浙江大学

课程名称: 回归分析

报告题目: King County 房屋销售价格案例分析

指导老师: 庞天晓

学　　院: 数学科学学院

专　　业: 数学与应用数学（金融学交叉创新平台）

姓名学号: 陈栩淦　3160104014

King County 房屋销售价格案例分析

——Final Project For Regression Analysis

陈栩淦 *3160104014*

数学科学学院

12/19/2018

【摘要】 本报告是 King County（包括 Seattle）的房屋销售价格的案例分析，来源于 Kaggle 的 Datasets，其中包括了 2014 年 5 月到 2015 年 5 月的房屋销售情况（价格数据和 18 个房屋特征），在分析过程中选取了其中 300 个样本点。报告首先对数据进行了回归诊断（包括模型、数据和多重共线性的诊断），其次分别进行了线性回归（变量选择)、主成分回归和岭回归分析，并进行预测。此外，本报告还拓展了其他内容，包括 resample 方法（Boostrap 和 Cross-Validation）和 Elastic net 方法。[1]

[1]注：为了减少文章篇幅，本报告 PDF 文件中的代码只展示了必要的代码，而结果是直接运行得到。如若要看完整代码，可以查看同文件夹中的.Rmd 或者.html 文件。此外，本报告中所有带有随机性的过程均设随机种子为 123.

目录

首先定义运行环境空间，并加载相关的 packages。

```
Sys.setlocale("LC_ALL", locale = "en_US.UTF-8")
setwd("~/Project/RegressionAnalysis")

library(reshape2)  # for melt
library(GGally)  # for ggpairs
library(car)  # for outlierTest
library(DAAG)  # for vif, eigen and kappa
library(leaps)  # for leaps and regsubsets
library(DAAG)  # for press
library(SignifReg)  # for SignifReg
library(boot)  # for lm.boot and cv.glm
library(MASS)  # for lm.ridge
library(glmnet) # for glmnet and cv.glmnet
library(ggplot2)
```

1 数据集

1.1 数据说明及来源

本报告所用数据为 King County（包括 Seattle）的房屋销售价格，其中包括了从 2014 年 5 月到 2015 年 5 月的房屋销售情况，总共有 21613 个样本点。其中数据来源为 Kaggle/Datasets/House Sales in King County, USA, 网址为 https://www.kaggle.com/harlfoxem/housesalesprediction。下面为网站上该数据集的描述。

- Overview

This dataset contains house sale prices for King County, which includes Seattle. It includes homes sold between May 2014 and May 2015.

It's a great dataset for evaluating simple regression models.

- about this file

19 house features plus the price and the id columns, along with 21613 observations.

1.1.1 数据信息

该数据包括了房屋 id、销售时间和销售价格的基本信息：

- id: a notation for a house
- date: Date house was sold
- price: Price is prediction target

以及 18 个房屋特征 features，如下：

- bedrooms: Number of Bedrooms/House
- bathrooms: Number of bathrooms/House
- sqft_living: square footage of the home
- sqft_lot: square footage of the lot
- floors: Total floors (levels) in house
- waterfront: House which has a view to a waterfront
- view: Has been viewed
- condition: How good the condition is (Overall)
- grade: overall grade given to the housing unit, based on King County grading system
- sqft_above: square footage of house apart from basement
- sqft_basement: square footage of the basement
- yr_built: Built Year
- yr_renovated: Year when house was renovated
- zipcode: zip
- lat: Latitude coordinate
- long: Longitude coordinate
- sqft_living15: Living room area in 2015(implies– some renovations) This might or might not have affected the lotsize area
- sqft_lot15: lotSize area in 2015(implies– some renovations)

1.2 读取数据

首先读取数据，该文件 “kc_house_data.csv” 为直接下载得到。下面查看数据结构。

```
str(data)
```

```
## 'data.frame':    21613 obs. of  20 variables:
##  $ id           : num  7.13e+09 6.41e+09 5.63e+09 2.49e+09 1.95e+09 ...
##  $ date         : Factor w/ 372 levels "20140502T000000",..: 165 221 291 221 284 11 57 252 340 306 .
##  $ price        : num  221900 538000 180000 604000 510000 ...
##  $ bedrooms     : int  3 3 2 4 3 4 3 3 3 3 ...
##  $ bathrooms    : num  1 2.25 1 3 2 4.5 2.25 1.5 1 2.5 ...
##  $ sqft_living  : int  1180 2570 770 1960 1680 5420 1715 1060 1780 1890 ...
```

```
##  $ sqft_lot     : int  5650 7242 10000 5000 8080 101930 6819 9711 7470 6560 ...
##  $ floors       : num  1 2 1 1 1 1 2 1 1 2 ...
##  $ waterfront   : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ view         : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ condition    : int  3 3 3 5 3 3 3 3 3 3 ...
##  $ grade        : int  7 7 6 7 8 11 7 7 7 7 ...
##  $ sqft_above   : int  1180 2170 770 1050 1680 3890 1715 1060 1050 1890 ...
##  $ yr_built     : int  1955 1951 1933 1965 1987 2001 1995 1963 1960 2003 ...
##  $ yr_renovated : int  0 1991 0 0 0 0 0 0 0 0 ...
##  $ zipcode      : int  98178 98125 98028 98136 98074 98053 98003 98198 98146 98038 ...
##  $ lat          : num  47.5 47.7 47.7 47.5 47.6 ...
##  $ long         : num  -122 -122 -122 -122 -122 ...
##  $ sqft_living15: int  1340 1690 2720 1360 1800 4760 2238 1650 1780 2390 ...
##  $ sqft_lot15   : int  5650 7639 8062 5000 7503 101930 6819 9711 8113 7570 ...
```

下面查看数据的描述性统计特征

```
summary(data)
```

```
##        id                          date           price
##  Min.   :1.000e+06   20140623T000000:  142   Min.   :  75000
##  1st Qu.:2.123e+09   20140625T000000:  131   1st Qu.: 321950
##  Median :3.905e+09   20140626T000000:  131   Median : 450000
##  Mean   :4.580e+09   20140708T000000:  127   Mean   : 540088
##  3rd Qu.:7.309e+09   20150427T000000:  126   3rd Qu.: 645000
##  Max.   :9.900e+09   20150325T000000:  123   Max.   :7700000
##                      (Other)        :20833
##     bedrooms        bathrooms      sqft_living       sqft_lot
##  Min.   : 0.000   Min.   :0.000   Min.   :  290   Min.   :    520
##  1st Qu.: 3.000   1st Qu.:1.750   1st Qu.: 1427   1st Qu.:   5040
##  Median : 3.000   Median :2.250   Median : 1910   Median :   7618
##  Mean   : 3.371   Mean   :2.115   Mean   : 2080   Mean   :  15107
##  3rd Qu.: 4.000   3rd Qu.:2.500   3rd Qu.: 2550   3rd Qu.:  10688
##  Max.   :33.000   Max.   :8.000   Max.   :13540   Max.   :1651359
##
##      floors        waterfront           view          condition
##  Min.   :1.000   Min.   :0.000000   Min.   :0.0000   Min.   :1.000
##  1st Qu.:1.000   1st Qu.:0.000000   1st Qu.:0.0000   1st Qu.:3.000
```

```
##  Median :1.500   Median :0.000000   Median :0.0000   Median :3.000
##  Mean   :1.494   Mean   :0.007542   Mean   :0.2343   Mean   :3.409
##  3rd Qu.:2.000   3rd Qu.:0.000000   3rd Qu.:0.0000   3rd Qu.:4.000
##  Max.   :3.500   Max.   :1.000000   Max.   :4.0000   Max.   :5.000
##
##      grade          sqft_above      yr_built     yr_renovated
##  Min.   : 1.000   Min.   : 290   Min.   :1900   Min.   :   0.0
##  1st Qu.: 7.000   1st Qu.:1190   1st Qu.:1951   1st Qu.:   0.0
##  Median : 7.000   Median :1560   Median :1975   Median :   0.0
##  Mean   : 7.657   Mean   :1788   Mean   :1971   Mean   :  84.4
##  3rd Qu.: 8.000   3rd Qu.:2210   3rd Qu.:1997   3rd Qu.:   0.0
##  Max.   :13.000   Max.   :9410   Max.   :2015   Max.   :2015.0
##
##     zipcode           lat             long        sqft_living15
##  Min.   :98001   Min.   :47.16   Min.   :-122.5   Min.   : 399
##  1st Qu.:98033   1st Qu.:47.47   1st Qu.:-122.3   1st Qu.:1490
##  Median :98065   Median :47.57   Median :-122.2   Median :1840
##  Mean   :98078   Mean   :47.56   Mean   :-122.2   Mean   :1987
##  3rd Qu.:98118   3rd Qu.:47.68   3rd Qu.:-122.1   3rd Qu.:2360
##  Max.   :98199   Max.   :47.78   Max.   :-121.3   Max.   :6210
##
##    sqft_lot15
##  Min.   :   651
##  1st Qu.:  5100
##  Median :  7620
##  Mean   : 12768
##  3rd Qu.: 10083
##  Max.   :871200
##
```

