STA 206 Final Project: Abalone Age Prediction

Yehong Qiu[1]

[1]University of California, Davis

yehqiu@ucdavis.edu

Abstract. *An end-to-end data analysis on the Abalone dataset is conducted. From the original* $R^2 = 0.60$ *to the* $R^2 = 0.67$ *of the final model, a workflow of model selection, validation and diagnostics is done.*

1. Introduction

The Abalone dataset contains 8 physical measures and rings of 4177 abalones. It is assumed that the relation between the rate of ring deposition on the shell and age is invariant in the abalone population, and the domain knowledge shows that abalone's age can be calculated as the number of major rings present, plus 1.5 years to account for the period during which the minor rings are deposited.
Counting rings on abalone's shell is tedious. With an accurate and robust statistical model, can we estimate its age roughly with some easily-measurable physical characteristics?
Further, we might use the unsupervised model trained on these ageable abalone samples to guess the rough age of unageable abalone, of which the inability to age is because of either physical abrasion of the spire or extensive boring of the shell by other marine organisms, if the ageable and unageable abalone's rings both come from the same population (no survivor-bias).[1]

2. Methods and Results

2.1. Exploratory Data Analysis:

Among the physical features in the Abalone dataset, "Sex" is a categorical variable with 3 levels as "Female", "Male", and "Infant", and the other 7 are quantitative variables.

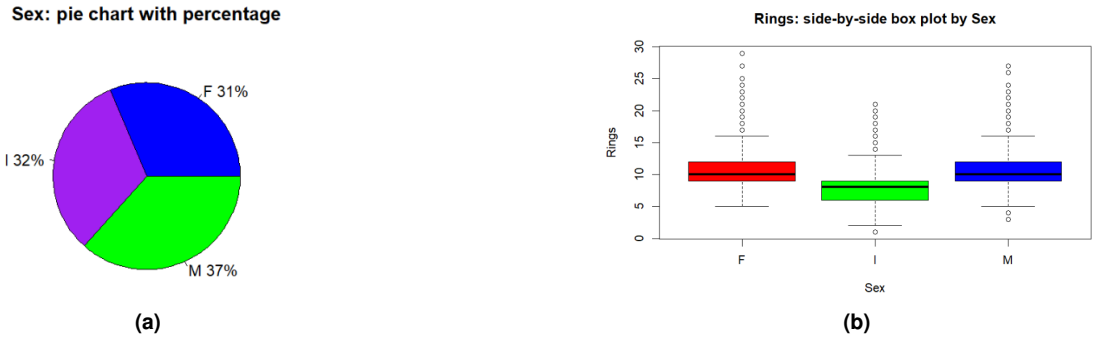
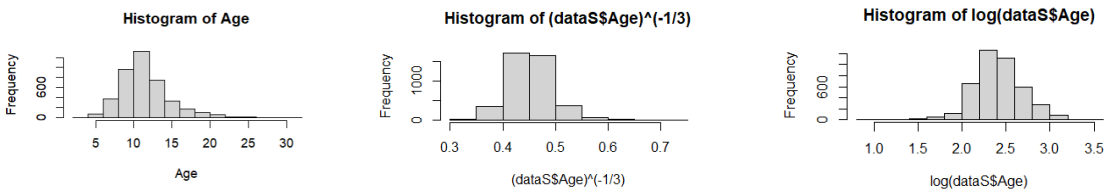


Figure 1. Categorical Variable: "Sex"

From the side-by-side boxplot, there are observations under the category "Infant" whose number of rings is 0, so for further convenience, the variable "Rings" are transformed

to "Age" by adding 1.5 to the original values. Besides, it is observed that abalone's age under the category "Infant" is smaller than that under "Female" and "Male".



(a) **(b)** $(Age)^{-1/3}$ **(c)** $log(Age)$

Figure 2. Distribution of "Age" and its transforms

From the subplot (a) in Figure 2, the distribution of abalone's age is right-skewed. Consider transform "Age" to its log or its inverse cubic root.

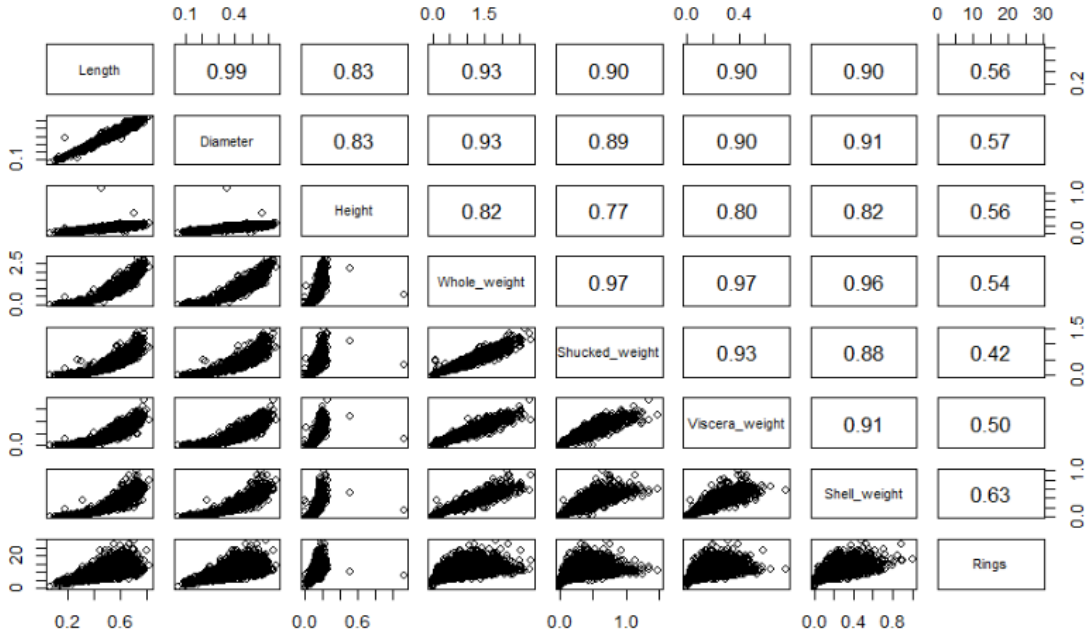


Figure 3. Correlation Coefficients and Pairwise Scatterplot

It's observed that there is strong correlation between any two of these variables. There are roughly linear relations among the three length-related variables "Length", "Diameter" and "Height", and among the four weight-related variables. But there are nonlinearity between length-related variables and weight-related variables. Besides, there are outliers which may be potential influential cases.

2.2. Preliminary Fitting:

First, without any transformation on "Age", a linear model regressing to all features is built. Based on the result of Box-cox transformation, transform "Age" to its inverse cubic root since the optimal $\lambda \sim -\frac{1}{3}$.

Next, five linear models are built. *Model 1* regresses $(Age)^{-\frac{1}{3}}$ to all the predictors, and *Model 2* deleted "Whole weight". *Model 3* uses the cubic root of "Shucked weight", "Viscera weight" and "Shell weight" when remaining "Whole weight" in the model and untransformed. *Model 4* is the same as the third, but deletes "Whole weight" from the model. *Model 5* uses the cubic root of all the four weight-related variables.
The reason behind building so many is that it is found that there exists a strong linear relationship between the four weight-related variables. What is more, the pairwise scatterplots of length-related variables and weight-related variables become linear after transforming the latter to their cubic root. With the second to fifth models, not only the nonlinearity within the X variables is alleviated, but it also relieves the strong colinearity among the 4 weight-related variables.

Model	p	R^2	Adjusted R^2	SSE	AIC	BIC	PRESSp
Model 1	10	0.6058	0.605	2.790551	-30518.5	-30455.13	2.876316
Model 2	9	0.599	0.5982	2.839135	-30448.41	-30391.37	2.925858
Model 3	10	0.6283	0.6275	2.631468	-30763.68	-30700.31	2.666614
Model 4	9	0.6156	0.6148	2.721753	-30624.77	-30567.74	2.74642
Model 5	10	0.6255	0.6247	2.651205	-30732.47	-30669.1	2.676058

Table 1. Summary of Linear Model Statistics

In conclusion, *Model 3* performs the best. In the following sections, use variables in *Model 3* for further analyses. With respect to multi-colinearity, *Model 3* alleviates multi-colinearity among X variables because the VIFs in *Model 3* are all within 10, while *Model 1* has one variable of which the VIF is larger than 10.

2.3. Model Selection:

Split the dataset into a training set (70% of the observations) and a validation set (30% of the observations).

First-order Model Selection

First, performing first-order model selection with the ‘regsubsets()‘ from the ‘leaps‘ library ends with 9 different sub-models each representing the best model with (predictor) sizes up to 9. Comparing the 9 models, there is one model achieving the best R_a^2, Cp, AIC and the second best BIC, in which the dummy variable "Sex M" (if a male abalone or not) and "Length" are excluded. It arrives at the same model by using a stepwise forward regression under the AIC criterion. Denote the small model with only 7 predictors as *Model fs1*.

Second-order Model Selection

Secondly, expand the dataset with quadratic and interaction terms based on all predictors except "Sex M" and "Length" so that the models built on the expanded dataset will not be too large. In result, the expanded dataset has 35 columns in total(7 first-order predictors + 6 quadratic terms + 21 interaction terms + 1 response variable). Then, perform second-order model selection using a forward stepwise regression, a forward, and a backward regression, all under the AIC criterion. Denote the three models derived using the above methods as *Model fs2.1*, *Model fs2.2*, and *Model fs2.3*, repectively.

2.4. Model Validation:

Internal Validation

The following table shows the four model's size p, R^2,R_a^2 and other metrics calculated on the training data. When calculating C_p, denote the model using all the data columns in the expanded dataset as the *full model*, and assume that the *full model* is correct.

Model	p	R^2	R_a^2	SSE	AIC	BIC	$PRESS_p$	C_p
fs1	8	0.6330	0.6321	1.8469	-21517	-21470	1.8611	429.67
fs2.1	16	0.6778	0.6762	1.6214	-21882	-21786	1.6482	38.39
fs2.2	22	0.6805	0.6782	1.6081	-21894	-21763	1.6452	26.20
fs2.3	23	0.6817	0.6792	1.6021	-21903	-21765	1.6359	17.52

Table 2. Summary of Linear Model Statistics on the training data

If considering R_a^2 and AIC, the best model is *Model fs2.3*. But if considering BIC, the best model is *Model fs2.1*. The in-sample biases of *Model fs2.1, fs2.2, fs2.3* are all moderate since $C_p \sim p$, while *Model fs2.2, fs2.3* are likely to overfit the training data due to their large sizes p.

External Validation

Fit the above four models on the validation data to see if the estimated parameters and their standard errors undergo significant changes. Calculate the model's mean square prediction error (MSPE) on the validation data, and then compare MSPE with $PRESS_p/n$ and SSE/n.

Model	MSPE	$PRESS_p/n$	SSE/n
fs1	6.5722e-4	6.3672e-4	6.3184e-4
fs2.1	5.6669e-4	5.6388e-4	5.5472e-4
fs2.2	1.7356e-3	5.6284e-4	5.5014e-4
fs2.3	2.1234e-3	5.5965e-4	5.4812e-4

Table 3. Summary of Prediction/Estimation Error of Linear Models

The MSPE of *Model fs2.2, fs2.3* are much larger than their respective $PRESS_p/n$ and SSE/n, which means there are excessive in-sample model variances. This is due to large multi-colinearity within the predictors.

In conclusion, *Model fs2.1* is the best model. Next, the analysis workflow splits into two parts. In one part, fit the entire dataset on *Model fs2.1* and use that as the final model, and in the other part, perform ridge regression and PCR on the expanded dataset derived before to relieve multi-colinearity.

