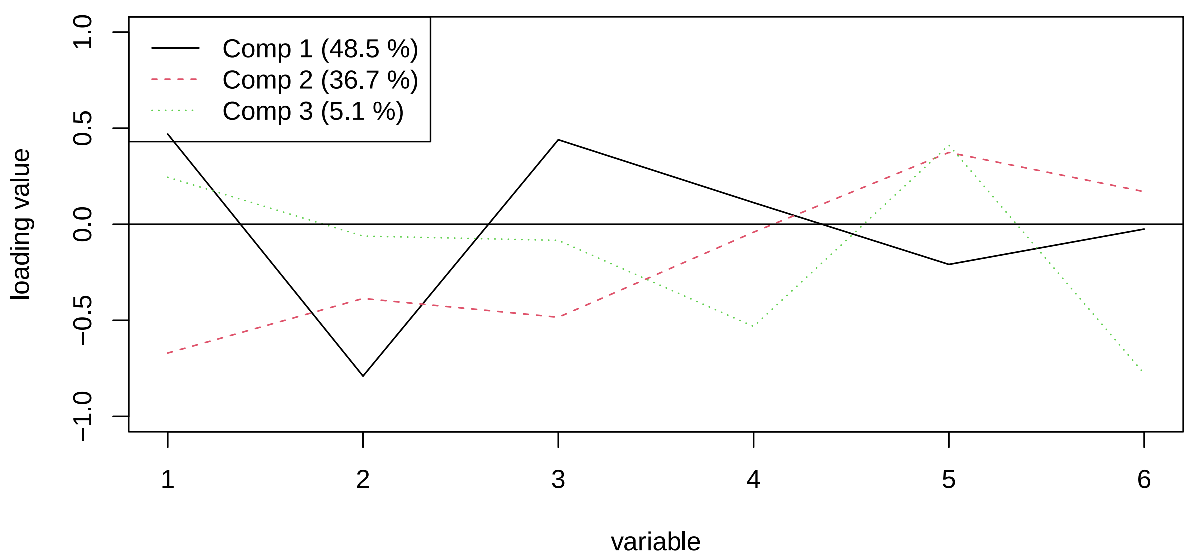
```r
pls.fit <- plsr(y ~ ., ncomp = 6, data = data.train, validation = "LOO")
summary(pls.fit)
```

```
## Data:    X dimension: 20 6
##  Y dimension: 20 1
## Fit method: kernelpls
## Number of components considered: 6
##
## VALIDATION: RMSEP
## Cross-validated using 20 leave-one-out segments.
##        (Intercept)  1 comps  2 comps  3 comps  4 comps  5 comps  6 comps
## CV           7.317    3.036    1.123    1.085    1.104    1.111    1.118
## adjCV        7.317    3.027    1.112    1.075    1.093    1.101    1.107
##
## TRAINING: % variance explained
##    1 comps  2 comps  3 comps  4 comps  5 comps  6 comps
## X    48.51    85.19    90.30    93.89    97.46   100.00
## y    88.41    98.65    98.91    98.96    98.97    98.98
```

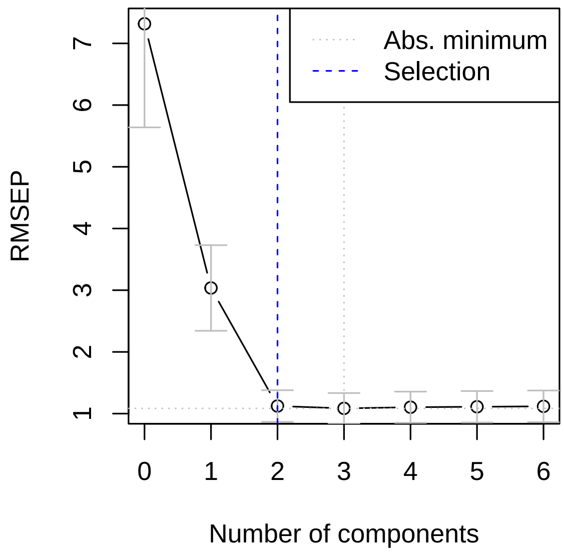
```r
par(mfrow = c(1,2))
plot(pls.fit, pch = 19, line = TRUE)
plot(pls.fit, plottype = "coef", comps = 1:3, legendpos = "bottomright")
```