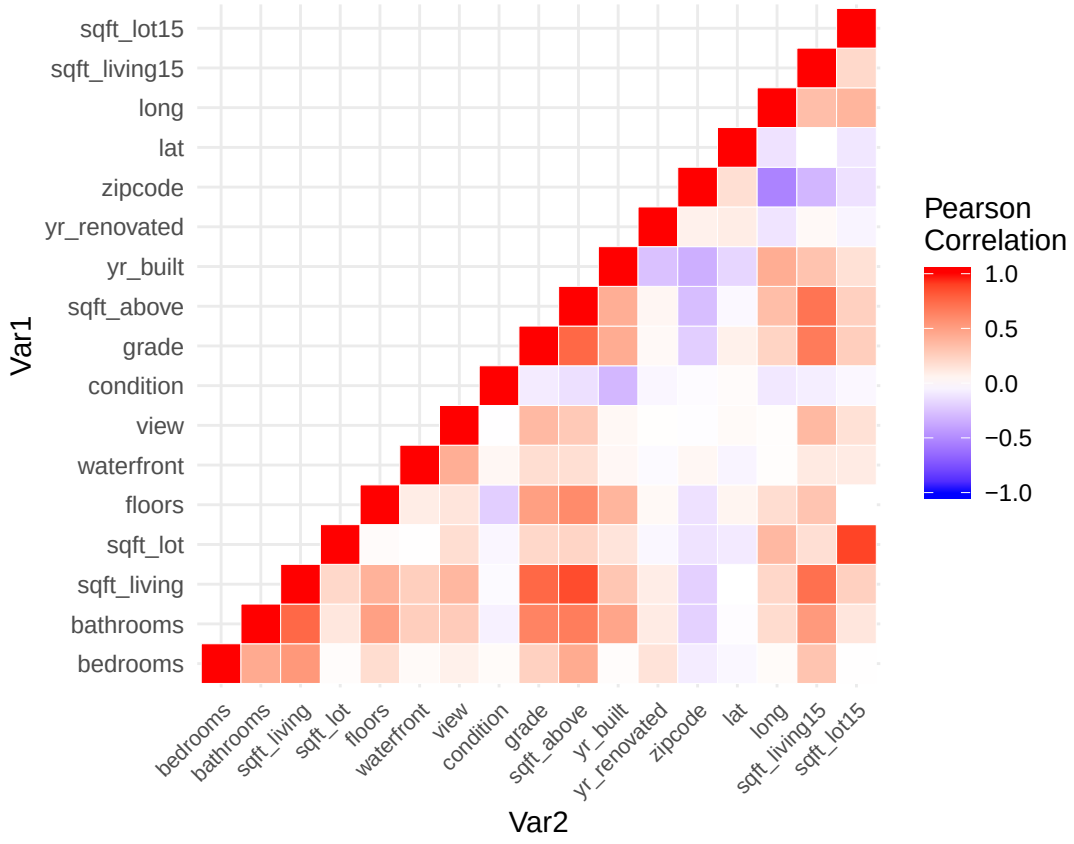
1.2.1 划分训练集和测试集

该数据集是从2014年5月2日到2015年5月27日的21613条数据，我们基于时间划分为训练集和测试集。其中训练集作为回归拟合的数据，测试集作为测试数据。我们选取其中2015年1月1日之前的数据作为训练集，2015年1月1日之后的数据作为测试集。并在训练集中随机抽取200个样本点、测试集中随机抽取100个样本点作为我们回归分析的数据。

```r
data.trans <- data[data$date < as.Date("20150101", format="%Y%m%d"), ]
data.test <- data[data$date >= as.Date("20150101", format="%Y%m%d"), ]

set.seed(123)

n.trans <- length(data.trans[, 1])
data.trans <- data.trans[sample(1:n.trans, 200), ]
rownames(data.trans) <- 1:200

n.test <- length(data.test[, 1])
data.test <- data.test[sample(1:n.test, 100), ]
rownames(data.test) <- 1:100

data <- rbind(data.trans, data.test)
```

1.3 数据特征

下面查看 18 个 features 之间的协方差关系。



2 案例分析

2.1 回归诊断

2.1.1 模型的诊断

首先进行下面模型的拟合和诊断：

$$Price = X\beta + \varepsilon$$

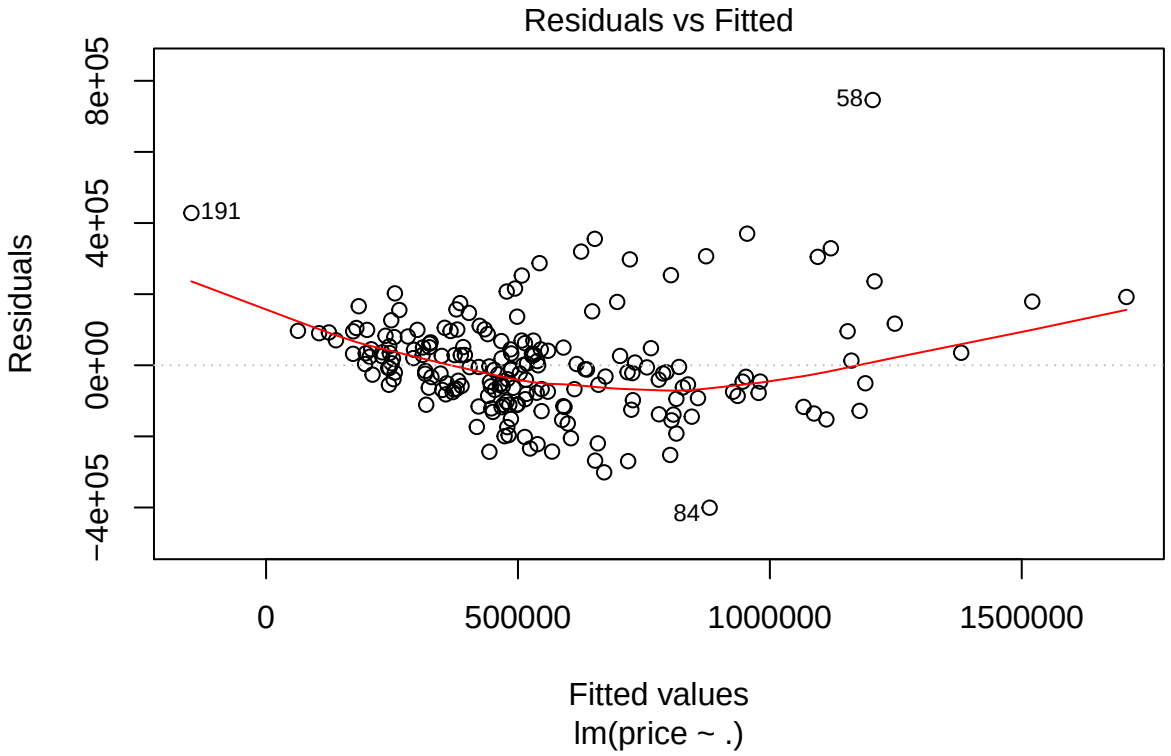
其中，β 包括了上述提到的 18 个特征。

```
lm.sol <- lm(price~., data = data.trans)
summary(lm.sol)
```

```
## Call:
## lm(formula = price ~ ., data = data.trans)

## Residuals:
##     Min      1Q  Median      3Q     Max
## -400467  -78374  -10380   62692  745889
##
## Coefficients:
##                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)   -3.648e+06  2.187e+07  -0.167  0.86768
## bedrooms      -3.528e+04  1.349e+04  -2.615  0.00968 **
## bathrooms      3.202e+04  2.547e+04   1.257  0.21044
## sqft_living    1.100e+02  3.235e+01   3.399  0.00083 ***
## sqft_lot      -1.165e+00  5.300e-01  -2.199  0.02916 *
## floors         6.905e+04  2.815e+04   2.453  0.01509 *
## waterfront     1.892e+05  1.265e+05   1.496  0.13644
## view           6.977e+04  1.494e+04   4.670 5.83e-06 ***
## condition      9.604e+03  1.717e+04   0.560  0.57649
## grade          8.160e+04  1.732e+04   4.713 4.84e-06 ***
## sqft_above     2.395e+01  3.526e+01   0.679  0.49786
## yr_built      -3.474e+03  5.439e+02  -6.387 1.38e-09 ***
## yr_renovated  -6.436e+00  2.841e+01  -0.227  0.82102
## zipcode       -2.787e+02  2.511e+02  -1.110  0.26850
## lat            5.234e+05  8.952e+04   5.847 2.27e-08 ***
```

```
## long          -1.008e+05  1.054e+05  -0.956  0.34035
## sqft_living15  6.516e+01  2.781e+01   2.343  0.02022 *
## sqft_lot15     1.543e+00  1.103e+00   1.399  0.16358
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 148400 on 182 degrees of freedom
## Multiple R-squared:  0.7998, Adjusted R-squared:  0.7812
## F-statistic: 42.78 on 17 and 182 DF,  p-value: < 2.2e-16
```

```
plot(lm.sol, which = 1)
```



从残差图看出，这是一个喇叭型残差图，是方差齐性不符合的一个症状。我们考虑对因变量 $price$ 作变换，尝试 $z = log(y)$, 得到回归方程

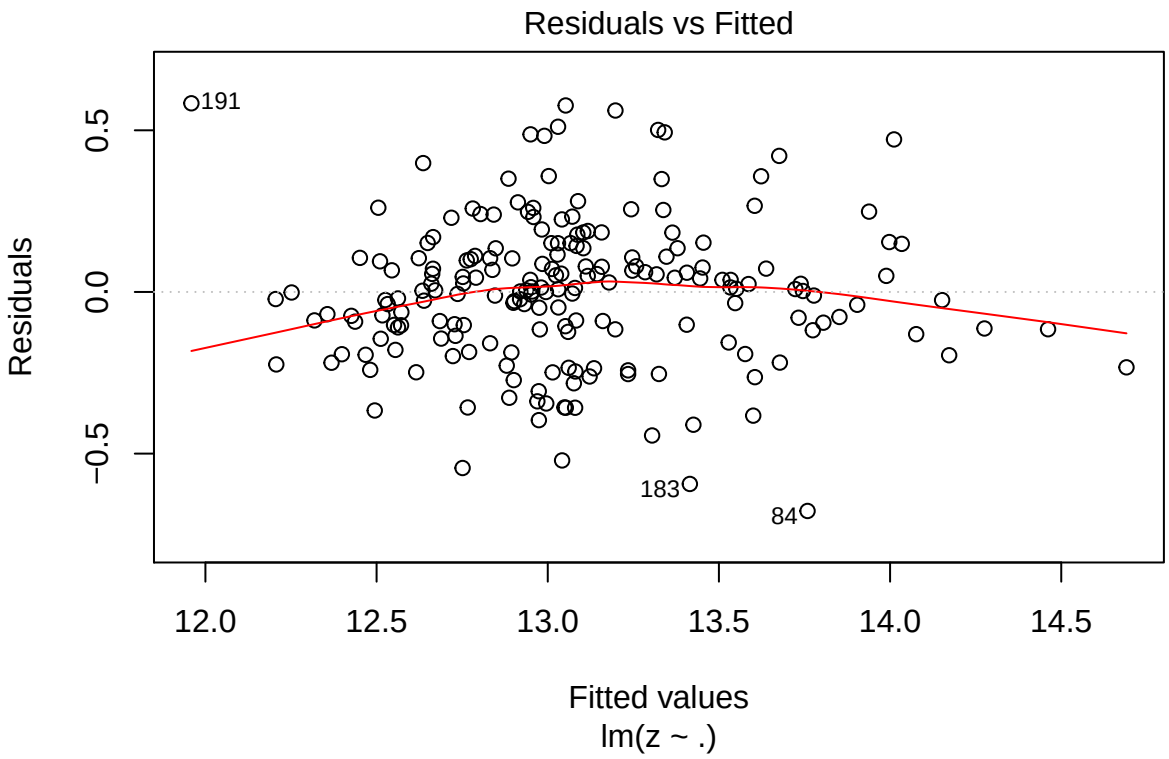
$$z = X\beta + \varepsilon$$

```
z <- log(data.trans[, 'price'])
data2.trans <- data.frame(z, data.trans[, - 1])
lm.sol2 <- lm(z~., data = data2.trans)
summary(lm.sol2)
```

```
## Call:
```