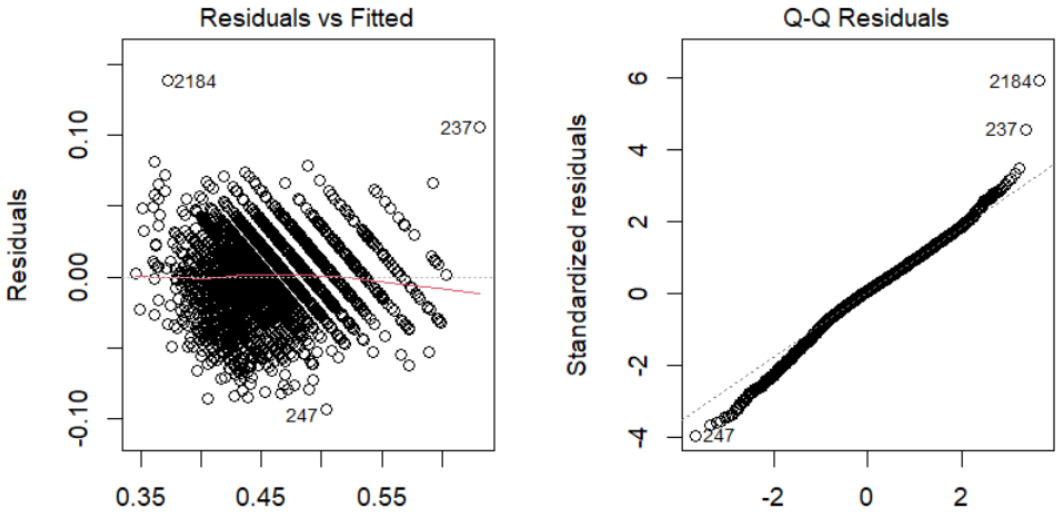
2.5. Model Diagnostics

The details for implementing Ridge regression and PCR are in Appendix 1. From the previous steps, two final models are generated. They are *Model fs2.1* and a PCR regression with 10 principle components. After fitting them on the entire dataset, their model metrics are as follows:

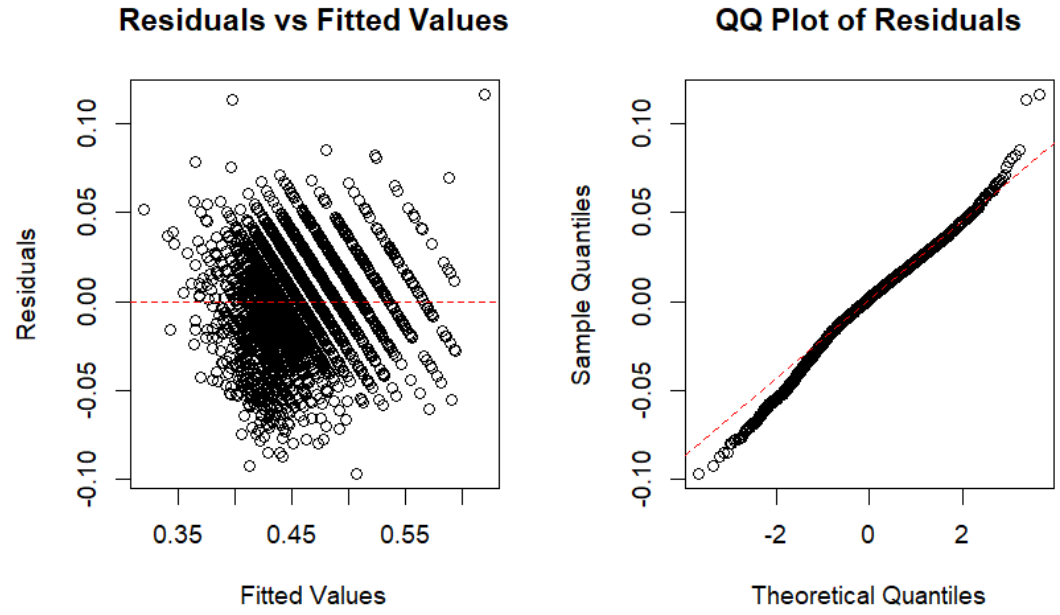
Model	p	R^2	R_a^2	SSE	AIC	BIC	$PRESS_p$	C_p
fs2.1	16	0.6739	0.6727	2.3090	-31298	-31196	2.3431	47.37
PCR	11	0.6524	0.6515	2.4612	-31041	-30971	2.4828	313.74

Table 4. Summary of Linear Model Statistics on the training data

From the residuals vs. fitted values plot and residuals QQ plot of these two models, there shows heavy-tailed residuals distribution.



(a) Model fs2.1



(b) PCR (ncomp=10)

Figure 4. Diagnostics: Residual Plots

Look into the X outliers and Y outliers of *Model fs2.1*, and then identify influencial cases

with cook's distance.

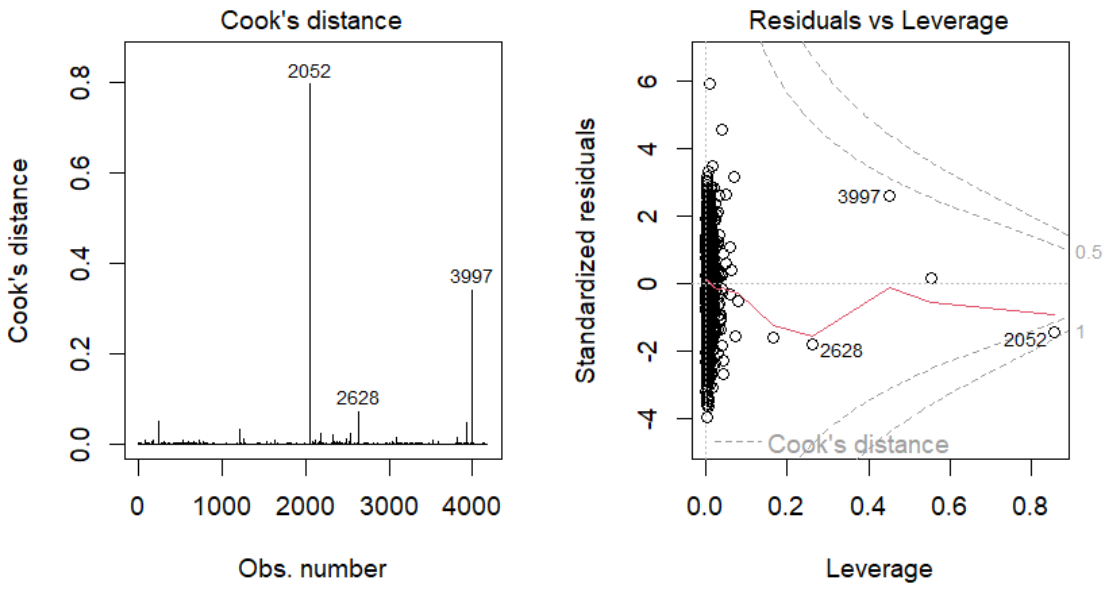


Figure 5. Outliers and Cook's Distance

After calculation, the average absolute percent difference in the fitted values with and without the most influential case 2052 is very small, only being 0.00067, indicating that it is unnecessary to delete it.

3. Conclusions and Discussion

In the data analysis on the Abalone dataset, an end-to-end statistical modeling process is performed. From the original $R^2 = 0.60$ to the $R^2 = 0.67$, the workflow does not simply equal to adding X variables, since other criterion should be considered. Problems remains and I wonder if ML models such as Random Forest, Support Vector Machine, etc. can have better performance.

4. Appendices

4.1. Appendix 1: Ridge Regression and PCR

Model	RMSPEt	RMSPEv
OLS	0.02339	0.04474
Ridge	0.02341	0.04308
PCR	0.02430	0.02420

Table 5. RMSPE bewteen the OLS full model, the Ridge and the PCR

Ridge Regression

The implementation of ridge regression does not help much in relieving multi-colinearity. After choosing an optimal λ based on the generalized cross-validation criterion (GCV),

the root mean square prediction error (RMSPE) on the validation set is only slightly smaller than that of the ordinary least square model (See table 4). The effect of Ridge is so trivial because the effective number of its smoothing matrix is 27.99, still close to the number of predictors, which is 34. Moreover, although after the ridge regression, the variances of the 34 estimated regression coefficients shrink a little, all of them do not shrink more than by 10 times (See Figure 5).

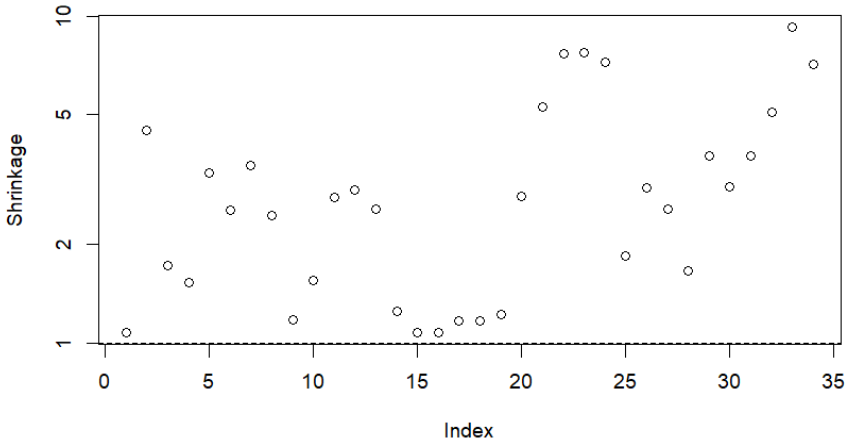


Figure 6. Ratio of Variance Between OLS Estimators and Ridge estimators

PCR

After performing PCA on the predictors in the expanded dataset, it is found that $ncomp = 10$ is an elbow. If ploting the "PCR CV Plot", letting the number of principle components be 10 is also acceptable (See Figure 6).

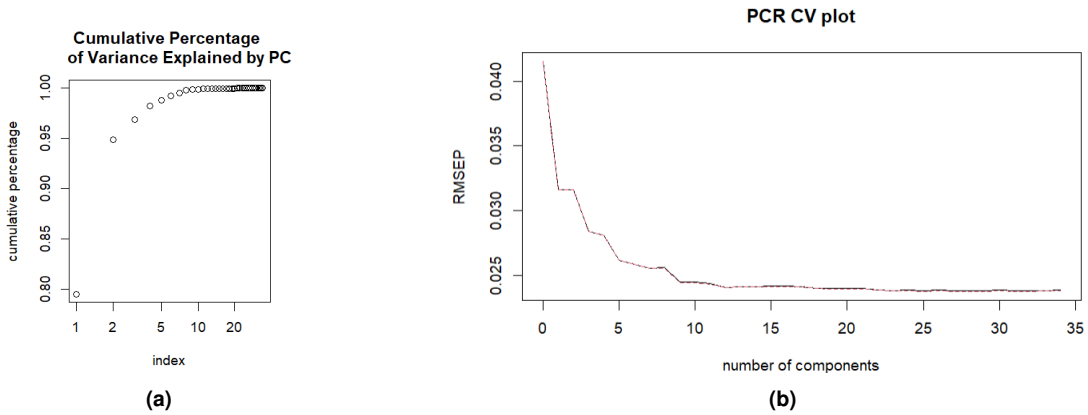


Figure 7. PCA and PCR

The difference between the RMSPE of PCR on the training set and the validation set is small compared to that of OLS and Ridge, which means PCR solves the issue of data overfitting.

4.2. Appendix 2: Coding

Please see the pdf file after the References

References

[1] Warwick Nash et al. “The Population Biology of Abalone (Haliotis species) in Tasmania. I. Blacklip Abalone (H. rubra) from the North Coast and Islands of Bass Strait”. In: *Sea Fisheries Division, Technical Report No* 48 (Jan. 1994).

