# Regression analysis_Data Analysis Exam

**心理所碩二 R08227112 林子堯**

**2020/12/14**

Please follow the instructions below:

- training.csv is the dataset for regression analysis
- testing.csv are the “future” observations

The goals are to build a “best” regression model (1) to interpret the predictor variables concerning the response variable and (2) to predict the “future” observations. You may focus on linear regression models in this analysis. Please summarize your results with at least the following main items:

(You may compile all other supporting materials in the appendix.)

載入必要的套件和資料

```
library(tidyverse)
library(leaps)
```

```
training_data <- read.csv("training.csv")
testing_data <- read.csv("testing.csv")
head(training_data)
```

```
       Y   x1    x2    x3 x4    x5    x6     x7   x8
1  90.91 3.72 -2.61 11.20  0  2.28  2.06  30.33 0.92
2  59.18 3.90  4.62  1.12  1  1.41  1.18  22.14 0.54
3  94.71 4.80  1.01 11.13  0 -0.11 -0.41  10.63 0.85
4  84.82 4.07  0.15 10.48  0  0.70  0.37   8.51 0.58
5  74.51 4.11 -0.97  9.19  0 -0.34 -0.48  16.25 0.67
6 105.42 4.91  3.80  7.25  0 -0.52 -0.42 135.09 0.51
```

```
training_data$x4 <- factor(training_data$x4)
testing_data$x4 <- factor(testing_data$x4)
```

## 1. Select the “best” model among the possible candidates considered.

### (1) State your final regression model explicitly, including the model assumptions.

```
train_best <- lm(Y ~ x3 + x4 + x5 + x7 + x3:x4 + x3:x7 + x4:x7,
data = training_data)
summary(train_best)
```

```

Call:
lm(formula = Y ~ x3 + x4 + x5 + x7 + x3:x4 + x3:x7 + x4:x7, data 
= training_data)

Residuals:
    Min      1Q  Median      3Q     Max 
-73.147 -13.568   3.008  15.413  51.262 

Coefficients:
             Estimate Std. Error t value Pr(>|t|)    
(Intercept) 109.91931   18.00760   6.104 2.43e-08 ***
x3           -2.55139    1.83527  -1.390  0.16782    
x41         -42.49427   19.64092  -2.164  0.03309 *  
x5            2.51343    2.51828   0.998  0.32086    
x7           -0.41734    0.18833  -2.216  0.02916 *  
x3:x41        7.76604    2.92479   2.655  0.00934 ** 
x3:x7         0.05678    0.02221   2.556  0.01222 *  
x41:x7        0.82827    0.17209   4.813 5.80e-06 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 23.35 on 92 degrees of freedom
Multiple R-squared:  0.3875,    Adjusted R-squared:  0.3409 
F-statistic: 8.316 on 7 and 92 DF,  p-value: 7.74e-08
```

```
AIC(train_best)
```

```
[1] 923.536
```

```
BIC(train_best)
```

```
[1] 946.9826
```

這是我目前找到最佳的的模型為，

$$\hat{Y} = \hat{\beta}_0 + \hat{\beta}_1 X_3 + \hat{\beta}_2 X_4 + \hat{\beta}_3 X_5 + \hat{\beta}_4 X_7 + \hat{\beta}_5 X_3 X_4 + \hat{\beta}_6 X_3 X_7 + \hat{\beta}_6 X_4 X_7$$

其中模型的假設為：

- 所有資料點是獨立的
- 解釋變項 $\boldsymbol{X}$ 和 預測變相 $Y$ 之間有線性關係
- Additive error
- Errors $\varepsilon_i \sim iid N(0, \sigma)$，i = 1, ..., N，且具有 homoscedasticity 的性質

**(2) Briefly describe your model building procedure toward this final model. Give your reasons for choosing this model.**