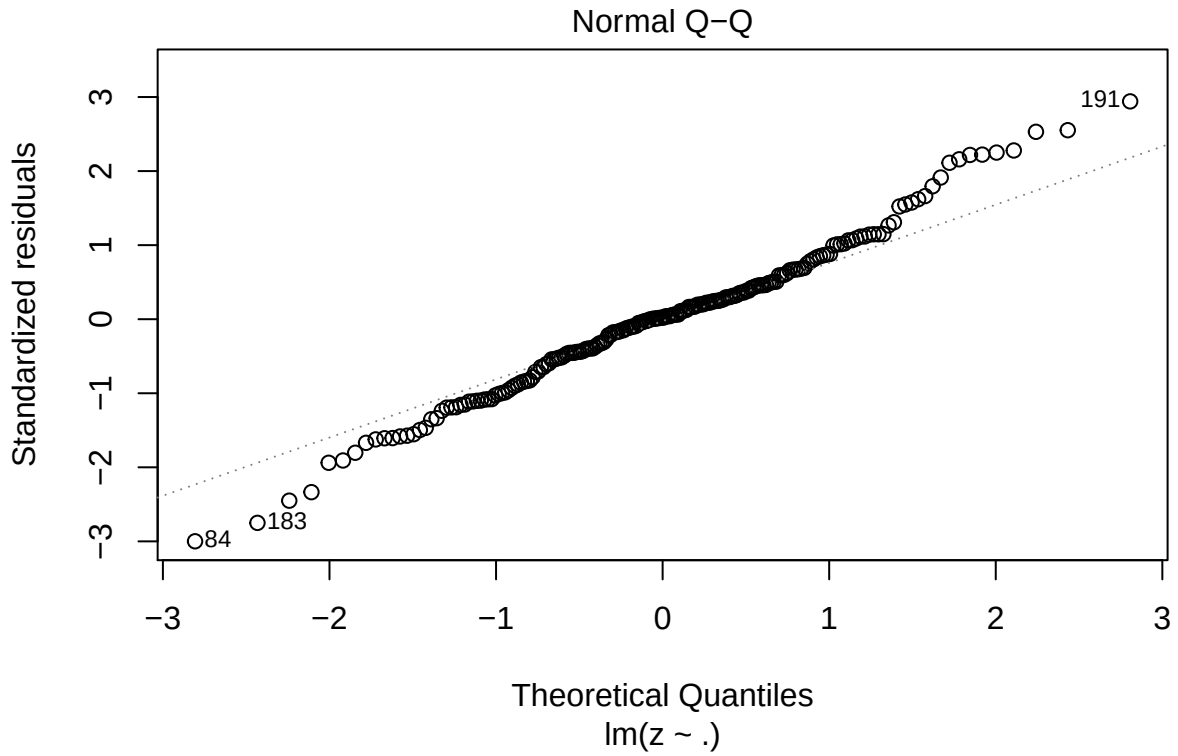
```
## lm(formula = z ~ ., data = data2.trans)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.67735 -0.11967  0.00482  0.11253  0.58352
##
## Coefficients:
##                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)   -5.375e+01  3.430e+01  -1.567  0.11892
## bedrooms      -4.460e-02  2.117e-02  -2.107  0.03647 *
## bathrooms      7.378e-02  3.997e-02   1.846  0.06650 .
## sqft_living    1.193e-04  5.076e-05   2.351  0.01981 *
## sqft_lot      -1.419e-06  8.315e-07  -1.706  0.08969 .
## floors         9.215e-02  4.416e-02   2.087  0.03830 *
## waterfront     3.624e-01  1.984e-01   1.827  0.06940 .
## view           7.196e-02  2.344e-02   3.070  0.00247 **
## condition      2.522e-02  2.693e-02   0.936  0.35027
## grade          1.264e-01  2.717e-02   4.653 6.28e-06 ***
## sqft_above     5.839e-05  5.533e-05   1.055  0.29262
## yr_built      -4.237e-03  8.534e-04  -4.965 1.57e-06 ***
## yr_renovated   1.934e-05  4.457e-05   0.434  0.66492
## zipcode        4.965e-05  3.940e-04   0.126  0.89986
## lat            1.356e+00  1.404e-01   9.654  < 2e-16 ***
## long          -3.224e-02  1.654e-01  -0.195  0.84566
## sqft_living15  1.214e-04  4.363e-05   2.781  0.00599 **
## sqft_lot15     2.734e-06  1.731e-06   1.580  0.11589
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.2328 on 182 degrees of freedom
## Multiple R-squared:  0.806,  Adjusted R-squared:  0.7879
## F-statistic: 44.48 on 17 and 182 DF,  p-value: < 2.2e-16
```

```
plot(lm.sol2, which = 1)
```



新的残差图不呈现任何明显的规则性，这表明我们的变化是合适的。

```
plot(lm.sol2, which = 2)
```

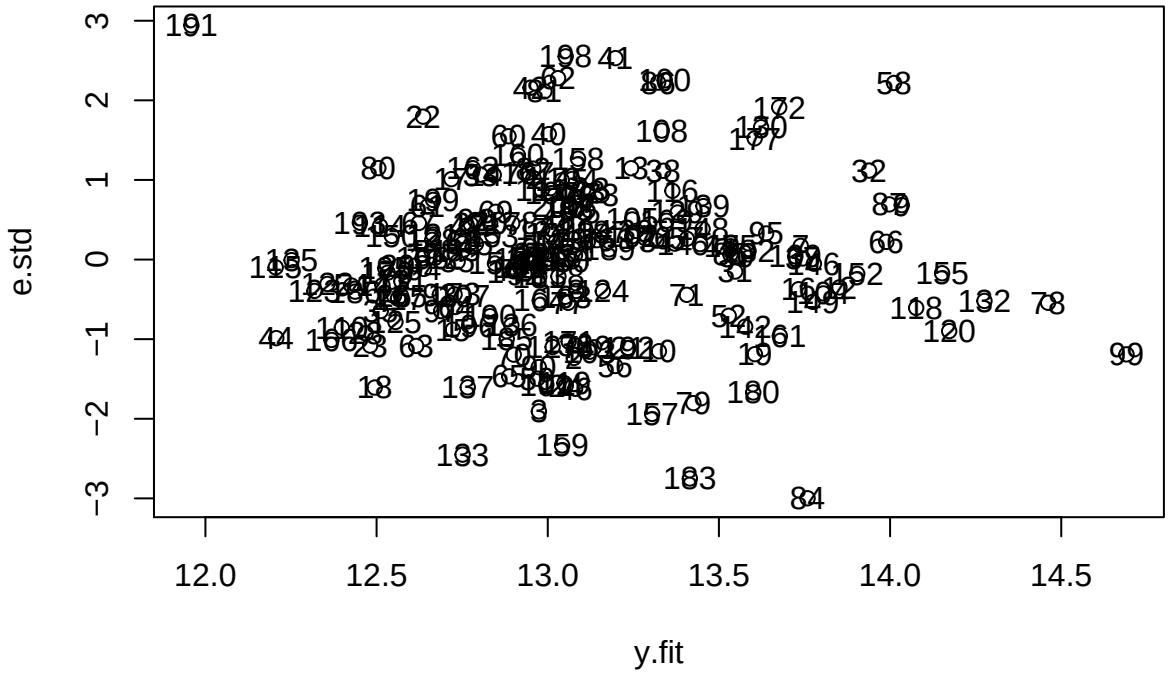


通过残差图和 QQ 图，我们接受线性假设、方差齐性假设、不相关性假设和正态性假设。

2.1.2 数据的诊断

2.1.2.1 异常点诊断

```
y.fit <- predict(lm.sol2)
e.std <- rstandard(lm.sol2)

plot(e.std~y.fit)
text(y.fit, e.std)
```



可以看到，其中有着许多数据的学生化残差是大于 2 的，我们将其打印出来。

```
point.abnormal <- abs(e.std) > 2
point.abnormal <- (1:200)[point.abnormal]
point.abnormal
```

```
##  [1]  41  42  58  62  81  84  86 100 133 159 183 191 198
```

我们剔除学生化残差 $|r_i| > 2$ 的异常点数据。

```
data2.trans <- data2.trans[-c(point.abnormal),]
```

2.1.2.2 强影响点诊断

综合 cook 距离，我们将 $D_i > 0.01$ 的数据，作为强影响点，将其打印出来如下。

```r
influ.mea <- influence.measures(lm.sol2)
point.influ <- influ.mea$infmat[, 'cook.d'] > 0.01
point.influ <- (1:200)[point.influ]
point.influ
```

```
##  [1]   3  22  28  41  42  46  56  58  62  65  84  86  89  99 100 101 108
## [18] 113 130 133 137 159 160 172 177 183 191 197 198
```

2.1.3 多重共线性诊断

下面我们做多重共线性诊断。

```r
# VIF
vif(lm.sol2)
```

```
##      bedrooms     bathrooms   sqft_living      sqft_lot        floors
##        1.8006        3.2566        7.2363        3.5865        2.1318
##    waterfront          view     condition         grade    sqft_above
##        1.4381        1.6057        1.3285        4.1328        7.1292
##      yr_built  yr_renovated       zipcode           lat          long
##        2.4939        1.2627        1.7185        1.1966        2.3457
## sqft_living15    sqft_lot15
##        3.1510        3.9131
```

可以看到没有任何一个变量的 VIF 值大于 10，因此不存在着多重共线性。

```r
# 特征根诊断法
rho <- cor(data.trans[, -1])
eigen(rho)$values
```

```
##  [1] 5.28417978 2.14009690 1.59152696 1.37595250 1.19937212 1.08767712
##  [7] 0.84101974 0.72555094 0.55451358 0.51223902 0.48431923 0.34120744
## [13] 0.26190073 0.21565551 0.17827100 0.12830668 0.07821076
```

再看特征根诊断法，其中没有特征根近似为零，则有不存在多重共线性关系。

```r
# 条件数诊断法
kappa(rho, exact = TRUE)
```

```
## [1] 67.56333
```