Higher order model selection

Yehong Qiu

2024-12-09

Data Pre-processing

Load the Data

```
data = read.table("abalone.txt", header=FALSE, sep = ",")
names(data) = c("Sex","Length","Diameter","Height","Whole_weight",
                "Shucked_weight","Viscera_weight","Shell_weight","Rings")
```

Data Transformation

```
# Convert Categorical Variables to Factors
data$Sex<- as.factor(data$Sex)

# Create Feature AGE as response data
data$Age = data$Rings + 1.5
```

```
dataS = data[,1:8]
dataS$Age = data$Age
```

```
# Age being its inverse cubic root; Weight variables being their cubic root
newdata = dataS
newdata$Age = (dataS$Age)^(-1/3)
newdata$Shucked_weight = (newdata$Shucked_weight)^(1/3)
newdata$Viscera_weight = (newdata$Viscera_weight)^(1/3)
newdata$Shell_weight = (newdata$Shell_weight)^(1/3)
```

Fit the Model.1.1

```
model.1.1 = lm(Age~.,data=newdata)
```

Calculate SSEp,Square_Rp,Square_Rap,AICp,BICp, and Pressp (model.1.1)

Compare with model.0.1-3, model.1.1 is better

```
n=nrow(newdata)
p_full=length(model.1.1$coefficients)
sse_full=sum(model.1.1$residuals^2)
aic_full=n*log(sse_full/n)+2*p_full
bic_full = n*log(sse_full/n)+log(n)*p_full
PRESS_p_full=sum((model.1.1$residuals/(1-influence(model.1.1)$hat))^2)

p_full
```

```
## [1] 10
```

```
sse_full
```

```
## [1] 2.631468
```

```
aic_full
```

```
## [1] -30763.68
```

```
bic_full
```

```
## [1] -30700.31
```

```
PRESS_p_full # not much larger than sse, not overfitting
```

```
## [1] 2.666614
```

Calculate VIF

strong multi-colinearity

```
# VIF
library(car)
```

```
## Warning: 程序包'car'是用R版本4.4.2 来建造的
```

```
## 载入需要的程序包：carData
```

```
## Warning: 程序包'carData'是用R版本4.4.2 来建造的
```

```
vif_values <- vif(model.1.1)

vif_values
```

```
##                     GVIF Df GVIF^(1/(2*Df))
## Sex             1.598758  2        1.124464
## Length         48.890051  1        6.992142
## Diameter       46.069871  1        6.787479
## Height          3.789780  1        1.946736
## Whole_weight   10.981291  1        3.313803
## Shucked_weight 20.662026  1        4.545550
## Viscera_weight 20.043235  1        4.476967
## Shell_weight   19.010328  1        4.360084
```

Model Selection

data splitting

```
set.seed(123)
idx = sample(1:nrow(newdata), size=0.7*nrow(newdata), replace=FALSE)

data_t = newdata[idx,]
data_v = newdata[-idx,]


library(fastDummies)
```

```
## Warning: 程序包'fastDummies'是用R版本4.4.2 来建造的
```

```
data_t_with_dummies <- dummy_cols(data_t, select_columns = "Sex", remove_first_dummy = TRUE)
data_t_with_dummies$Sex <- NULL
data_t_with_dummies <- data_t_with_dummies[, c(9, 10, 1:8)]

data_v_with_dummies <- dummy_cols(data_v, select_columns = "Sex", remove_first_dummy = TRUE)
data_v_with_dummies$Sex <- NULL
data_v_with_dummies <- data_v_with_dummies[, c(9, 10, 1:8)]
```

criterion calculation function

```
cri_stat <- function(model, ndata, flag) {
  cri_sta_c = 0
  n = ndata
  mod_sum = summary(model)
  if (flag == 1) {
    p_m = as.integer(as.numeric(rownames(mod_sum$which))+1)
    sse = mod_sum$rss
    aic = n*log(sse/n)+2*p_m
    bic = n*log(sse/n)+log(n)*p_m
    cri_stat_c = cbind(mod_sum$which, p_m, sse, mod_sum$rsq, mod_sum$adjr2, aic, bic, mod_sum$cp)
  }
  else {
    p_m = length(coef(model))
    residuals <- resid(model)
    sse <- sum(residuals^2)
    tss <- sum((model$model[[1]] - mean(model$model[[1]]))^2)
    rsq <- 1 - (sse / tss)
    adjr2 <- 1 - ((1 - rsq) * (n - 1) / (n - p_m))
    aic = n*log(sse/n)+2*p_m
    bic = n*log(sse/n)+log(n)*p_m
    PRESSp = sum((residuals / (1 - hatvalues(model)))^2)
    cri_stat_c = cbind(p_m, sse, rsq, adjr2, aic, bic, PRESSp)
  }

  return(cri_stat_c)
}
```

first-order effects

```
library(leaps)
```

```
## Warning: 程序包'leaps'是用R版本4.4.2 来建造的
```

```
sub_set = regsubsets(Age ~ ., data = data_t, nbest = 1, nvmax = 9, method = 'exhaustive', really.big=T)
n = nrow(data_t)
cri_sub = cri_stat(sub_set, n, flag=1)

fit0 = lm(Age~1, data=data_t)
fit0_cp = cri_stat(fit0, n, flag=0)[2]*(n-10)/cri_stat(model.1.1, nrow(model.1.1$model), flag=0)[2]-(n-2*10)
none = c(1, rep(0, 9), cri_stat(fit0, n, flag=0)[1:6], fit0_cp)

cri_sub = rbind(none, cri_sub)
colnames(cri_sub) = c(colnames(summary(sub_set)$which), "p_m", "sse", "R^2", "R^2_a", "aic", "bic", "Cp")
cri_sub
```

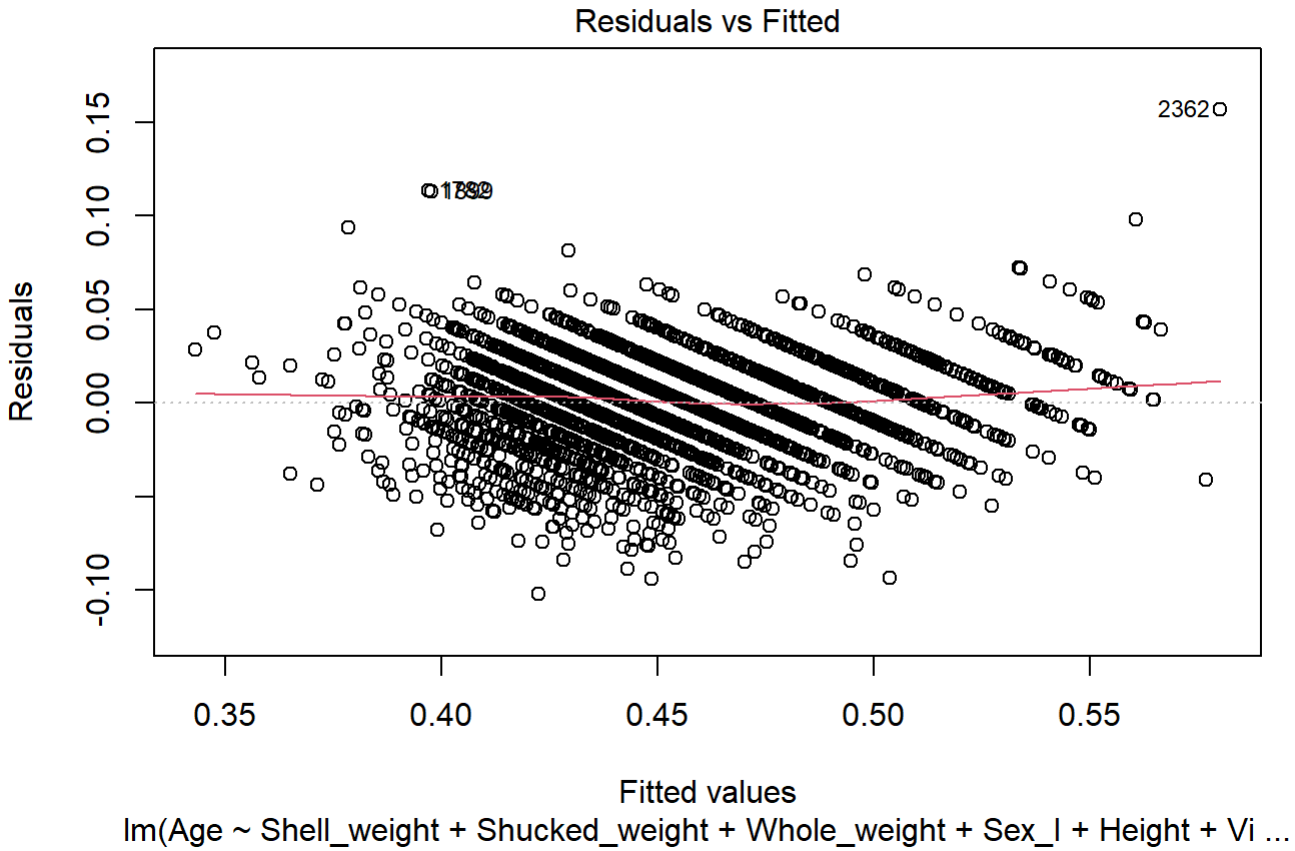
```
##      (Intercept) SexI SexM Length Diameter Height Whole_weight Shucked_weight
## none           1    0    0      0        0      0            0              0
## 1              1    0    0      0        0      0            0              0
## 2              1    0    0      0        0      0            0              1
## 3              1    0    0      0        0      0            1              1
## 4              1    1    0      0        0      0            1              1
## 5              1    1    0      0        0      1            1              1
## 6              1    1    0      0        0      1            1              1
## 7              1    1    0      0        1      1            1              1
## 8              1    1    1      0        1      1            1              1
## 9              1    1    1      1        1      1            1              1
##      Viscera_weight Shell_weight p_m      sse          R^2        R^2_a
## none              0            0   1 5.032679 4.440892e-15 4.551914e-15
## 1                 0            1   2 2.288957 5.451812e-01 5.450255e-01
## 2                 0            1   3 1.997554 6.030833e-01 6.028114e-01
## 3                 0            1   4 1.941820 6.141577e-01 6.137611e-01
## 4                 0            1   5 1.896313 6.232000e-01 6.226835e-01
## 5                 0            1   6 1.860755 6.302655e-01 6.296317e-01
## 6                 1            1   7 1.850175 6.323678e-01 6.316113e-01
## 7                 1            1   8 1.846874 6.330237e-01 6.321424e-01
## 8                 1            1   9 1.846371 6.331236e-01 6.321164e-01
## 9                 1            1  10 1.846085 6.331805e-01 6.320471e-01
##            aic       bic          Cp
## none -18601.18 -18595.20 2668.108406
## 1    -20902.08 -20890.12  692.822993
## 2    -21298.12 -21280.18  235.008626
## 3    -21378.83 -21354.91  149.064181
## 4    -21446.15 -21416.25   79.256786
## 5    -21499.48 -21463.60   25.148688
## 6    -21514.15 -21472.28   10.453726
## 7    -21517.36 -21469.52    7.244882
## 8    -21516.16 -21462.34    8.451424
## 9    -21514.61 -21454.81   10.000000
```