我先透過不考慮有交互作用項的模型，並使用 R 原生套件 `stats` 逐步回歸的方式 `step()`，考慮 $Y = \beta_0 + \beta_1 x_1 + \ldots + \beta_8 x_8$ 下最佳模型，其中以 AIC 作為判准，得到的結果為

```
train_lm_all ← lm(Y ~ x1+x2+x3+x4+x5+x6+x7+x8, data = training_
data)
train_lm_step ← step(train_lm_all, direction = "both", trace =
0)
summary(train_lm_step)
```

```


Call:
lm(formula = Y ~ x3 + x4 + x5 + x7, data = training_data)

Residuals:
    Min      1Q  Median      3Q     Max 
-87.857 -13.343   1.806  17.090  53.379 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept) 58.13160   14.01773   4.147 7.33e-05 ***
x3           2.52230    1.27075   1.985 0.050041 .  
x41         36.73642    9.96584   3.686 0.000379 ***
x5           4.33340    2.78344   1.557 0.122832    
x7           0.13899    0.05448   2.551 0.012335 *  
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 26.36 on 95 degrees of freedom
Multiple R-squared:  0.1935,	Adjusted R-squared:  0.1595 
F-statistic: 5.698 on 4 and 95 DF,  p-value: 0.0003731
```

```
AIC(train_lm_step)
```

```
[1] 945.0579
```

```
BIC(train_lm_step)
```

```
[1] 960.6889
```

$$Y \sim x3 + x4 + x5 + x7$$

此時我再加入考慮所有的二階交互作用項，同樣是使用逐步回歸的方式，以 AIC 作為判准，得到的結果為

```
train_lm_2way <- lm(Y ~ (x1+x2+x3+x4+x5+x6+x7+x8)^2, data = training_data)
train_lm_2way_step <- step(train_lm_2way, direction = c("both"), trace = 0)
summary(train_lm_2way_step)
```

```

Call:
lm(formula = Y ~ x1 + x3 + x4 + x5 + x6 + x7 + x8 + x1:x5 + x1:x
6 +
    x1:x7 + x1:x8 + x3:x4 + x3:x7 + x3:x8 + x4:x7 + x4:x8 + x5:x
8 +
    x6:x7 + x7:x8, data = training_data)

Residuals:
    Min      1Q  Median      3Q     Max 
-49.735 -12.803   3.287  13.953  36.194 

Coefficients:
              Estimate Std. Error t value Pr(>|t|)    
(Intercept)  1.378e+02  5.378e+01   2.563  0.01226 *  
x1           4.141e-03  1.009e+01   0.000  0.99967    
x3          -4.767e+00  2.519e+00  -1.892  0.06205 .  
x41         -6.198e+01  2.252e+01  -2.752  0.00732 ** 
x5          -1.063e+02  6.769e+01  -1.570  0.12031    
x6           1.497e+02  7.228e+01   2.072  0.04151 *  
x7          -1.543e+00  6.089e-01  -2.534  0.01324 *  
x8           4.506e+01  8.282e+01   0.544  0.58792    
x1:x5        2.889e+01  1.507e+01   1.917  0.05885 .  
x1:x6       -3.556e+01  1.616e+01  -2.200  0.03068 *  
x1:x7        2.082e-01  1.314e-01   1.585  0.11699    
x1:x8       -2.459e+01  1.701e+01  -1.445  0.15224    
x3:x41       7.585e+00  3.082e+00   2.461  0.01600 *  
x3:x7        6.159e-02  2.802e-02   2.198  0.03083 *  
x3:x8        5.179e+00  3.879e+00   1.335  0.18557    
x41:x7       8.611e-01  1.993e-01   4.320 4.44e-05 ***
x41:x8       4.496e+01  2.836e+01   1.585  0.11687    
x5:x8       -1.670e+01  9.765e+00  -1.710  0.09109 .  
x6:x7       -8.258e-02  5.563e-02  -1.484  0.14161    
x7:x8        3.245e-01  2.442e-01   1.329  0.18769    
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 22.38 on 80 degrees of freedom
Multiple R-squared:  0.5108,	Adjusted R-squared:  0.3946 
F-statistic: 4.397 on 19 and 80 DF,  p-value: 1.386e-06
```

```
AIC(train_lm_2way_step)
```

```
[1] 925.0646
```

```
BIC(train_lm_2way_step)
```

```
[1] 979.7731
```