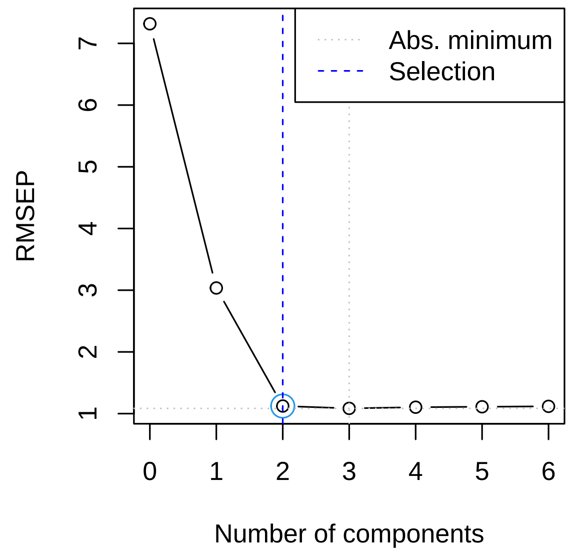



```r
plot(pls.fit, "loadings", comps = 1:3, legendpos = "topleft", ylim = c(-1, 1))
abline(h = 0)
```




```r
par(mfrow = c(1, 2))
ncomp1 <- selectNcomp(pls.fit, method = "onesigma", plot = TRUE)
ncomp2 <- selectNcomp(pls.fit, method = "randomization", plot = TRUE)
```





Then, we can choose 2 components to fit the model.

Based on the above results we have the following possible models:

1. By the PCA by correlation matrix: y ~ z1 + z2 + z3 + z4 + z5
2. By the stepwsie regression with p-values: y ~ x1 + x2
3. By the stepwsie regression with aic: y ~ x1 + x2 + x6
4. By the LASSO regression: y ~ x1 + x2 + x3
5. By the Partial least squares regression

We fit the three models and to predict the testing data. Note that we should evaluate the predict PC scores when using the PCA technique.

```
### Model 1
### PCA need
pca.test <- predict(pca, newdata = data.test[,1:6])
colnames(pca.test) <- c("z11", "z12", "z13", "z14", "z15", "z16")
#pca.test
can.fit1 <- lm(y~ z11+ z12 + z13 + z14 + z15, data = data.train)
pre.y1 <- predict(can.fit1, newdata = data.frame(pca.test))

### Model 2
can.fit2 <- lm(y~x1+x2, data = data.train)
pre.y2 <- predict(can.fit2, newdata = data.test)

### Model 3
can.fit3 <- lm(y~x1+x2+x6, data = data.train)
pre.y3 <- predict(can.fit3, newdata = data.test)

### Model 4
```

```r
can.fit4 <- lm(y~x1 + x2 +x3, data = data.train)
pre.y4 <- predict(can.fit4, newdata = data.test)

### Model 5: PLS regression
can.fit5 <- plsr(y ~ ., ncomp = 2, data = data.train)
pre.y5 <- predict(can.fit5, ncomp = 2, newdata = data.test)
```

Evaluate the mean squares error for the predictions.

$$MSEP = \frac{\sum_{i=1}^{m}(y_i - \hat{y}_i)^2}{m}.$$

```r
real.y <- data.test$y
e1 <- mean((real.y - pre.y1)^2)
e2 <- mean((real.y - pre.y2)^2)
e3 <- mean((real.y - pre.y3)^2)
e4 <- mean((real.y - pre.y4)^2)
e5 <- mean((real.y - pre.y5)^2)

comparison <- matrix(c(e1, e2, e3, e4, e5), nrow = 1)
rownames(comparison) <- "MSEP"
colnames(comparison) <- c("Model 1", "Model 2", "Model 3", "Model 4", "Model 5")  ### e2 is the smalles
round(comparison, 3)
```

```
##      Model 1 Model 2 Model 3 Model 4 Model 5
## MSEP   0.537   0.389    0.71   0.434   0.548
```

Hence, based on the simulation data and the model selection, I would select the model y~x1+x2, which is the same as the true model!