最后看条件数诊断法，CI 小于 100，则可以认为没有多重共线性。

- 结论：数据集不存在多重共线性。

2.2 线性回归

2.2.1 线性回归

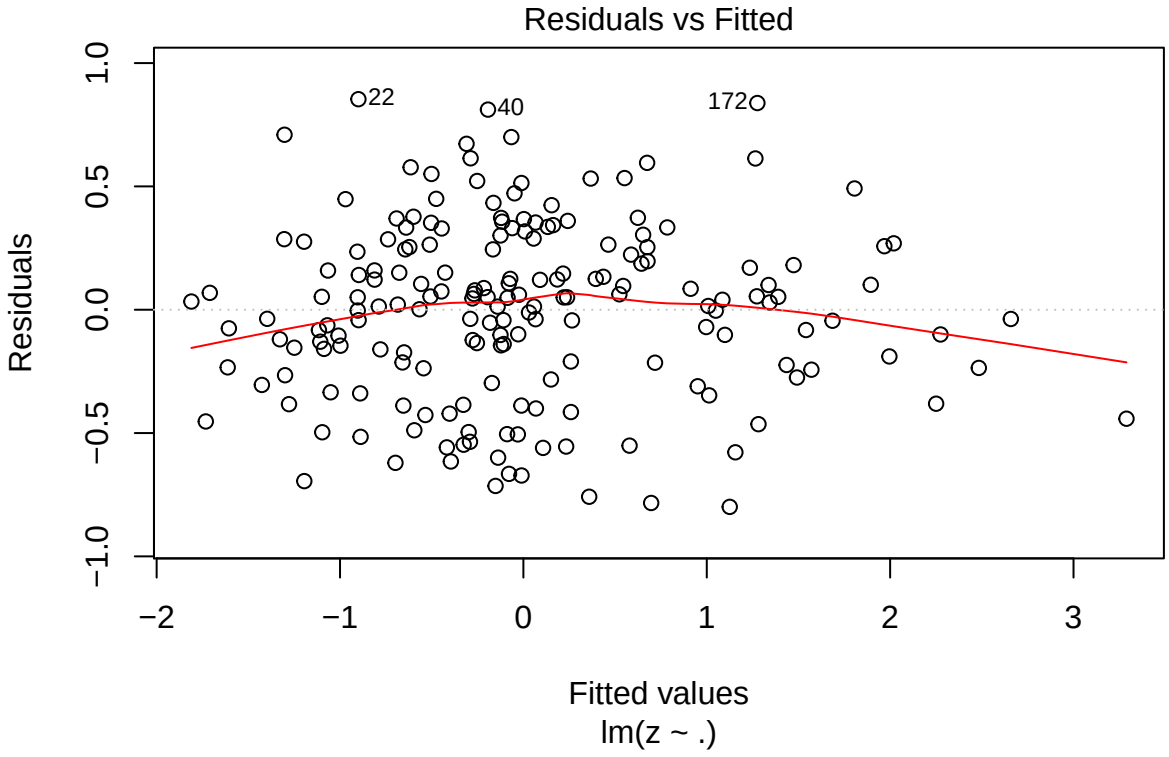
我们将回归诊断后的模型和数据标准化，重新运行一次模型。

```
data2.trans.scale <- scale(data2.trans)
data3.trans <- data.frame(data2.trans.scale)
lm.sol3 <- lm(z~., data = data3.trans)
summary(lm.sol3)
```

```
## Call:
## lm(formula = z ~ ., data = data3.trans)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.79926 -0.23477  0.03316  0.25577  0.85355
##
## Coefficients:
##                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)   -1.365e-15  2.721e-02   0.000  1.00000
## bedrooms      -6.160e-02  3.736e-02  -1.649  0.10106
## bathrooms      5.789e-02  4.923e-02   1.176  0.24129
## sqft_living    1.940e-01  7.656e-02   2.534  0.01219 *
## sqft_lot      -9.258e-02  5.247e-02  -1.765  0.07944 .
## floors         1.320e-01  4.040e-02   3.268  0.00131 **
## waterfront     9.810e-02  3.323e-02   2.952  0.00360 **
## view           8.710e-02  3.521e-02   2.474  0.01437 *
## condition      7.812e-02  3.228e-02   2.420  0.01657 *
## grade          3.541e-01  5.728e-02   6.182 4.59e-09 ***
## sqft_above     4.176e-02  7.572e-02   0.552  0.58202
## yr_built      -2.275e-01  4.306e-02  -5.283 3.87e-07 ***
## yr_renovated   6.237e-02  3.104e-02   2.009  0.04613 *
## zipcode        6.622e-02  3.636e-02   1.821  0.07036 .
## lat            3.445e-01  2.997e-02  11.495  < 2e-16 ***
## long          -1.085e-02  4.223e-02  -0.257  0.79755
```

```
## sqft_living15  2.476e-01  4.927e-02   5.027 1.26e-06 ***
## sqft_lot15     1.059e-01  5.451e-02   1.943  0.05364 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.3721 on 169 degrees of freedom
## Multiple R-squared:  0.8742, Adjusted R-squared:  0.8616
## F-statistic: 69.09 on 17 and 169 DF,  p-value: < 2.2e-16
```

```
plot(lm.sol3, which = 1)
```



2.2.1.1 对回归方程的显著性作检验（显著性水平 $\alpha = 0.05$）

可以看到，$F_H = 69.09$, p 值小于 $2.2 \times 10^{-6} < 0.05$, 所以我们认为回归自变量对因变量有着显著的线性影响。

2.2.1.2 对每一个回归系数的显著性作检验（显著性水平 $\alpha = 0.05$）

可以看到，bedrooms、bathrooms、sqft_lot、sqft_above、zipcode、long、sqft_lot15 的 p 值均大于 0.05，我们接受对应自变量的显著性假设。因此我们需要进行变量选择。

2.2.2 变量选择

2.2.2.1 自变量选择的准则

首先对所有子模型进行运行，得到所有子模型的各个准则，总共有 131071 个子模型。

```
## [1] "The number of submodel"
## [1] 131071
```

```
formu <- paste(yname, "~.")
search.results <- regsubsets(as.formula(formu), data = data3.trans,
                             method = "exhaustive", nbest = nbest,
                             really.big = T)
selection.criteria <- summary(search.results)

n <- length(data3.trans[, 1])
q <- as.integer(row.names(selection.criteria$which))
R.sq <- selection.criteria$rsq
AdjR.sq <- selection.criteria$adjr2
rms <- selection.criteria$rss / (n - q - 1)
Cp <- selection.criteria$cp
aic.f <- n * log(selection.criteria$rss) + 2 * (q + 1)
bic.f <- n * log(selection.criteria$rss) + (q + 1) * log(n)
var <- as.matrix(selection.criteria$which[, 2: length(data3.trans[1,]) ])
press.f <- calcuPress(data3.trans, var, nbest, yname = yname)
criteria.table <- data.frame(cbind(q, rms, R.sq, AdjR.sq, Cp, aic.f, bic.f, press.f), var)
```

- 调整后的 R^2 准则可以得到，根据调整后的 R^2 准则，我们选择以下自变量子集：

```
var.AdjR.sq <- criteria.table[which.max(criteria.table[,'AdjR.sq']), ]
varnameList[as.matrix(var.AdjR.sq[1, 9:25])[1, ]]
```

```
## [1] "sqft_living"   "floors"        "waterfront"    "view"
## [5] "grade"         "yr_built"      "lat"           "sqft_living15"
```

- Cp 准则可以得到，根据 Cp 准则，我们选择以下自变量子集：

```
var.Cp <- criteria.table[which.min(criteria.table[,'Cp']),]
varnameList[as.matrix(var.Cp[1, 9:25])[1, ]]
```

```
## [1] "sqft_living"   "floors"        "waterfront"    "view"
## [5] "grade"         "yr_built"      "lat"           "sqft_living15"
```

- AIC 准则可以得到，根据 AIC 准则，我们选择以下自变量子集：

```
var.aic.f <- criteria.table[which.min(criteria.table[,'aic.f']),]
varnameList[as.matrix(var.aic.f[1, 9:25])[1, ]]
```

```
## [1] "sqft_living"   "floors"        "waterfront"    "view"
## [5] "grade"         "yr_built"      "lat"           "sqft_living15"
```

- BIC 准则可以得到，根据 BIC 准则，我们同样选择以下自变量子集：

```
var.bic.f<- criteria.table[which.min(criteria.table[,'bic.f']),]
varnameList[as.matrix(var.bic.f[1, 9:25])[1, ]]
```

```
## [1] "sqft_living"   "floors"        "waterfront"    "grade"
## [5] "yr_built"      "lat"           "sqft_living15"
```

- $PRESS$ 准则可以得到，根据 $PRESS$ 准则，我们同样选择以下自变量子集：

```
var.press.f <- criteria.table[which.min(criteria.table[,'press.f']),]
varnameList[as.matrix(var.press.f[1, 9:25])[1, ]]
```

```
## [1] "sqft_living"   "floors"        "waterfront"    "view"
## [5] "grade"         "yr_built"      "lat"           "sqft_living15"
```

2.2.2.2 逐步回归法

我们使用 $SignifReg$ 函数进行逐步回归，这个函数相比于 $step$ 函数有三个优点：

- 具有更多的 criterion。除了 $step$ 函数有的 AIC 准则外，还具有 BIC、C_p、$r-adj$、$p-value$ 等准则。
- 在模型选择过程中，会同步进行变量回归系数的显著性检验。如若没有通过 α 置信水平的检验，则不将变量选入模型中。
- 此外，对于逐步回归法，除了从一个空模型开始之外，还能够从一个全模型开始，逐步删除变量。

```
yname <- "z"
varnameList <- colnames(data3.trans)
varnameList <- varnameList[- which(varnameList == yname)]
formu <- as.formula(paste(yname, paste("~", paste(varnameList, collapse = "+"))))
step.model <- SignifReg(scope = formu, data = data3.trans, alpha = 0.05,
                        direction = "step_null", criterion = "AIC")
```

```
## Call:
## lm(formula = formu, data = data3.trans)
##
```

```
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.92360 -0.23559  0.00277  0.27197  0.85419
##
## Coefficients:
##                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)   -5.816e-15  2.816e-02   0.000 1.000000
## grade          4.179e-01  5.171e-02   8.082 9.20e-14 ***
## lat            3.533e-01  2.969e-02  11.900  < 2e-16 ***
## sqft_living    2.242e-01  4.941e-02   4.537 1.04e-05 ***
## yr_built      -2.776e-01  3.353e-02  -8.280 2.78e-14 ***
## waterfront     1.641e-01  2.873e-02   5.713 4.56e-08 ***
## sqft_living15  2.460e-01  4.653e-02   5.287 3.59e-07 ***
## floors         1.193e-01  3.489e-02   3.420 0.000777 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.385 on 179 degrees of freedom
## Multiple R-squared:  0.8573, Adjusted R-squared:  0.8517
## F-statistic: 153.6 on 7 and 179 DF,  p-value: < 2.2e-16
```

可以看到，逐步回归法得到的结果，选取了 grade, lat, sqft_living, yr_built, waterfront, sqft_living15, floors 共 7 个自变量。得到的回归方程为：

$$\begin{aligned} z^* =& 0.4179\, grade^* + 0.3533\, lat^* + 0.2242\, sqft\_living^* \\ &+ -0.2776\, yr\_built^* + 0.1641\, waterfront^* + 0.246\, sqft\_living15^* \\ &+ 0.1193\, floors^* \end{aligned}$$

上述方程是经过标准化后的，我们将其还原成原来的系数，如下：

$$\begin{aligned} z =& 4.5396\, grade + 35.7958\, lat + 0.0033\, sqft\_living \\ &+ -0.1177\, yr\_built + 20.7789\, waterfront + 0.0047 sqft\_living15 \\ &+ 2.8302\, floors - 1513.858 \end{aligned}$$

即：

$$\begin{aligned}\log(y) =& 4.5396\, grade + 35.7958\, lat + 0.0033\, sqft\_living \\ &+ -0.1177\, yr\_built + 20.7789\, waterfront + 0.0047 sqft\_living15 \\ &+ 2.8302\, floors - 1513.858\end{aligned}$$