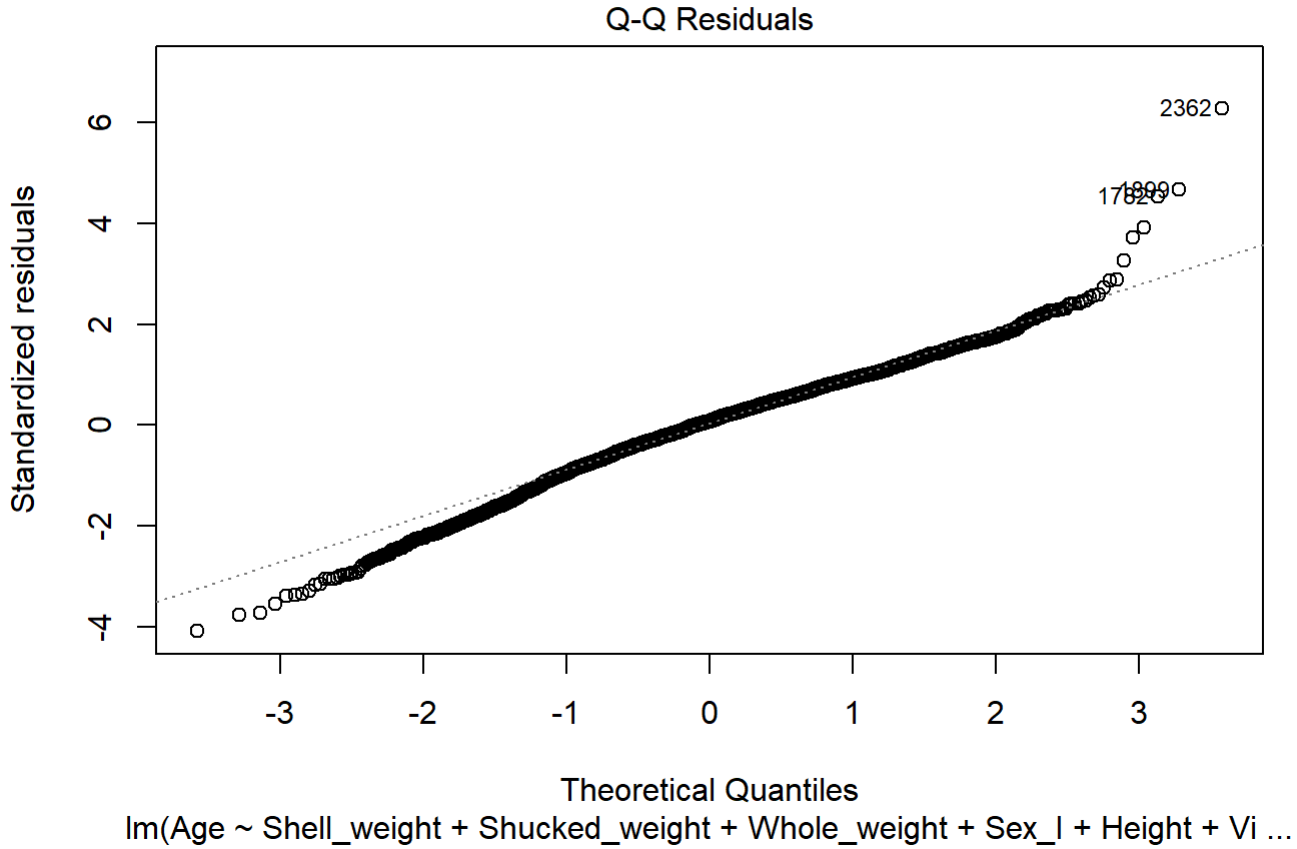
Best model: Based on SSE or R^2: model 9 Based on R^2_a, Cp, aic: model 7 Based on bic: model 6

```
library(MASS)
fit0=lm(Age~1, data=data_t_with_dummies)
fit1=lm(Age~., data=data_t_with_dummies)
model.1.2 = stepAIC(fit0, scope=list(upper=fit1, lower=~1), trace=0, direction='both', k=2)
model.1.2$anova
```

```
## Stepwise Model Path
## Analysis of Deviance Table
##
## Initial Model:
## Age ~ 1
##
## Final Model:
## Age ~ Shell_weight + Shucked_weight + Whole_weight + Sex_I +
##     Height + Viscera_weight + Diameter
##
##
##                Step Df    Deviance Resid. Df Resid. Dev       AIC
## 1                                       2922   5.032679 -18601.18
## 2   + Shell_weight  1 2.743721727      2921   2.288957 -20902.08
## 3 + Shucked_weight  1 0.291402813      2920   1.997554 -21298.12
## 4   + Whole_weight  1 0.055733923      2919   1.941820 -21378.83
## 5          + Sex_I  1 0.045507227      2918   1.896313 -21446.15
## 6         + Height  1 0.035557953      2917   1.860755 -21499.48
## 7 + Viscera_weight  1 0.010580268      2916   1.850175 -21514.15
## 8       + Diameter  1 0.003301053      2915   1.846874 -21517.36
```

```
plot(model.1.2, which=1:2)
```


Residuals vs Fitted
2362
1739
Residuals
0.15
0.10
0.05
0.00
-0.10
0.35
0.40
0.45
0.50
0.55
Fitted values
lm(Age ~ Shell_weight + Shucked_weight + Whole_weight + Sex_I + Height + Vi ...




```
cri_stat(model.1.2, nrow(model.1.2$model), flag=0)
```

```
##      p_m      sse       rsq     adjr2       aic       bic   PRESSp
## [1,]   8 1.846874 0.6330237 0.6321424 -21517.36 -21469.52 1.861138
```

Second-Order Model Selection

```
library(fastDummies)
newdata_with_dummies <- dummy_cols(newdata, select_columns = "Sex", remove_first_dummy = TRUE)
newdata_with_dummies$Sex <- NULL
newdata_with_dummies <- newdata_with_dummies[, c(9, 10, 1:8)]
newdata.epd <- newdata_with_dummies[,c(1,4:10)] #only choose variables in model.1.2
```

```
# second-order terms for quantitative variables
tmp = newdata.epd
n = ncol(newdata.epd)
for (i in 1:ncol(tmp[2:(n-1)])) {
  col_name <- names(tmp[2:(n-1)])[i]
  newdata.epd[[paste0(col_name, "2")]] <- tmp[[col_name]]^2
}

# interaction terms
for (i in 1:(ncol(tmp[1:(n-1)]) - 1)) {
  for (j in (i + 1):ncol(tmp[1:(n-1)])) {
    col_name_i <- names(tmp[1:(n-1)])[i]
    col_name_j <- names(tmp[1:(n-1)])[j]
    newdata.epd[[paste0(col_name_i, "x", col_name_j)]] <- tmp[1:(n-1)][[col_name_i]] * tmp[1:(n-1)][[col_name_j]]
  }
}
newdata.epd <- newdata.epd[, c(1:7, 9:35, 8)]
head(newdata.epd)
```

```
##   Sex_I Diameter Height Whole_weight Shucked_weight Viscera_weight Shell_weight
## 1     0    0.365  0.095       0.5140      0.6077693      0.4657010    0.5313293
## 2     0    0.265  0.090       0.2255      0.4633840      0.3646817    0.4121285
## 3     0    0.420  0.135       0.6770      0.6353735      0.5210973    0.5943922
## 4     0    0.365  0.125       0.5160      0.5995367      0.4848808    0.5371685
## 5     1    0.255  0.080       0.2050      0.4473090      0.3405642    0.3802952
## 6     1    0.300  0.095       0.3515      0.5204828      0.4263509    0.4932424
##   Diameter2  Height2 Whole_weight2 Shucked_weight2 Viscera_weight2
## 1  0.133225 0.009025    0.26419600       0.3693836       0.2168774
## 2  0.070225 0.008100    0.05085025       0.2147247       0.1329927
## 3  0.176400 0.018225    0.45832900       0.4036995       0.2715424
## 4  0.133225 0.015625    0.26625600       0.3594442       0.2351094
## 5  0.065025 0.006400    0.04202500       0.2000854       0.1159840
## 6  0.090000 0.009025    0.12355225       0.2709023       0.1817751
##   Shell_weight2 Sex_IxDiameter Sex_IxHeight Sex_IxWhole_weight
## 1     0.2823108          0.000        0.000              0.0000
## 2     0.1698499          0.000        0.000              0.0000
## 3     0.3533021          0.000        0.000              0.0000
## 4     0.2885500          0.000        0.000              0.0000
## 5     0.1446245          0.255        0.080              0.2050
## 6     0.2432881          0.300        0.095              0.3515
##   Sex_IxShucked_weight Sex_IxViscera_weight Sex_IxShell_weight DiameterxHeight
## 1            0.0000000            0.0000000          0.0000000        0.034675
## 2            0.0000000            0.0000000          0.0000000        0.023850
## 3            0.0000000            0.0000000          0.0000000        0.056700
## 4            0.0000000            0.0000000          0.0000000        0.045625
## 5            0.4473090            0.3405642          0.3802952        0.020400
## 6            0.5204828            0.4263509          0.4932424        0.028500
##   DiameterxWhole_weight DiameterxShucked_weight DiameterxViscera_weight
## 1             0.1876100              0.2218358              0.16998085
## 2             0.0597575              0.1227968              0.09664064
## 3             0.2843400              0.2668569              0.21886086
## 4             0.1883400              0.2188309              0.17698148
## 5             0.0522750              0.1140638              0.08684388
## 6             0.1054500              0.1561448              0.12790528
##   DiameterxShell_weight HeightxWhole_weight HeightxShucked_weight
## 1            0.19393519           0.0488300            0.05773809
## 2            0.10921406           0.0202950            0.04170456
```

```
## 3            0.24964472           0.0913950            0.08577543
## 4            0.19606652           0.0645000            0.07494208
## 5            0.09697529           0.0164000            0.03578472
## 6            0.14797272           0.0333925            0.04944586
##   HeightxViscera_weight HeightxShell_weight Whole_weightxShucked_weight
## 1            0.04424159          0.05047628                   0.31239344
## 2            0.03282135          0.03709157                   0.10449309
## 3            0.07034813          0.08024295                   0.43014788
## 4            0.06061009          0.06714607                   0.30936093
## 5            0.02724514          0.03042362                   0.09169835
## 6            0.04050334          0.04685803                   0.18294970
##   Whole_weightxViscera_weight Whole_weightxShell_weight
## 1                  0.23937029                0.27310325
## 2                  0.08223571                0.09293498
## 3                  0.35278286                0.40240352
## 4                  0.25019847                0.27717896
## 5                  0.06981567                0.07796053
## 6                  0.14986236                0.17337471
##   Shucked_weightxViscera_weight Shucked_weightxShell_weight
## 1                     0.2830388                   0.3229256
## 2                     0.1689876                   0.1909738
## 3                     0.3310914                   0.3776611
## 4                     0.2907038                   0.3220522
## 5                     0.1523375                   0.1701095
## 6                     0.2219083                   0.2567242
##   Viscera_weightxShell_weight       Age
## 1                   0.2474406 0.3928005
## 2                   0.1502957 0.4899973
## 3                   0.3097362 0.4566711
## 4                   0.2604627 0.4430309
## 5                   0.1295150 0.4899973
## 6                   0.2102944 0.4721632
```

```
data.epd_t = newdata.epd[idx,]
data.epd_v = newdata.epd[-idx,]
```

```
model.2 = lm(Age~., data=data.epd_t)
anova(model.2)["Residuals",1:3]
```

```
##             Df Sum Sq    Mean Sq
## Residuals 2888 1.5985 0.00055351
```

```
fit0=lm(Age~1, data=data.epd_t)
model.2.1 = stepAIC(fit0, scope=list(upper=model.2, lower=~1), trace=0, direction='both', k=2)
```

```
model.2.2 = stepAIC(fit0, scope=list(upper=model.2, lower=~1), trace=0, direction='forward', k=2)
```

```
model.2.3 = stepAIC(model.2, scope=list(upper=model.2, lower=~1), trace=0, direction='backward', k=2)
```

```
cri_stat(model.1.2, nrow(model.1.2$model), flag=0)
```

```
##      p_m      sse       rsq     adjr2       aic       bic   PRESSp
## [1,]   8 1.846874 0.6330237 0.6321424 -21517.36 -21469.52 1.861138
```

```
cri_stat(model.2.3, nrow(model.2.3$model), flag=0)
```

```
##      p_m      sse       rsq     adjr2       aic       bic  PRESSp
## [1,]  23 1.602141 0.6816524 0.6792374 -21902.88 -21765.33 1.63585
```

```
cri_stat(model.2.2, nrow(model.2.2$model), flag=0)
```

```
##      p_m      sse      rsq    adjr2       aic       bic   PRESSp
## [1,]  22 1.608052 0.680478 0.678165 -21894.12 -21762.55 1.645196
```

```
cri_stat(model.2.1, nrow(model.2.1$model), flag=0)
```

```
##      p_m      sse       rsq     adjr2       aic       bic  PRESSp
## [1,]  16 1.621443 0.6778171 0.6761547 -21881.88 -21786.19 1.64823
```