2.3 主成分回归

由于上述线性回归过程中，我们可以看到自变量个数较多，下面尝试主成分回归方法。

```
yname <- "z"
varnameList <- colnames(data3.trans)
varnameList <- varnameList[- which(varnameList == yname)]
formu <- as.formula(paste("~", paste(varnameList, collapse = "+")))
y.pr <- princomp(formu, data = data3.trans, cor = TRUE)
summary(y.pr)
```

```
## Importance of components:
##                           Comp.1    Comp.2     Comp.3     Comp.4
## Standard deviation     2.3270123 1.4407455 1.26992673 1.18082869
## Proportion of Variance 0.3185286 0.1221028 0.09486552 0.08202096
## Cumulative Proportion  0.3185286 0.4406314 0.53549692 0.61751788
##                           Comp.5     Comp.6     Comp.7     Comp.8
## Standard deviation     1.08988417 1.05454190 0.92136091 0.83712864
## Proportion of Variance 0.06987338 0.06541521 0.04993564 0.04122261
## Cumulative Proportion  0.68739127 0.75280648 0.80274212 0.84396473
##                           Comp.9    Comp.10    Comp.11    Comp.12
## Standard deviation     0.72533956 0.71395682 0.67055480 0.56875075
## Proportion of Variance 0.03094809 0.02998437 0.02644963 0.01902808
## Cumulative Proportion  0.87491282 0.90489719 0.93134682 0.95037491
##                           Comp.13    Comp.14    Comp.15     Comp.16
## Standard deviation     0.51029847 0.46281328 0.42236338 0.344631100
## Proportion of Variance 0.01531791 0.01259977 0.01049358 0.006986506
## Cumulative Proportion  0.96569282 0.97829259 0.98878617 0.995772675
##                           Comp.17
## Standard deviation     0.268075585
## Proportion of Variance 0.004227325
```

```
## Cumulative Proportion  1.000000000
```

可以看到，前 9 个主成分的 Cumulative Proportion 达到了 0.87491282，我们可以只选取前 9 个主成分，删除后 8 个主成分。

前九个主成分对应的标准正交化特征向量可由下面这个命令打印出来。我们只查看前 4 个标准正交化特征向量。

```
y.pr$loadings[, 1:9]
```

```
##                    Comp.1      Comp.2       Comp.3      Comp.4
## bedrooms       0.17275752  0.26633220  0.020224937  0.50175944
## bathrooms      0.32210921  0.20370328 -0.031434934  0.02101727
## sqft_living    0.36722569  0.17906689 -0.132195885  0.18932991
## sqft_lot       0.15998719 -0.50084208 -0.199735800  0.13550622
## floors         0.26378636  0.21097062  0.072467151 -0.28804415
## waterfront     0.04533319 -0.04403962 -0.440505893 -0.25970622
## view           0.15940143 -0.05741524 -0.530455853 -0.12144031
## condition     -0.07877952 -0.04929760 -0.095049354  0.40840179
## grade          0.36418604  0.06917520 -0.105471467 -0.07991571
## sqft_above     0.38946624  0.12948326  0.026510238  0.03534522
## yr_built       0.26086170 -0.07965103  0.310507821 -0.37090949
## yr_renovated  -0.03979088  0.17871531 -0.114050930  0.38320617
## zipcode       -0.18645856  0.18665379 -0.422455385 -0.13786490
## lat           -0.02310463  0.24204175 -0.132707417 -0.18255780
## long           0.24069459 -0.35901771  0.285915449  0.05224916
## sqft_living15  0.35409497  0.06490277 -0.009420864  0.06036788
## sqft_lot15     0.17358343 -0.51587308 -0.228805553  0.10424251
```

计算主成分得分，进行主成分估计：

```
pre = predict(y.pr)
z <- data3.trans$z
data3.trans.pc <- data.frame(z, pre[, 1:9])
pc.sol <- lm(z~., data = data3.trans.pc)
summary(pc.sol)
```

```
## Call:
## lm(formula = z ~ ., data = data3.trans.pc)
##
```

```
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.83950 -0.24359 -0.00438  0.23949  1.12940
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -7.897e-16  2.818e-02   0.000 1.000000
## Comp.1       2.773e-01  1.211e-02  22.898  < 2e-16 ***
## Comp.2       2.108e-01  1.956e-02  10.780  < 2e-16 ***
## Comp.3      -3.140e-01  2.219e-02 -14.152  < 2e-16 ***
## Comp.4      -1.279e-02  2.386e-02  -0.536 0.592515
## Comp.5      -5.325e-02  2.585e-02  -2.060 0.040894 *
## Comp.6      -3.369e-01  2.672e-02 -12.609  < 2e-16 ***
## Comp.7      -2.310e-01  3.058e-02  -7.555 2.16e-12 ***
## Comp.8      -5.166e-02  3.366e-02  -1.535 0.126611
## Comp.9      -1.350e-01  3.885e-02  -3.476 0.000641 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.3853 on 177 degrees of freedom
## Multiple R-squared:  0.8587, Adjusted R-squared:  0.8515
## F-statistic: 119.5 on 9 and 177 DF,  p-value: < 2.2e-16
```

可以看到，有第 4、第 8 个主成分没有通过单变量回归系数的显著性检验，我们再次进行变量选择:

```
yname <- "z"
varnameList <- colnames(data3.trans.pc)
varnameList <- varnameList[- which(varnameList == yname)]
formu <- as.formula(paste(yname, paste("~", paste(varnameList, collapse = "+"))))
step.model.pc <- SignifReg(scope = formu, data = data3.trans.pc, alpha = 0.05,
                           direction = "step_null", criterion = "r-adj")
```

```
## Call:
## lm(formula = formu, data = data3.trans.pc)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.0395 -0.2624  0.0287  0.2719  1.1312
```

```
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -7.949e-16  2.848e-02   0.000 1.000000
## Comp.1       2.773e-01  1.224e-02  22.654  < 2e-16 ***
## Comp.3      -3.140e-01  2.243e-02 -14.001  < 2e-16 ***
## Comp.6      -3.369e-01  2.701e-02 -12.475  < 2e-16 ***
## Comp.2       2.108e-01  1.977e-02  10.666  < 2e-16 ***
## Comp.7      -2.310e-01  3.091e-02  -7.475 3.26e-12 ***
## Comp.9      -1.350e-01  3.926e-02  -3.439 0.000726 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.3895 on 180 degrees of freedom
## Multiple R-squared:  0.8532, Adjusted R-squared:  0.8483
## F-statistic: 174.4 on 6 and 180 DF,  p-value: < 2.2e-16
```

最终，我们可以得到第 1、2、3、6、7、9 个主成分选入模型，它们的累计贡献值达到了 0.6817959，回归模型如下:

$$\begin{aligned} z^* =& 0.2773\, Comp.1 + -0.314\, Comp.3 + -0.3369\, Comp.6 \\ &+ 0.2108\, Comp.2 + -0.231\, Comp.7 + -0.135\, Comp.9 \end{aligned}$$

也就是

$$\begin{aligned} z^* =& 1.7626\, bedrooms^* + 1.244\, bathrooms^* + 0.3888\, sqft\_living^* \\ &+ 0.186\, sqft_lot^* + 0.9191\, floors^* - 0.776\, waterfront^* \\ &- 0.7024\, view^* - 0.3198\, condition^* - 0.2675\, grade^* \\ &+ 0.3662\, sqft\_above^* + 0.71\, yr\_built^* + 0.5819\, yr\_renovated^* \\ &+ 0.0634\, zipcode^* - 0.9196\, lat^* - 0.1973\, long^* \\ &- 0.3737\, sqft\_living15^* - 0.2022\, sqft\_lot15^* \end{aligned}$$

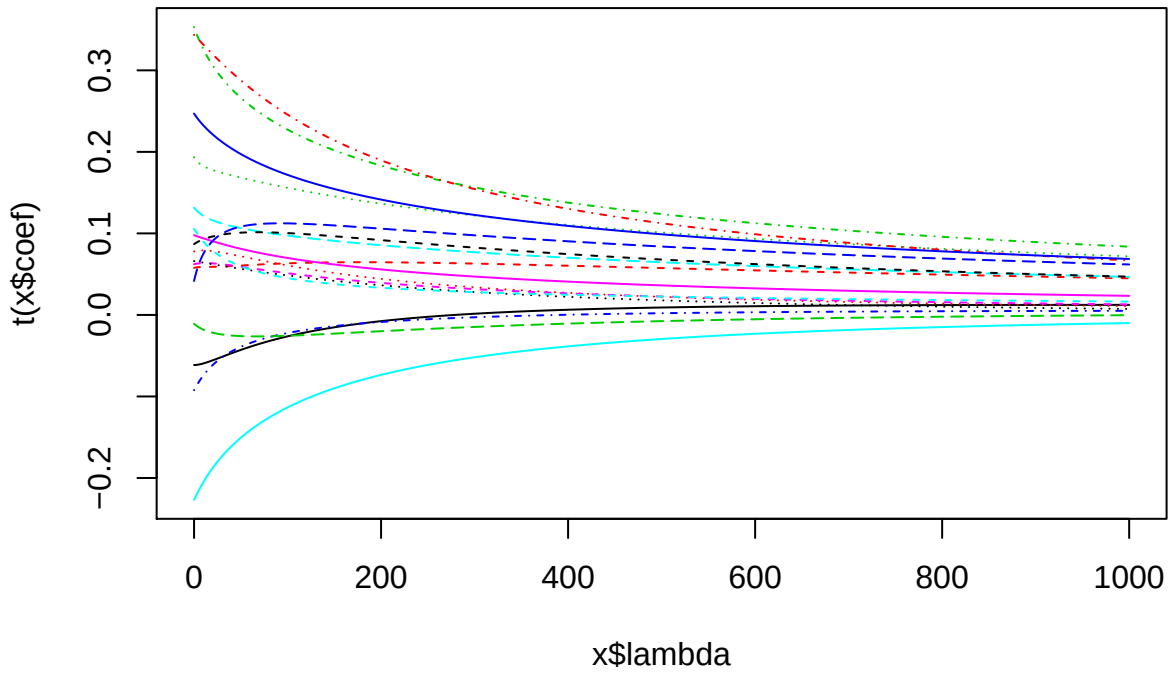
即

$$\begin{aligned}\log(y) =& 21.9432\, bathrooms + 0.0057\, sqft\_living + 0.0001 sqft\_lot \\ &+ 21.8017\, floors - 98.241\, waterfront - 10.4364\, view \\ &- 5.9259\, condition - 2.9055\, grade + 0.0059\, sqft\_above \\ &+ 0.3011\, yr\_built + 0.0187\, yr\_renovated + 0.0151\, zipcode \\ &- 93.1814\, lat - 16.9743\, long - 0.0072\, sqft\_living15 \\ &- 0.0001\, sqft\_lot15 + 254.5485\end{aligned}$$

2.4 岭回归

下面我们用岭估计的方法寻找岭回归方程。

```
yname <- "z"
varnameList <- colnames(data3.trans)
varnameList <- varnameList[- which(varnameList == yname)]
formu <- as.formula(paste(paste(yname, "~"), paste(varnameList, collapse = "+")))
rr.sol <- lm.ridge(formu, data = data3.trans,
                   lambda = c(seq(0, 1000, by = 1)))
plot(rr.sol)
```