Model Validation: model.1.2 v.s. model.2.1, model.2.2, and model.2.3

```
n = nrow(data.epd_t)
# use model.2 as the full model
sigma_sq = anova(model.2)["Residuals", 3]
# calculate Cp
Cp= rep(0,4)
Cp[1] = anova(model.1.2)["Residuals",2]/sigma_sq - (n - 2*length(model.1.2$coefficients))
Cp[2] = anova(model.2.1)["Residuals",2]/sigma_sq - (n - 2*length(model.2.1$coefficients))
Cp[3] = anova(model.2.2)["Residuals",2]/sigma_sq - (n - 2*length(model.2.2$coefficients))
Cp[4] = anova(model.2.3)["Residuals",2]/sigma_sq - (n - 2*length(model.2.3$coefficients))
names(Cp) = c("model.1.2", "model.2.1", "model.2.2", "model.2.3")
p = c(length(model.1.2$coefficients), length(model.2.1$coefficients), length(model.2.2$coefficients), length(model.2.3$coefficients))
names(p) = c("model.1.2", "model.2.1", "model.2.2","model.2.3")

Cp
```

```
## model.1.2 model.2.1 model.2.2 model.2.3
## 429.67099  38.39413  26.20043  17.52243
```

```
p
```

```
## model.1.2 model.2.1 model.2.2 model.2.3
##         8        16        22        23
```

```
ext_vali <- function(mod_t, data_v) {
  n = ncol(data_v)
  mod_v = lm(mod_t, data = data_v)
  # print("summary on training data:")
  # print(summary(mod_t))
  # print("summary on validation data:")
  # print(summary(mod_v))

  del_pc_para = round(abs(coef(mod_t) - coef(mod_v))/abs(coef(mod_t))*100, 3)
  sd = summary(mod_t)$coefficients[,"Std. Error"]
  sd_v = summary(mod_v)$coefficients[,"Std. Error"]
  del_pc_err = round(abs(sd - sd_v) / sd *100, 3)

  mspe = rep(0, 3)
  pred = predict.lm(mod_t, data_v[, -n])
  mspe[1] = mean((pred - data_v[, n])^2)
  mspe[2] = cri_stat(mod_t, nrow(mod_t$model), flag=0)[7] / nrow(mod_t$model)
  mspe[3] = cri_stat(mod_t, nrow(mod_t$model), flag=0)[2] / nrow(mod_t$model)
  names(mspe) = c("mspe", "PRESSp/n", "sse/n")
  # print("relative percentage change in parameter estimates:")
  # print(del_pc_para)
  # print("relative percentage change in std err:")
  # print(del_pc_err)
  print(mspe)
}
```

```
ext_vali(model.1.2, data_v_with_dummies)
```

```
##         mspe     PRESSp/n        sse/n
## 0.0006572192 0.0006367220 0.0006318419
```

```
ext_vali(model.2.1, data.epd_v)
```

```
##         mspe     PRESSp/n        sse/n
## 0.0005668673 0.0005638829 0.0005547188
```

```
ext_vali(model.2.2, data.epd_v)
```

```
##         mspe     PRESSp/n        sse/n
## 0.0017356447 0.0005628449 0.0005501374
```

```
ext_vali(model.2.3, data.epd_v)
```

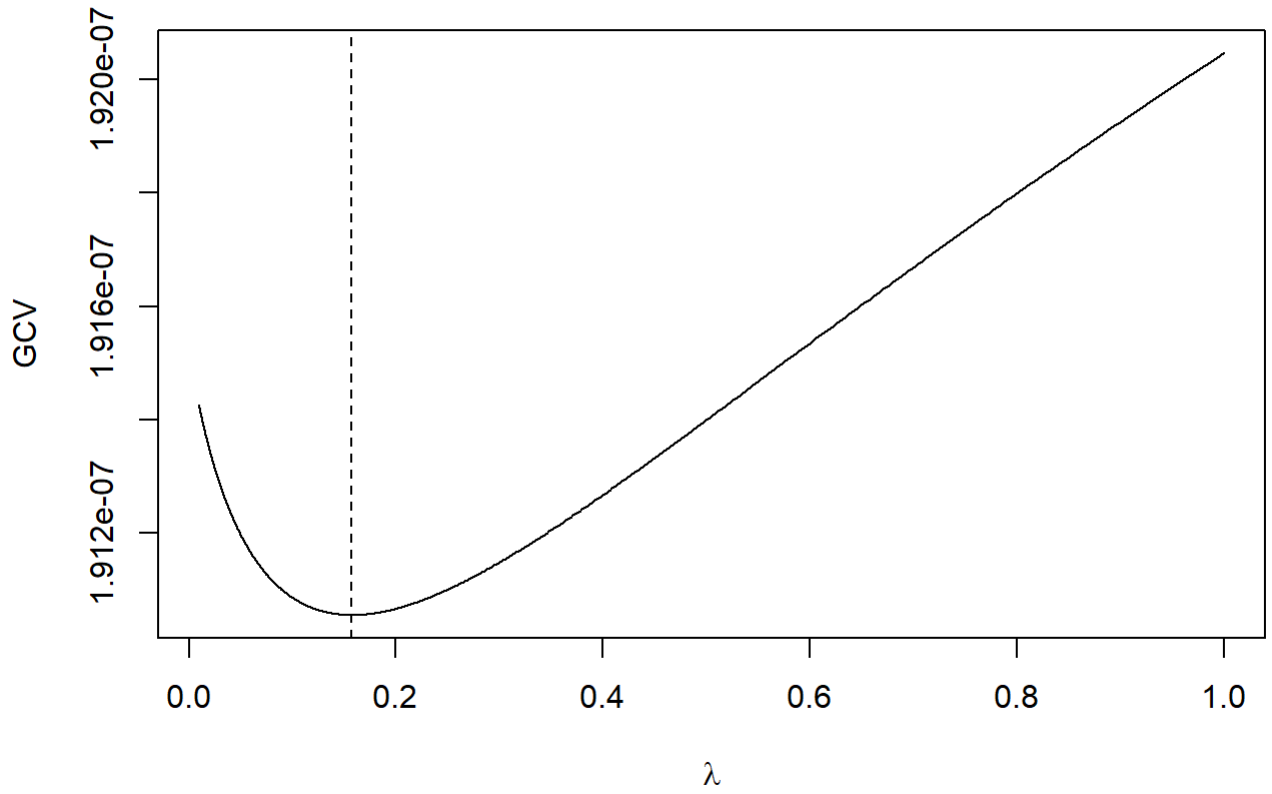
```
##         mspe     PRESSp/n        sse/n
## 0.0021233642 0.0005596476 0.0005481153
```

Ridge and PCR on Model.2

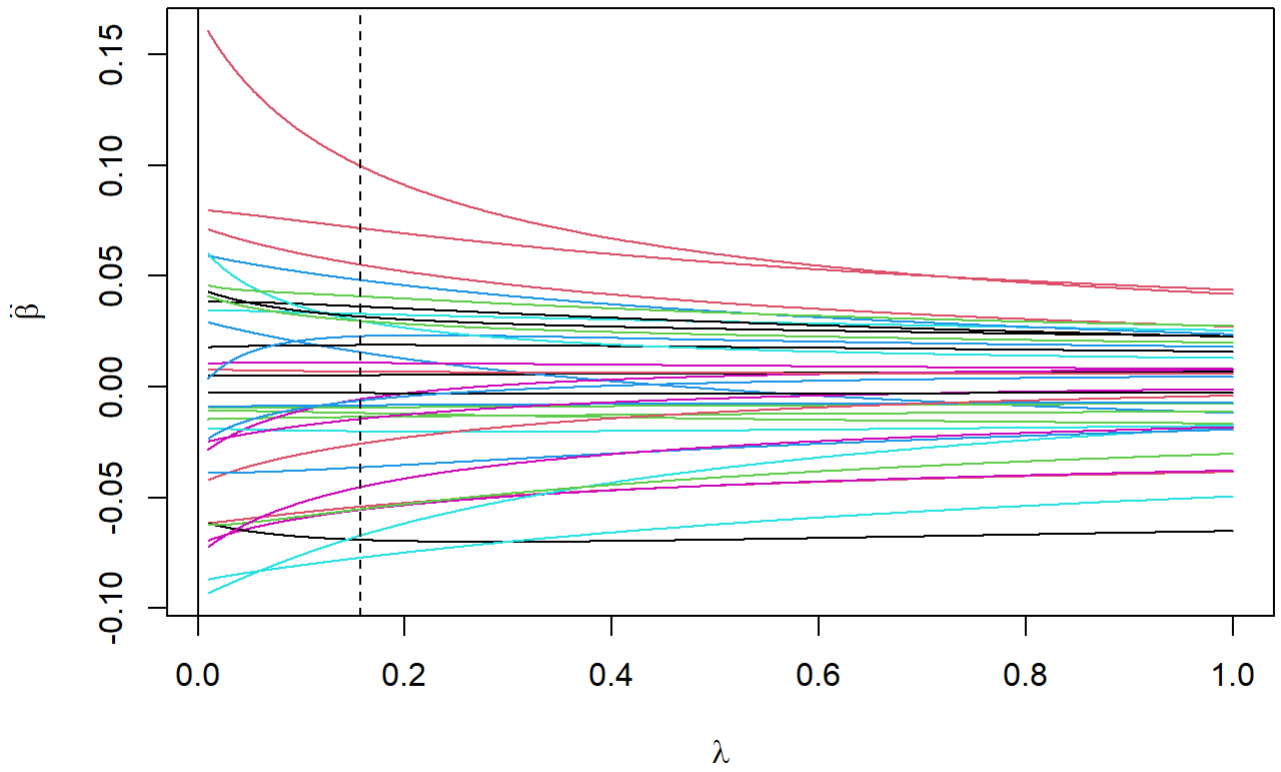
```
# Estimate RMSPE on ols
Age.vali.ols = predict(model.2, data.epd_v)

c(sqrt(mean((model.2$fitted.values - data.epd_t$Age)^2)), sqrt(mean((Age.vali.ols - data.epd_v$Age)^2)))
```

```
## [1] 0.02338547 0.04474333
```

```
model.2.ridge = lm.ridge(Age~., data=data.epd_t,
                  lambda = exp(seq(log(1e0),log(1e-2),length.out = 1000)))
lambda.GCV = model.2.ridge$lambda[which.min(model.2.ridge$GCV)]
plot(model.2.ridge$lambda, model.2.ridge$GCV, xlab = expression(lambda), ylab='GCV', type='l')
abline(v=lambda.GCV, lty=2)
```




```
matplot(model.2.ridge$lambda, t(model.2.ridge$coef), type='l', lty=1,
        xlab=expression(lambda), ylab=expression(hat(beta)))
abline(v=lambda.GCV, lty=2) ##GCV
abline(v=0, lty=1) ##OLS
```