可以看到，岭回归的方法需要较大的 λ 才能够使得岭迹图趋于平稳。我们选取 $\lambda = 400$ 进行分析。

```
yname <- "z"
varnameList <- colnames(data3.trans)
varnameList <- varnameList[- which(varnameList == yname)]
formu <- as.formula(paste(paste(yname, "~"), paste(varnameList, collapse = "+")))
ridge.model <- lm.ridge(formu, data = data3.trans,
                        lambda = 400)
ridge.model
```

```
##                   bedrooms     bathrooms   sqft_living      sqft_lot
## -1.506414e-15  6.373284e-03  6.052329e-02  1.107143e-01  2.458317e-04
##        floors    waterfront          view     condition         grade
##  7.034580e-02  4.097970e-02  7.479548e-02  2.710777e-02  1.381468e-01
##    sqft_above      yr_built  yr_renovated       zipcode           lat
##  9.055236e-02 -3.863127e-02  2.649473e-02  2.225569e-02  1.304872e-01
##          long sqft_living15    sqft_lot15
## -1.068821e-02  1.095004e-01  2.481061e-02
```

2.5 预测

2.5.1 测试集数据变换

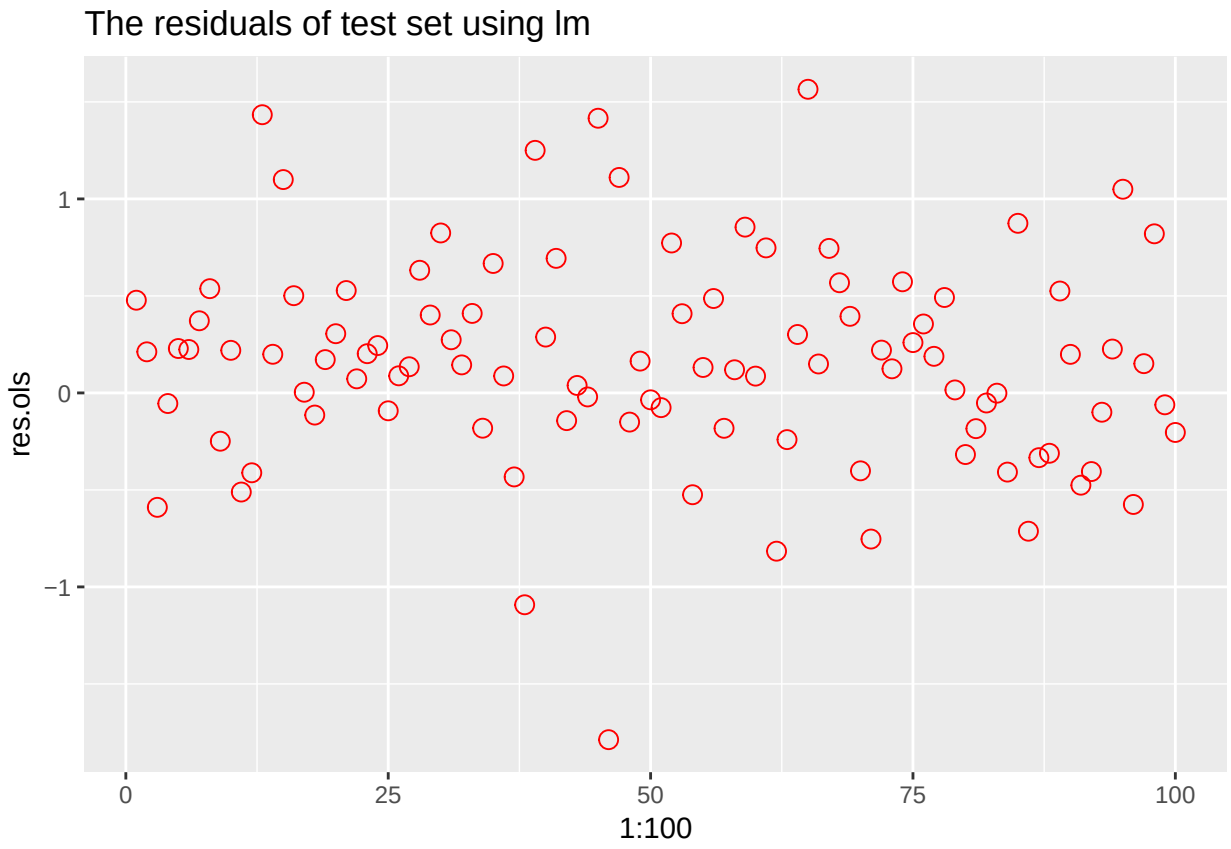
首先对测试集数据，做与训练集一样的 scale 变换。

```
data2.trans.scale.center <- attr(data2.trans.scale,"scaled:center")
data2.trans.scale.scale <- attr(data2.trans.scale,"scaled:scale")
data2.test <- data.test
for (i in colnames(data.test)){
  data2.test[i] <- (data.test[i] - data2.trans.scale.center[i]) / data2.trans.scale.scale[i]
}
data2.test <- data2.test[-1]

y.test <- data.test$price
y.center <- data2.trans.scale.center["z"]
y.scale <- data2.trans.scale.scale["z"]
y.log.test <- (log(y.test) - y.center) / y.scale
```

2.5.2 使用线性回归模型进行预测

```
ypred.log.ols <- predict(ols.model, newdata = data2.test)
ypred.ols <- exp(ypred.log.ols * y.scale + y.center)
res.ols <- y.log.test - ypred.log.ols
mse.ols <- mean(res.ols^2)
mse.ols
```

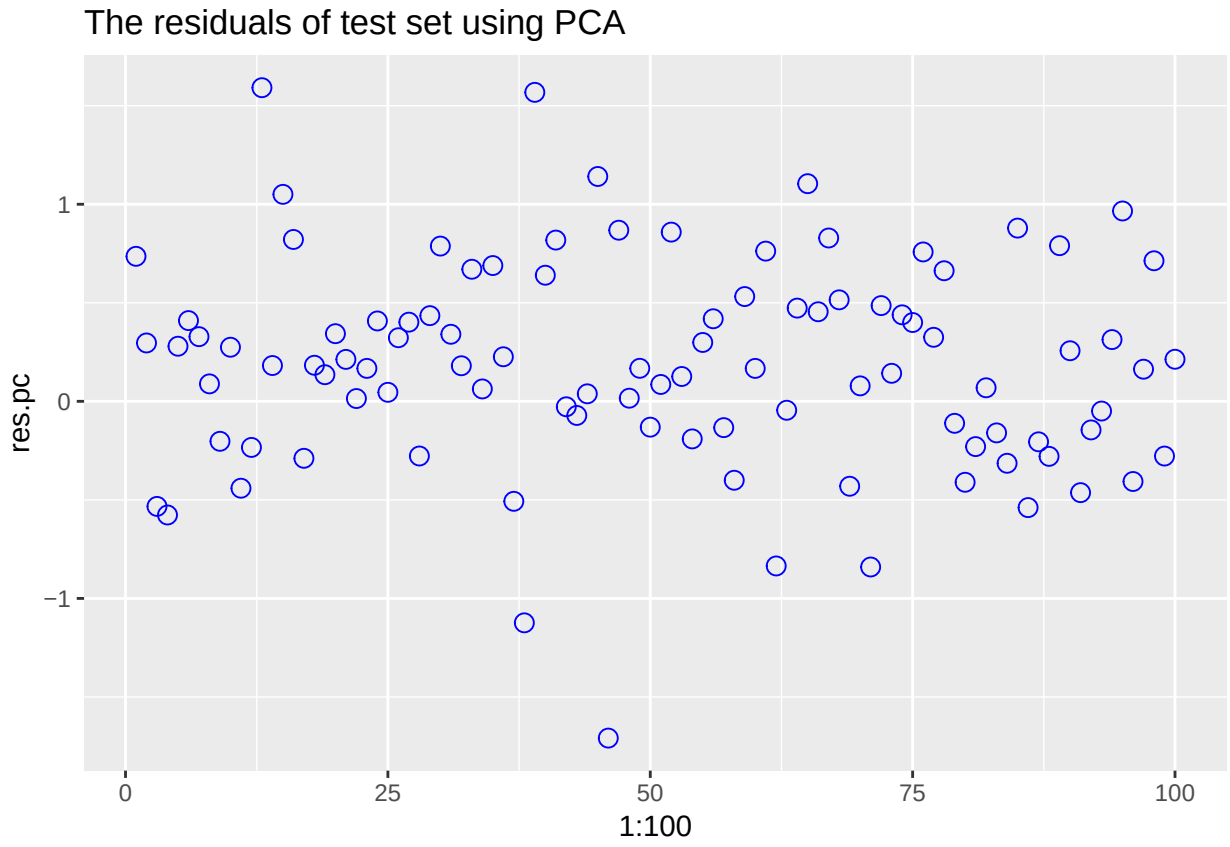
```
## [1] 0.3108866
```



2.5.3 使用主成分回归模型进行预测

```
data2.test.pc <- data.frame(predict(y.pr, newdata = data2.test))
ypred.log.pc <- predict(pc.model, newdata = data2.test.pc)
ypred.pc <- exp(ypred.log.pc * y.scale + y.center)
res.pc <- y.log.test - ypred.log.pc
mse.pc <- mean(res.pc^2)
mse.pc
```

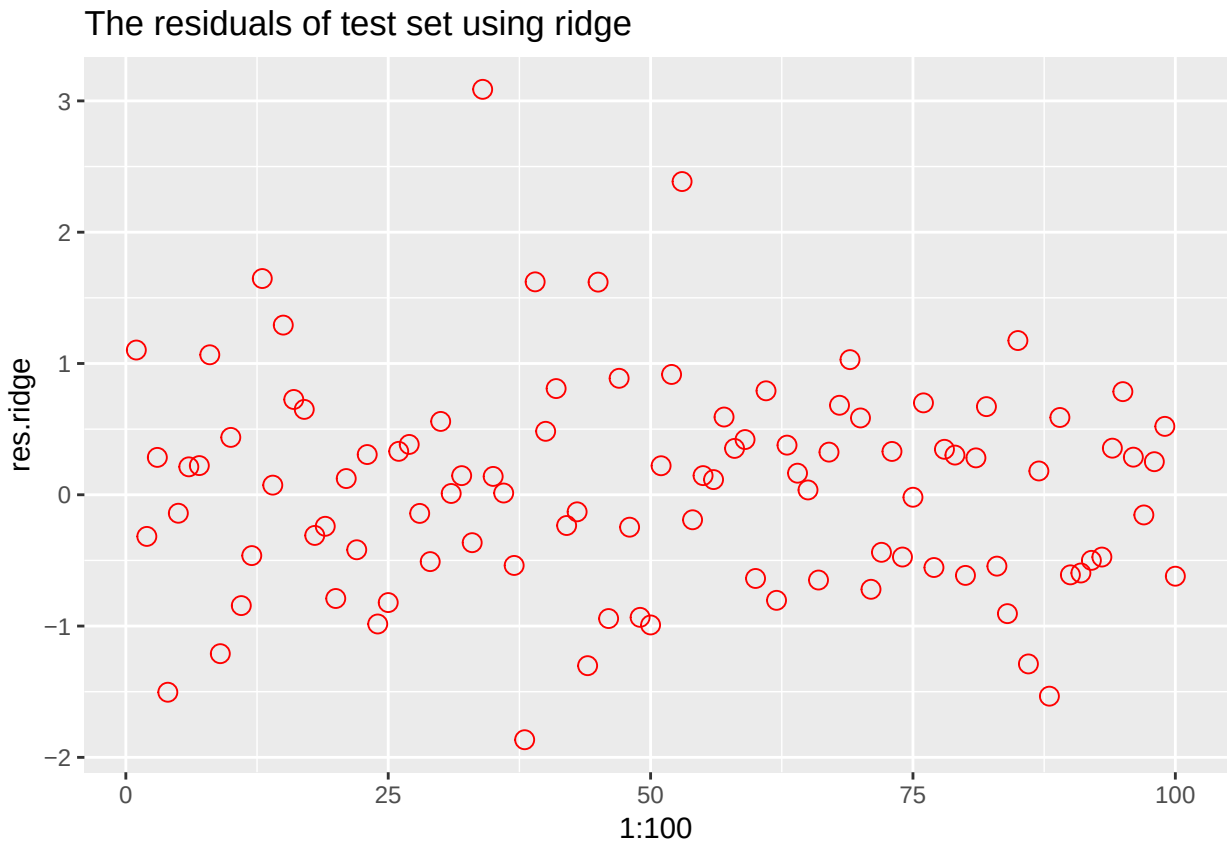
```
## [1] 0.3156006
```



2.5.4 使用岭回归模型进行预测

```
ypred.log.ridge <- rowSums(ridge.model$coef * data2.test)
ypred.ridge <- exp(ypred.log.ridge * y.scale + y.center)
res.ridge <- y.log.test - ypred.log.ridge
mse.ridge <- mean(res.ridge^2)
mse.ridge
```

```
## [1] 0.6553124
```