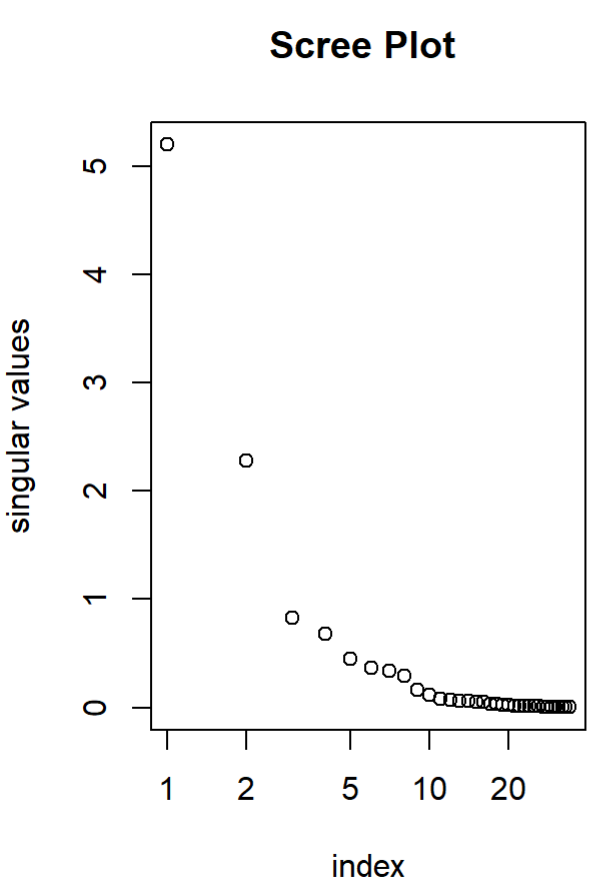


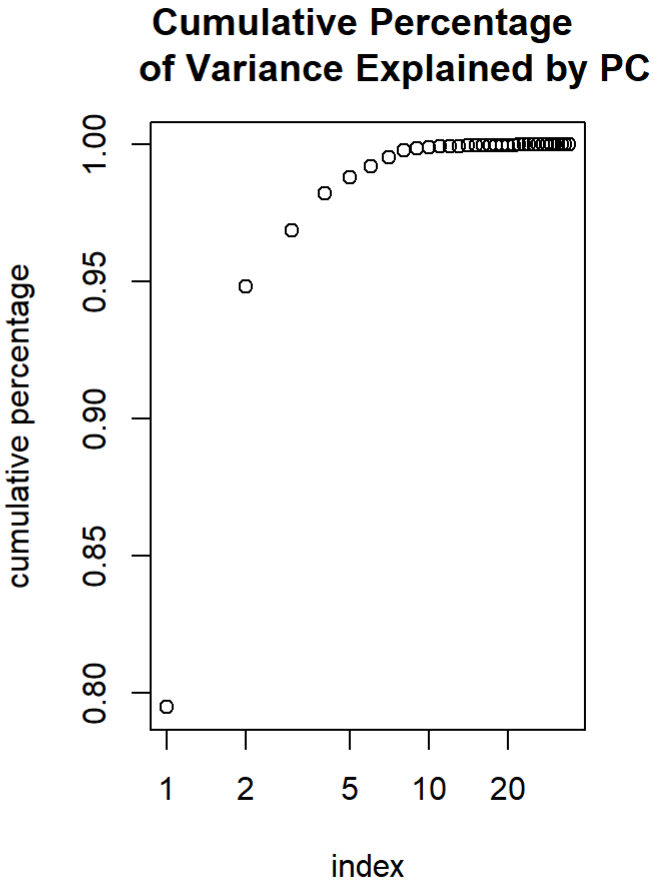

```
### RMSPE of `model.2.ridge`
ncol = 35
coef.GCV=model.2.ridge$coef[, which.min(model.2.ridge$GCV)]
X.train=scale(data.epd_t[,1:(ncol-1)], center=model.2.ridge$xm, scale=model.2.ridge$scales)
Yh.train=X.train%*%coef.GCV+model.2.ridge$ym
X.test=scale(data.epd_v[,1:(ncol-1)], center=model.2.ridge$xm, scale=model.2.ridge$scales)
Yh.test=X.test%*%coef.GCV+model.2.ridge$ym

c(sqrt(mean((Yh.train-data.epd_t$Age)^2)), sqrt(mean((Yh.test-data.epd_v$Age)^2)))
```

```
## [1] 0.02340525 0.04308126
```

Principle Component Analysis (PCA)

```
par(mfrow=c(1,2))
abapca<-prcomp(data.epd_t[,1:(ncol-1)], center=TRUE, scale=TRUE)
plot(abapca$sdev, type='p', xlab="index", ylab="singular values", log="x", main="Scree Plot ")
plot(cumsum((abapca$sdev)^2)/sum((abapca$sdev)^2), type='p', xlab="index", ylab="cumulative percentage", log="x", main="Cumulat
ive Percentage
     of Variance Explained by PC")
```

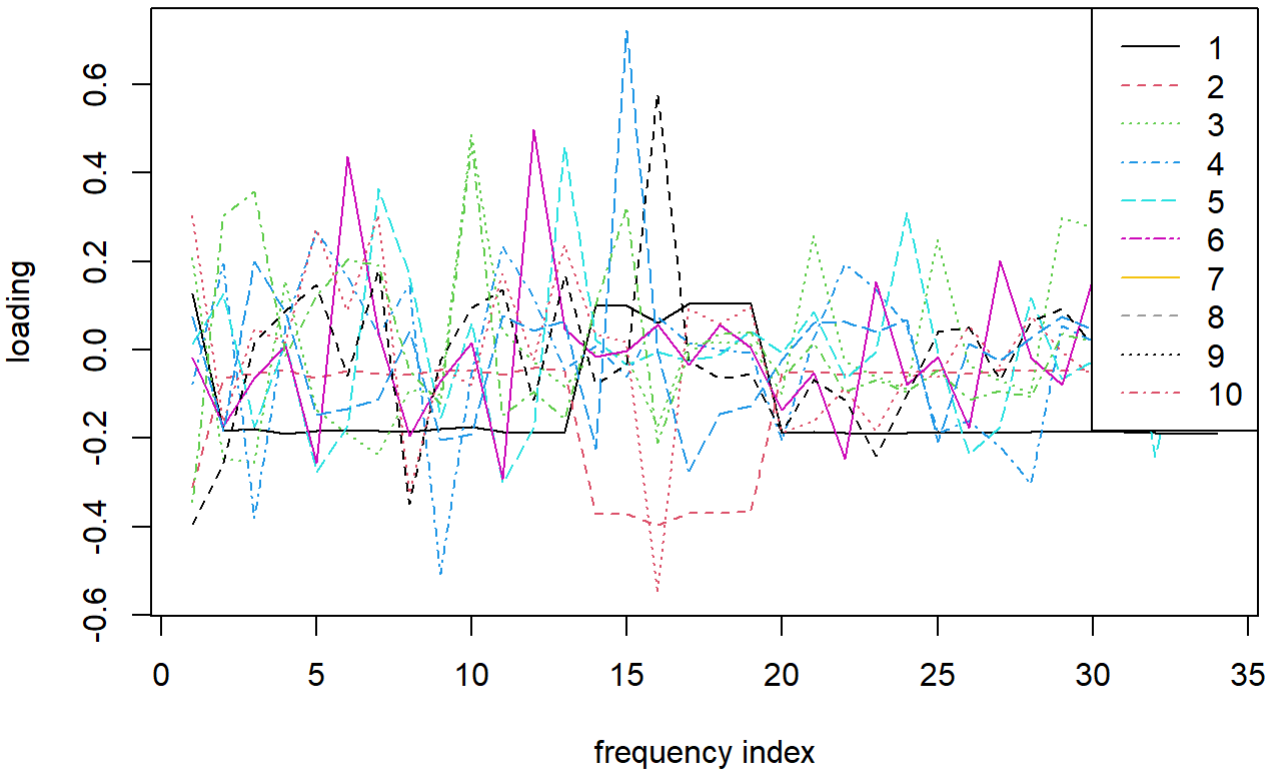




```
print(cumsum((abapca$sdev)^2)/sum((abapca$sdev)^2))
```

```
##  [1] 0.7948337 0.9483812 0.9687037 0.9822802 0.9882331 0.9921533 0.9955081
##  [8] 0.9980062 0.9987466 0.9991561 0.9993323 0.9994735 0.9995947 0.9996909
## [15] 0.9997753 0.9998451 0.9998817 0.9999135 0.9999366 0.9999504 0.9999601
## [22] 0.9999691 0.9999748 0.9999803 0.9999842 0.9999877 0.9999904 0.9999929
## [29] 0.9999948 0.9999965 0.9999979 0.9999989 0.9999995 1.0000000
```

```
par(mfrow=c(1,1))
#plot PC directions/loadings that generate the first five principal components against the X index (frequency index)
matplot(1:(ncol-1), abapca$rotation[,1:10], type='l', xlab="frequency index", ylab="loading")
legend('topright', legend=1:10, lty=1:10, col=1:10)
```



Principle Component Regression (PCR)

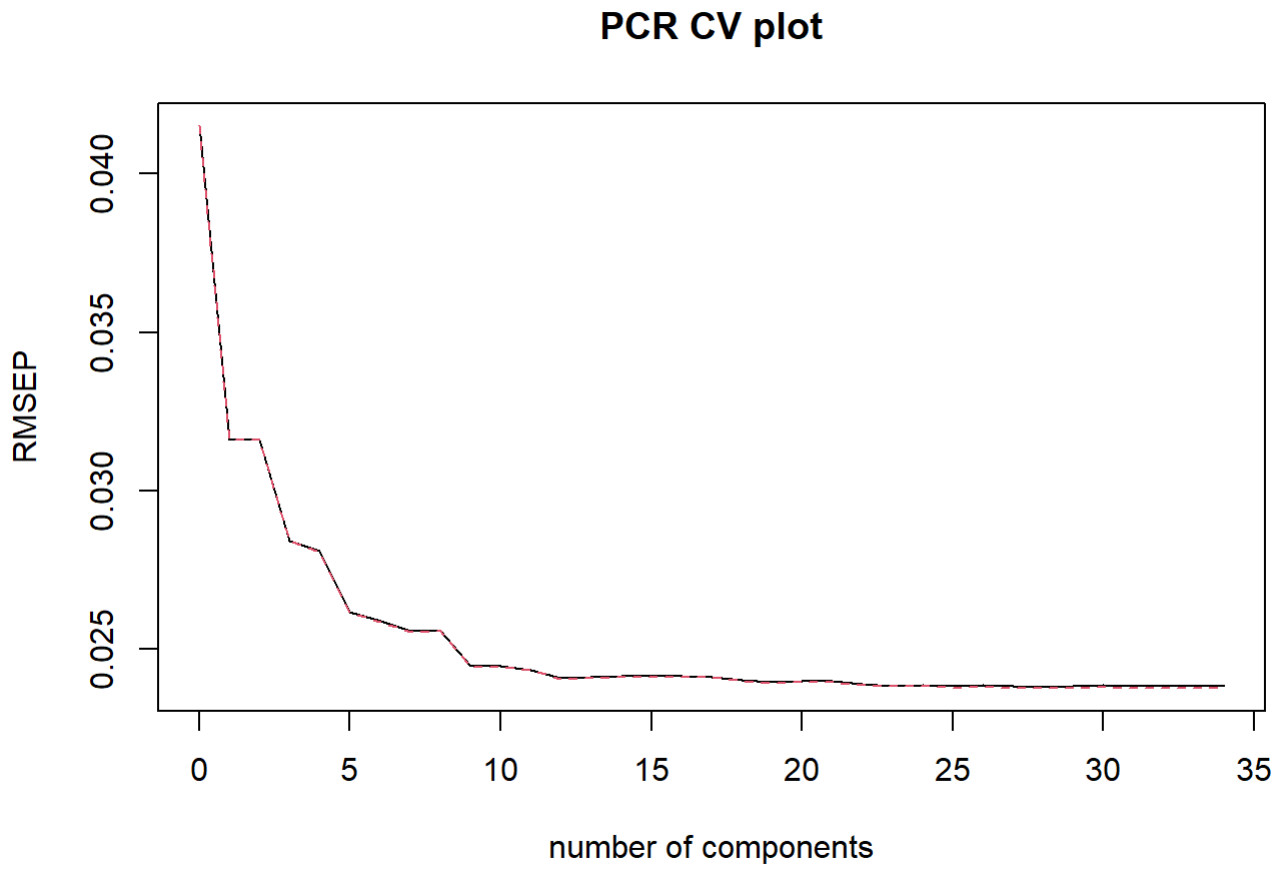
```
library(pls)  #pcr
```

```
## Warning: 程序包'pls'是用R版本4.4.2 来建造的
```

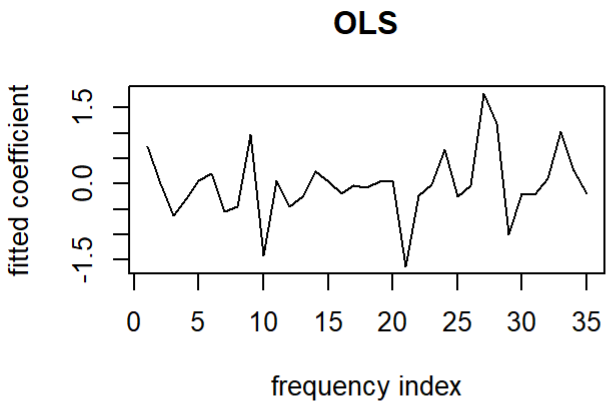
```
##
## 载入程序包：'pls'
```

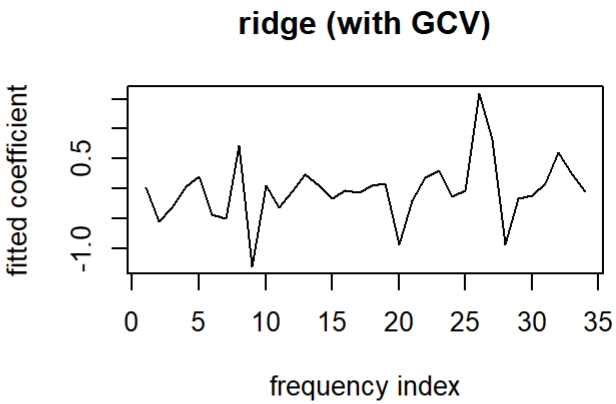
```
## The following object is masked from 'package:stats':
##
##     loadings
```

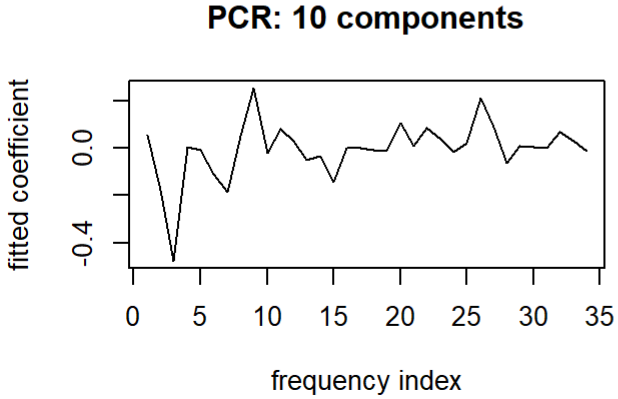
```
k.max=ncol-1
#fit models with 1,2,.., k.max components successively;
#standardize X and center Y; conduct "CV" (5 folds, consecutive segments)
#fitted values and coefficients (of the standardized X) of these k.max models are returned;
aba.pcr=pcr(Age~., data=data.epd_t, ncomp=k.max, center=TRUE, scale=TRUE,  validation="CV", segments = 5, segment.type = "conse
cutive")

validationplot(aba.pcr, main="PCR CV plot")
```




```
### plot fitted coefficients under the original scales
par(mfrow=c(2,2))
plot(coefficients(model.2), xlab="frequency index",ylab="fitted coefficient", main="OLS", type='l')
plot(model.2.ridge$coef[, which.min(model.2.ridge$GCV)]/model.2.ridge$scales, xlab="frequency index", ylab="fitted coefficien
t", main="ridge (with GCV)", type='l')

plot(coef(aba.pcr, ncomp = 10)/aba.pcr$scale,xlab="frequency index", ylab="fitted coefficient", main="PCR: 10 components", type
='l')
```







Comparison between OLS, Ridge, and PCR

```
rmspe<-function(y, yh) sqrt(mean((y-yh)^2))

##OLS
c(rmspe(data.epd_t$Age, model.2$fitted.values),rmspe(data.epd_v$Age, predict(model.2, data.epd_v)))
```

```
## [1] 0.02338547 0.04474333
```

```
##Ridge
coef.GCV=model.2.ridge$coef[, which.min(model.2.ridge$GCV)]
X.train=scale(data.epd_t[,1:(ncol-1)], center=model.2.ridge$xm, scale=model.2.ridge$scales)
Yh.train=X.train%*%coef.GCV+model.2.ridge$ym
X.test=scale(data.epd_v[,1:(ncol-1)], center=model.2.ridge$xm, scale=model.2.ridge$scales)
Yh.test=X.test%*%coef.GCV+model.2.ridge$ym
c(rmspe(data_t$Age, Yh.train), rmspe(data_v$Age, Yh.test))
```

```
## [1] 0.02340525 0.04308126
```

```
##PCR
c(rmspe(data.epd_t$Age, predict(aba.pcr, ncomp=10)), rmspe(data.epd_v$Age, predict(aba.pcr, data.epd_v, ncomp=10)))
```

```
## [1] 0.02430380 0.02419564
```

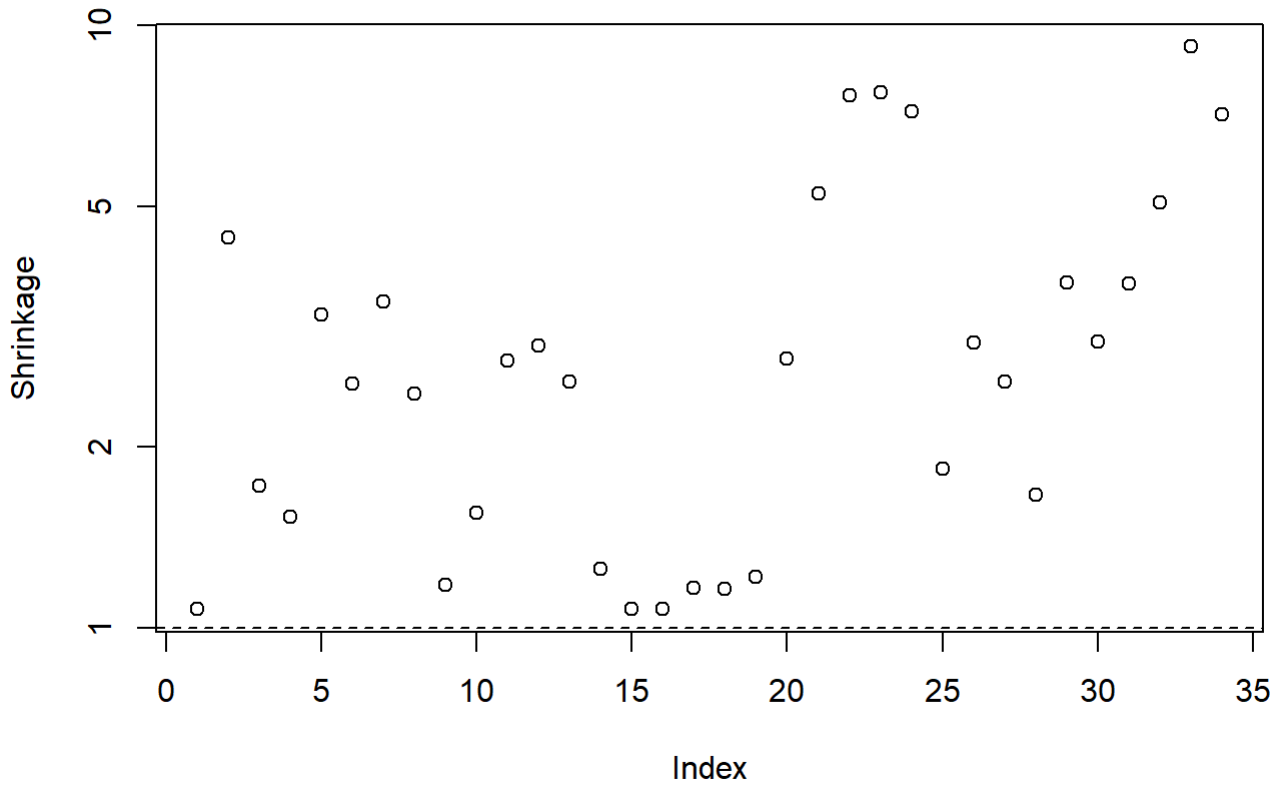
Why Ridge inefficient:

```
### the effective number of X variables in the ridge model with GCV selected lambda is:
lambda.GCV=model.2.ridge$lambda[which.min(model.2.ridge$GCV)]
S= X.train%*%solve(t(X.train)%*%X.train+lambda.GCV*diag(ncol-1), t(X.train)) ## the smoothing matrix
sum(diag(S)) ## effective number
```

```
## [1] 27.99324
```

The effective number (27.99) still close to 35, which means the penalization of ridge is very slight.

```
plot(diag(solve(t(X.train)%*%X.train))/diag(solve(t(X.train)%*%X.train+lambda.GCV*diag(ncol-1), t(X.train)%*%X.train)%*%solve(t
(X.train)%*%X.train+lambda.GCV*diag(ncol-1))), ylab="Shrinkage", log="y")
abline(h=1, lty=2)
```



Model Diagnositics

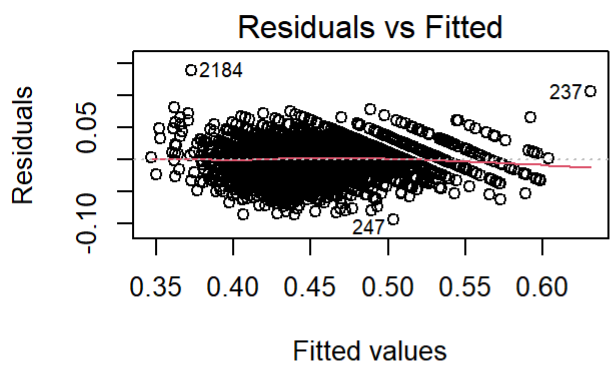
```
### model.2.1
model.2.1.final = lm(model.2.1, data=newdata.epd)
summary(model.2.1.final)
```

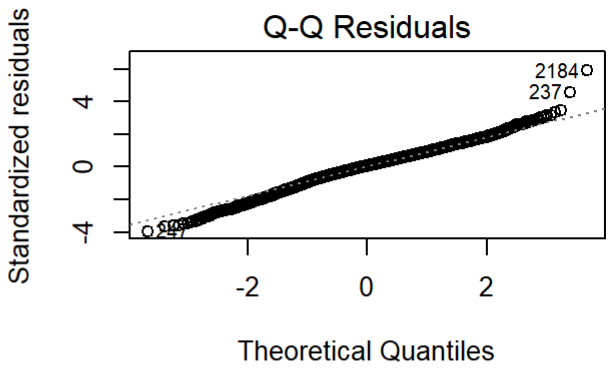
```
##
## Call:
## lm(formula = model.2.1, data = newdata.epd)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.093331 -0.013421  0.001511  0.015057  0.138562
##
## Coefficients:
##                              Estimate Std. Error t value Pr(>|t|)
## (Intercept)                  0.715332   0.006952 102.898  < 2e-16 ***
## Shell_weight                -0.894468   0.051664 -17.313  < 2e-16 ***
## Shucked_weightxShell_weight  0.908197   0.134702   6.742 1.77e-11 ***
## Sex_IxHeight                -0.137997   0.067170  -2.054  0.04000 *
## Height                      -0.437851   0.222670  -1.966  0.04932 *
## HeightxShucked_weight        1.500284   0.323817   4.633 3.71e-06 ***
## Sex_IxWhole_weight          -0.040195   0.004813  -8.351  < 2e-16 ***
## Whole_weightxShucked_weight -0.229464   0.025395  -9.036  < 2e-16 ***
## Whole_weight2                0.027596   0.005492   5.024 5.26e-07 ***
## Sex_IxShell_weight           0.193135   0.025397   7.605 3.51e-14 ***
## Sex_IxShucked_weight        -0.089854   0.018870  -4.762 1.99e-06 ***
## Whole_weightxViscera_weight  0.070614   0.012300   5.741 1.01e-08 ***
## HeightxWhole_weight         -0.106238   0.103031  -1.031  0.30254
## Shell_weight2                0.135712   0.082689   1.641  0.10082
## HeightxShell_weight         -1.048718   0.323953  -3.237  0.00122 **
## Shucked_weight2             -0.141614   0.058612  -2.416  0.01573 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.02356 on 4161 degrees of freedom
## Multiple R-squared:  0.6739, Adjusted R-squared:  0.6727
## F-statistic: 573.2 on 15 and 4161 DF,  p-value: < 2.2e-16
```