3 拓展

3.1 Resample

首先，在前面的报告中，可以看到我们并没有一次性将所有数据全部用于训练模型，因为否则我们将没有数据对我们的模型进行验证，从而评估模型的效果。前面的报告中，我们采用了一种最简单、也是最容易想到的方法，就是将整个数据集分成两部分，一部分用于训练，一部分用于验证，也就是训练集和测试集。

但是这种方法有两个弊端：首先是模型和参数的选择将很大依赖于对训练集和测试集的划分方法；此外，只使用了一部分数据进行模型的训练，而数据量越大模型效果通常会更好，所以模型的效果会受到一定的影响。

下面我们采用两种 resample 的方法，可以在一定程度上解决这个问题。

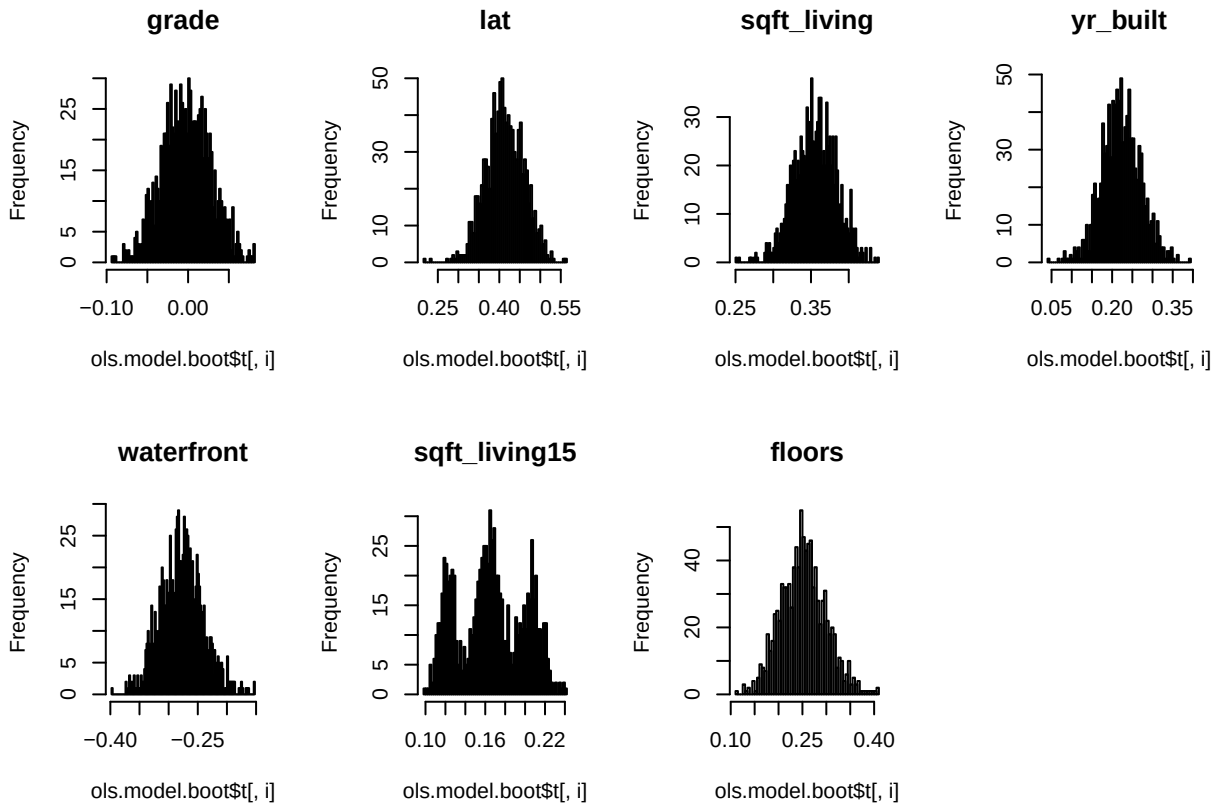
3.1.1 Boostrap

Boostrap 是一种 resample 的方法，用于通过对替换的数据集进行重复多次采样来估计总体的统计数据，如均值、标准差，或者是总体的分布。

其实现的方式是这样子的：从训练集中有放回的均匀抽样，每个样本可以得到一个估计，这样一来，可以得到 n 个估计，也就可以估计出总体的分布。

```
set.seed(123)
yname <- "z"
varnameList <- names(step.model$coefficients[-1])
formu <- as.formula(paste(yname, paste("~", paste(varnameList, collapse = "+"))))
getRegr <- function(data, idx) {
  yname <- "z"
  varnameList <- names(step.model$coefficients[-1])
  formu <- as.formula(paste(yname, paste("~", paste(varnameList, collapse = "+"))))
  bsFit <- lm(formu, subset=idx, data=data)
  coef(bsFit)
}
nR <- 1000
ols.model.boot <- boot(data3.trans, statistic=getRegr, R=nR)
ols.model.boot
```

```
## ORDINARY NONPARAMETRIC BOOTSTRAP
## Call:
## boot(data = data3.trans, statistic = getRegr, R = nR)
##
## Bootstrap Statistics :
##          original        bias    std. error
## t1* -5.815643e-15 -0.0021240787  0.02971774
## t2*  4.179040e-01 -0.0050539346  0.04656104
## t3*  3.532586e-01  0.0021085834  0.02780745
## t4*  2.241756e-01 -0.0021962496  0.04877214
## t5* -2.776035e-01  0.0006707451  0.03626421
## t6*  1.641309e-01  0.0011912137  0.03299041
## t7*  2.460243e-01  0.0060645856  0.04709809
## t8*  1.193177e-01  0.0034492458  0.03739097
```



3.1.2 Cross-Validation

为了解决上述问题，有人提出了 Cross-Validation 的方法。这是一种将样本数据切割成较小子集的方法，在部分子集中进行训练分析，其他子集则用来确认和验证，循环地将所有子集分析，可以得到一个较为稳健的估计。

3.1.2.1 Leave-One-Out Cross-Validation

Leave-One-Out Cross-Validation(LOOCV) 方法，也继承了上面的思路，但是不同的是，只用一个数据集作为验证集，其他数据作为训练集，并将此步骤重复 N 次（N 为数据集的数量）。而最终的 test 均方误差就是这 N 次训练结果的平均值。

$$CV_{(n)} = \frac{1}{n}\sum_{i=1}^{n} MSE_i$$

```
set.seed(123)
yname <- "z"
varnameList <- names(step.model$coefficients[-1])
formu <- as.formula(paste(yname, paste("~", paste(varnameList, collapse = "+"))))
ols.fit <- glm(formu, data = data3.trans)
cv.err <- cv.glm(data3.trans, ols.fit)
cv.err$delta
```

```
## [1] 0.1583323 0.1582820
```

3.1.2.2 K-fold Cross-Validation

另一种方法叫做 K-fold Cross-Validation，与 LOOCV 不同的是，每次的测试集不再只包含一个数据，而是 k 个数据，比如 k = 10，那么我们利用 10-fold 交叉验证的步骤，先将所有数据集分为 10 份，不重复地每次选取其中 1 份作为验证集，其他 9 份作为训练集训练模型，之后计算在该测试集的均方误差，最终将 10 次均方误差取评价。

$$CV_{(k)} = \frac{1}{k}\sum_{i=1}^{k} MSE_i$$

不难看出，LOOCV 是一种特殊的 K-fold Cross-Validation（K = 1）。

```
set.seed(123)
yname <- "z"
varnameList <- names(step.model$coefficients[-1])
formu <- as.formula(paste(yname, paste("~", paste(varnameList, collapse = "+"))))
ols.fit <- glm(formu, data = data3.trans)
cv.err <- cv.glm(data3.trans, ols.fit, K = 10)
cv.err$delta
```

```
## [1] 0.1583177 0.1573669
```

3.2 Lasso, Ridge and Elastic net

线性回归的最小二乘法，是为了使得目标函数最小化，即

$$\min_{\beta}\{||Y - X\beta||^2\}$$

Lasso, Ridge and Elastic net 这三个方法都是在回归过程中防止过拟合的出现的方法，在目标函数后添加正则项因子，其中 Lasso 是使用 $L1 - norm$ 正则因子，即

$$\min_{\beta}\{||Y - X\beta||^2 + \lambda||\beta||_1\}, \quad where\ ||\beta||_1 = \sum_i |\beta_i|,$$

Ridge 是使用 $L2 - norm$ 正则因子，即

$$\min_{\beta}\{||Y - X\beta||^2 + \lambda||\beta||_2\}, \quad where\ ||\beta||_2 = \sum_i \beta_i^2,$$

而 Elastic net 则是 $L1-norm$ 和 $L2-norm$ 的结合，即

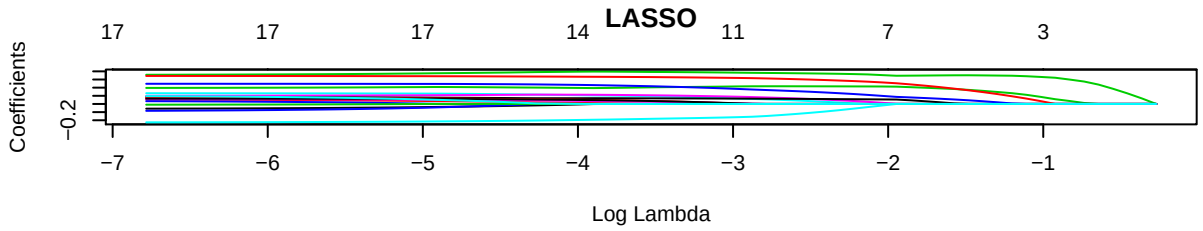
$$\min_{\beta}\{||Y-X\beta||^2+\lambda[\alpha||\beta||_1+(1-\alpha)||\beta||_2]\}.$$