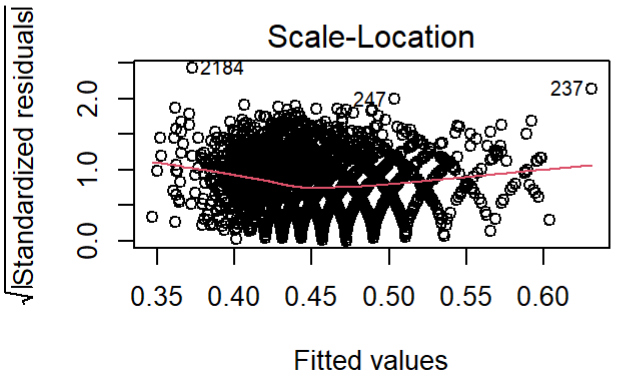
```
anova(model.2.1.final)
```

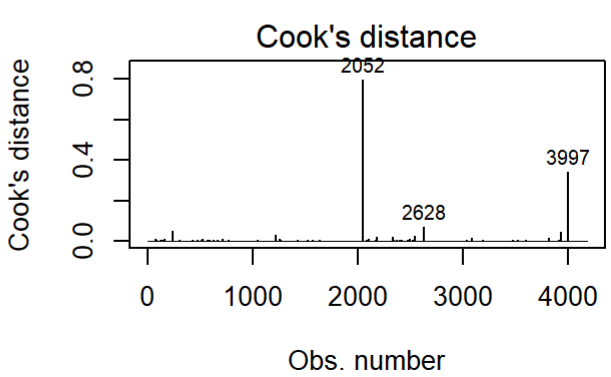
```
## Analysis of Variance Table
##
## Response: Age
##                                Df Sum Sq Mean Sq   F value    Pr(>F)
## Shell_weight                    1 3.8459  3.8459 6930.5740 < 2.2e-16 ***
## Shucked_weightxShell_weight     1 0.6113  0.6113 1101.5348 < 2.2e-16 ***
## Sex_IxHeight                    1 0.0408  0.0408   73.5188 < 2.2e-16 ***
## Height                          1 0.0194  0.0194   34.8874 3.772e-09 ***
## HeightxShucked_weight           1 0.0041  0.0041    7.4325 0.0064326 **
## Sex_IxWhole_weight              1 0.0156  0.0156   28.1411 1.186e-07 ***
## Whole_weightxShucked_weight     1 0.1192  0.1192  214.7679 < 2.2e-16 ***
## Whole_weight2                   1 0.0141  0.0141   25.4588 4.712e-07 ***
## Sex_IxShell_weight              1 0.0421  0.0421   75.8687 < 2.2e-16 ***
## Sex_IxShucked_weight            1 0.0216  0.0216   38.9456 4.793e-10 ***
## Whole_weightxViscera_weight     1 0.0205  0.0205   36.9737 1.305e-09 ***
## HeightxWhole_weight             1 0.0017  0.0017    3.0260 0.0820133 .
## Shell_weight2                   1 0.0067  0.0067   12.1590 0.0004936 ***
## HeightxShell_weight             1 0.0046  0.0046    8.2307 0.0041395 **
## Shucked_weight2                 1 0.0032  0.0032    5.8376 0.0157300 *
## Residuals                    4161 2.3090  0.0006
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
par(mfrow=c(2,2))
plot(model.2.1.final, which=1:5)
```



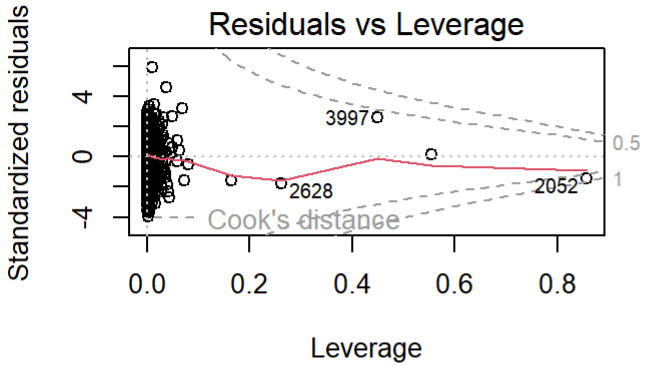







```
model.2.f = lm(model.2, data=newdata.epd)

sigma_sq = anova(model.2.f)["Residuals", 3]

cc = cri_stat(model.2.1.final, nrow(newdata.epd), flag=0)
Cp = cc[2] /sigma_sq - (nrow(newdata.epd) - 2*cc[1])
print(cbind(cri_stat(model.2.1.final, nrow(newdata.epd), flag=0), Cp))
```

```
##      p_m      sse       rsq     adjr2       aic       bic    PRESSp       Cp
## [1,]  16 2.308999 0.6738608 0.6726851 -31297.73 -31196.33 2.343078 47.37136
```




```
# check outliers in X, and Y
ns = nrow(newdata.epd)
res = residuals(model.2.1.final) # residuals of the final model
p = length(model.2.1.final$coefficients)
h1 = influence(model.2.1.final)$hat
d_res_std = studres(model.2.1.final) # studentized deleted residuals
qt(1-0.1/(2*ns), ns-1-p) # bonferronis thresh hold
```

```
## [1] 4.229351
```

```
# outliers in Y:
idx_Y =  as.vector(which(abs(d_res_std) >= qt(1-0.1/(2*ns), ns-1-p)))
```

```
# outliers in X:
idx_X = as.vector(which(h1 > (2*p/ns)))
```

```
per_average = function(model, dataset, idx){
  ns = nrow(dataset)
  model2 = lm(model, data = dataset[-idx,])
  f1 = fitted(model)
  f2 = fitted(model2)
  SUM = sum(abs((f1[-idx]-f2)/f1[-idx]))
  SUM = SUM + abs((f1[idx]- predict(model, newdata = dataset[idx,]))/f1[idx])
  return(SUM/ns)
}

per_average(model.2.1.final, newdata.epd, 2052)
```

```
##         2052
## 0.0006724596
```

```
per_average(model.2.1.final, newdata.epd, c(2052, 3997))
```

```
##         2052         3997
## 0.0009126611 0.0009126611
```

```
per_average(model.2.1.final, newdata.epd, c(2052, 2628))
```

```
##         2052         2628
## 0.0007767874 0.0007767874
```

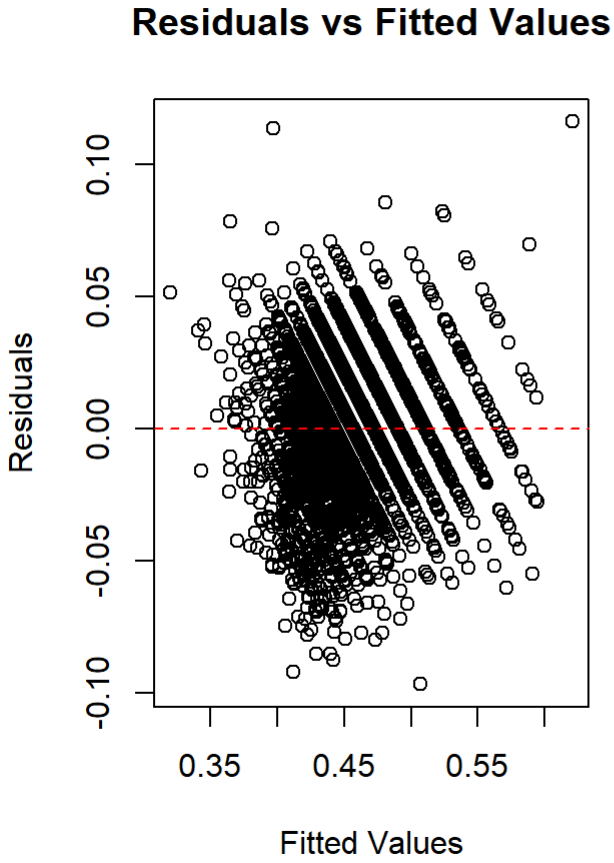
```
per_average(model.2.1.final, newdata.epd, c(3997, 2628))
```

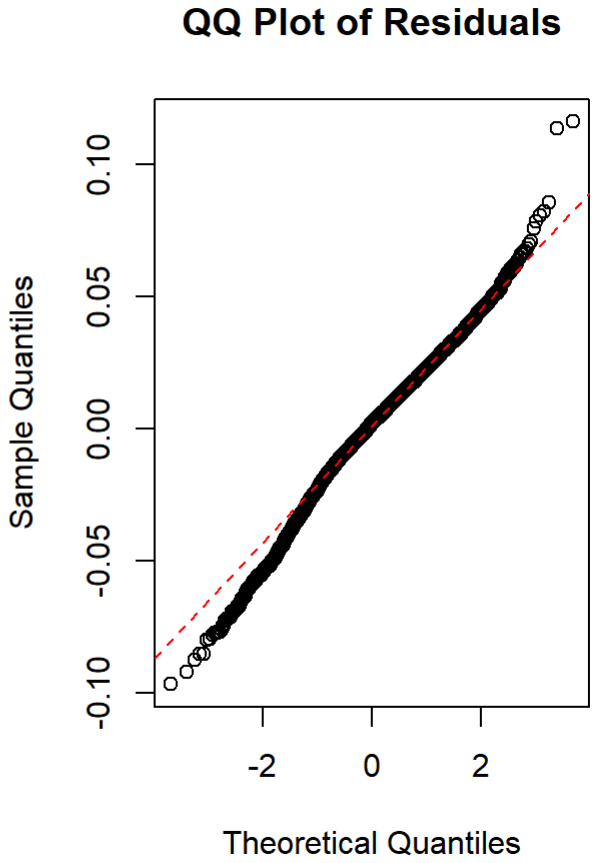
```
##         3997         2628
## 0.0008702895 0.0008702895
```

```
per_average(model.2.1.final, newdata.epd, c(2052, 3997, 2628))
```

```
##         2052         3997         2628
## 0.0009010491 0.0009010491 0.0009010491
```

```
### perform PCR with ncomp=10 on the entire dataset
library(pls)
set.seed(123)
pcr_model <- pcr(Age ~ ., data = newdata.epd, scale = TRUE, validation = "CV")

ncomp <- 10
pcr_pred <- predict(pcr_model, ncomp = ncomp)

# Residuals
residuals_pcr <- newdata.epd$Age - predict(pcr_model, ncomp = ncomp)
# Fitted values
fitted_pcr <- predict(pcr_model, ncomp = ncomp)

# Plot
par(mfrow=c(1,2))
plot(fitted_pcr, residuals_pcr,
     xlab = "Fitted Values",
     ylab = "Residuals",
     main = "Residuals vs Fitted Values")
abline(h = 0, col = "red", lty = 2)

qqnorm(residuals_pcr, main = "QQ Plot of Residuals")
qqline(residuals_pcr, col = "red", lty = 2)
```


Residuals vs Fitted Values
Residuals
0.10
0.05
0.00
-0.05
-0.10
0.35
0.45
0.55
Fitted Values


QQ Plot of Residuals
Sample Quantiles
0.10
0.05
0.00
-0.05
-0.10
-2
0
2
Theoretical Quantiles

```
SSE <- sum(residuals_pcr^2)
TSS <- sum((newdata.epd$Age - mean(newdata.epd$Age))^2)  # Total sum of squares
R2 <- 1 - (SSE / TSS)

# Adjusted R^2
n <- nrow(newdata.epd)  # Number of observations
p <- ncomp+1            # Number of components
adjR2 <- 1 - ((1 - R2) * (n - 1) / (n - p))

sigma2 <- SSE / n     # Residual variance estimate
AIC <- n * log(sigma2) + 2 * p
BIC <- n * log(sigma2) + log(n) * p
PRESSp <- sum(pcr_model$validation$PRESS[, ncomp])
# Cp
model.2.f = lm(model.2, data=newdata.epd)
sigma_sq = anova(model.2.f)["Residuals", 3]
Cp = SSE /sigma_sq - (n - 2*p)

cc = c(p, R2, adjR2, SSE, AIC, BIC, PRESSp, Cp)
names(cc) = c("p","R2", "adjR2", "SSE", "AIC", "BIC", "RRESSp", "Cp")
print(cc)
```

```
##             p            R2         adjR2           SSE           AIC
##  1.100000e+01  6.523610e-01  6.515265e-01  2.461213e+00 -3.104107e+04
##           BIC        RRESSp            Cp
## -3.097136e+04  2.482763e+00  3.137419e+02
```