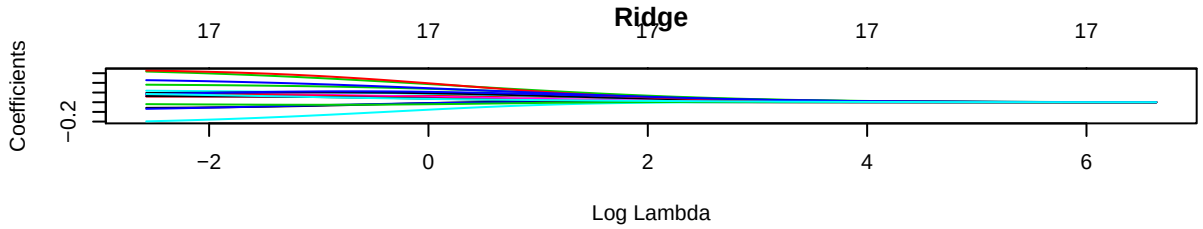
下面将结合这三种方法和 Cross-Validation 的方法进行分析。

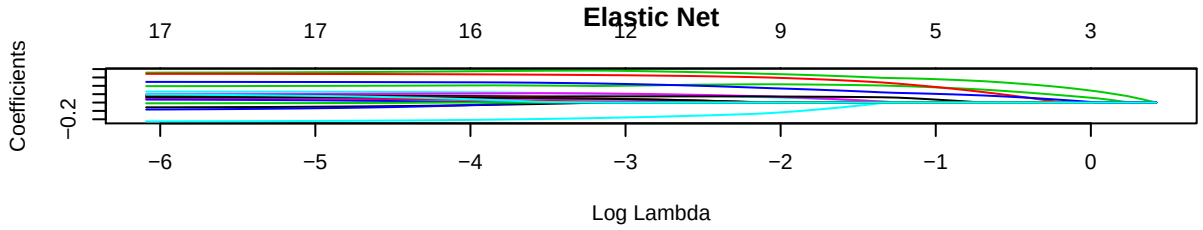
3.2.1 训练模型

我们使用的是 glmnet 这个包中的 glmnet 函数，其中的 alpha 参数也就是上述 Elastic net 中的 α。注意到 $\alpha=1$ 是 Lasso 模型，$\alpha=0$ 是 Ridge 模型。

```
fit.lasso <- glmnet(x.train, y.train, family="gaussian", alpha=1)
fit.ridge <- glmnet(x.train, y.train, family="gaussian", alpha=0)
fit.elnet <- glmnet(x.train, y.train, family="gaussian", alpha=.5)

# Plot solution paths:
```



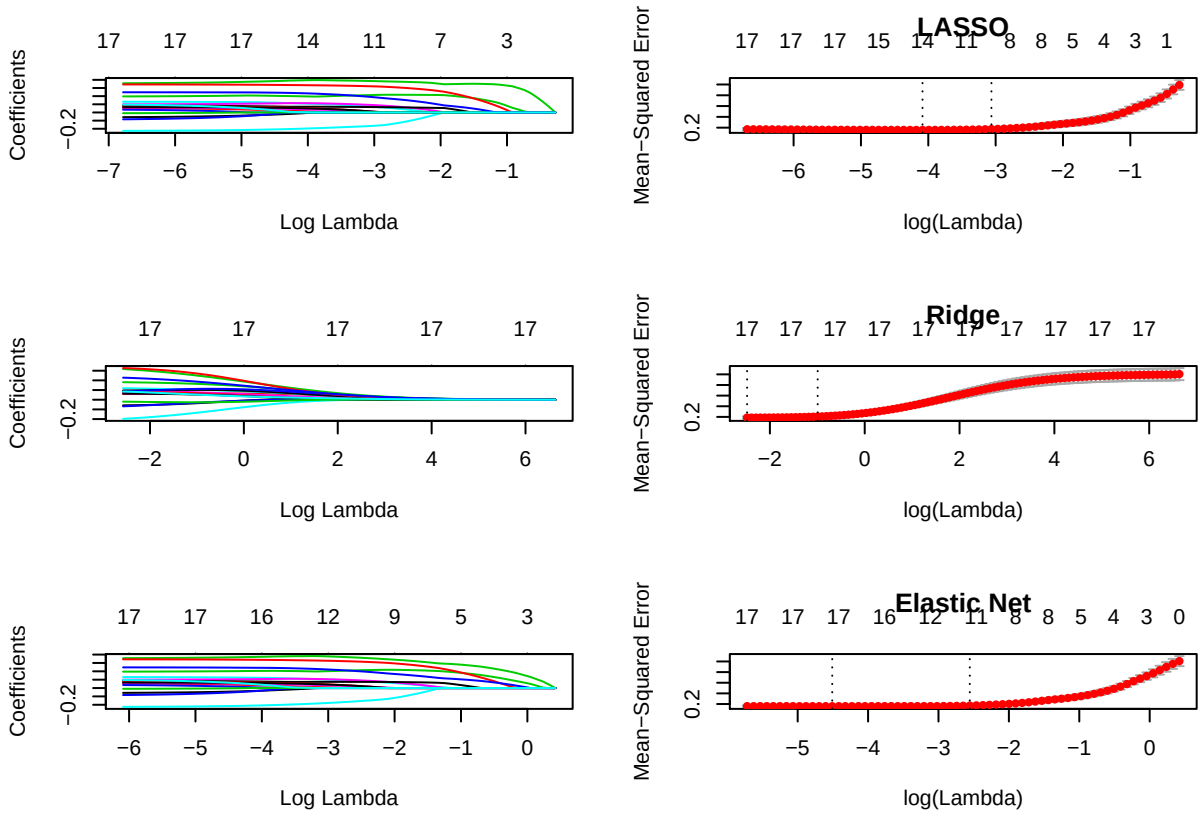




下面使用 10-Fold Cross-Validation 对 $\alpha=0,0.1,\ldots,0.9,1.0$ 进行分析。

```
set.seed(123)
for (i in 0:10) {
```

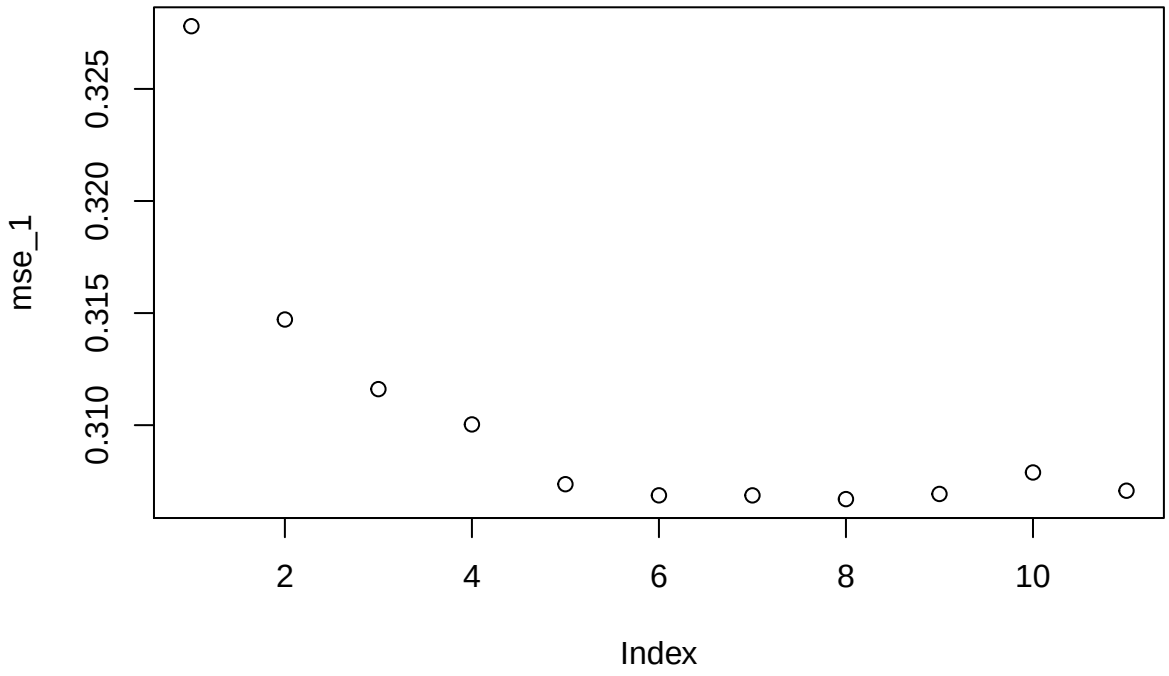
```
  assign(paste("fit", i, sep=""), cv.glmnet(x.train, y.train,
         type.measure="mse",  alpha=i/10,family="gaussian"))
}
```



3.2.2 预测

```
x.test <- as.matrix(data2.test)
y.test <- data.test$price
y.log.test <- (log(y.test) - y.center) / y.scale
yhat0 <- predict(fit0, s=fit0$lambda.1se, newx=x.test)
yhat1 <- predict(fit1, s=fit1$lambda.1se, newx=x.test)
yhat2 <- predict(fit2, s=fit2$lambda.1se, newx=x.test)
yhat3 <- predict(fit3, s=fit3$lambda.1se, newx=x.test)
yhat4 <- predict(fit4, s=fit4$lambda.1se, newx=x.test)
yhat5 <- predict(fit5, s=fit5$lambda.1se, newx=x.test)
yhat6 <- predict(fit6, s=fit6$lambda.1se, newx=x.test)
yhat7 <- predict(fit7, s=fit7$lambda.1se, newx=x.test)
yhat8 <- predict(fit8, s=fit8$lambda.1se, newx=x.test)
```

```
yhat9 <- predict(fit9, s=fit9$lambda.1se, newx=x.test)
yhat10 <- predict(fit10, s=fit10$lambda.1se, newx=x.test)

mse0 <- mean((y.log.test - yhat0)^2)
mse1 <- mean((y.log.test - yhat1)^2)
mse2 <- mean((y.log.test - yhat2)^2)
mse3 <- mean((y.log.test - yhat3)^2)
mse4 <- mean((y.log.test - yhat4)^2)
mse5 <- mean((y.log.test - yhat5)^2)
mse6 <- mean((y.log.test - yhat6)^2)
mse7 <- mean((y.log.test - yhat7)^2)
mse8 <- mean((y.log.test - yhat8)^2)
mse9 <- mean((y.log.test - yhat9)^2)
mse10 <- mean((y.log.test - yhat10)^2)
mse_1 = c(mse0,mse1,mse2,mse3,mse4,mse5,mse6,mse7,mse8,mse9,mse10)
```



我们可以看到，当 $\alpha = 0.7$ 的时候，均方误差最小。

```
which(mse_1 == min(mse_1))
```

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