如上所示，雖然此解果 AIC 明顯小於原先沒有考慮交互作用項的模型，但是各個 $\beta$ 都不一定有顯著，這樣難以後續的解釋。因此我先拿掉表現最差的 $x8$，

```
train_lm_no8 ← lm(Y ~ x1 + x3 + x4 + x5 + x6 + x7 + x1:x5 + x1:
x6 + x1:x7  + x3:x4 + x3:x7  + x4:x7 + x6:x7,
    data = training_data)
summary(train_lm_no8)
```

```


Call:
lm(formula = Y ~ x1 + x3 + x4 + x5 + x6 + x7 + x1:x5 + x1:x6 + 
    x1:x7 + x3:x4 + x3:x7 + x4:x7 + x6:x7, data = training_data)

Residuals:
    Min      1Q  Median      3Q     Max 
-55.867 -13.205   2.468  14.388  44.828 

Coefficients:
              Estimate Std. Error t value Pr(>|t|)    
(Intercept)  171.68167   35.12632   4.888 4.69e-06 ***
x1           -11.07586    6.58758  -1.681  0.09633 .  
x3            -4.06006    2.04102  -1.989  0.04985 *  
x41          -51.26620   20.41481  -2.511  0.01390 *  
x5          -110.78322   67.14646  -1.650  0.10262    
x6           142.55160   71.99541   1.980  0.05090 .  
x7            -1.44037    0.57995  -2.484  0.01495 *  
x1:x5         28.36535   14.95273   1.897  0.06118 .  
x1:x6        -34.54189   16.08579  -2.147  0.03458 *  
x1:x7          0.18987    0.13018   1.459  0.14834    
x3:x41         8.04136    2.91340   2.760  0.00706 ** 
x3:x7          0.08033    0.02729   2.944  0.00417 ** 
x41:x7         0.93605    0.19142   4.890 4.64e-06 ***
x6:x7         -0.06275    0.04275  -1.468  0.14581    
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 22.69 on 86 degrees of freedom
Multiple R-squared:  0.4591,	Adjusted R-squared:  0.3774 
F-statistic: 5.616 on 13 and 86 DF,  p-value: 2.676e-07
```

```
AIC(train_lm_no8)
```

```
[1] 923.1033
```

```
BIC(train_lm_no8)
```

```
[1] 962.1808
```

```
anova(train_lm_no8, train_lm_2way_step)
```

```
Analysis of Variance Table

Model 1: Y ~ x1 + x3 + x4 + x5 + x6 + x7 + x1:x5 + x1:x6 + x1:x7
+ x3:x4 +
    x3:x7 + x4:x7 + x6:x7
Model 2: Y ~ x1 + x3 + x4 + x5 + x6 + x7 + x8 + x1:x5 + x1:x6 +
x1:x7 +
    x1:x8 + x3:x4 + x3:x7 + x3:x8 + x4:x7 + x4:x8 + x5:x8 + x6:x
7 +
    x7:x8
  Res.Df   RSS Df Sum of Sq      F Pr(>F)
1     86 44282
2     80 40052  6    4229.5 1.408 0.2218
```

與先前模型沒有顯著的差異，而且 AIC BIC 還便小了。因次我繼續拿掉一些可能不重要的變量。經過幾論嘗試後，下方可能是我目前找到的最佳模型。

```
train_lm_no8_no1and7 ← lm(Y ~ x3 + x4 + x5 +  x7 + x3:x4 + x3:x
7  + x4:x7 ,
    data = training_data)
summary(train_lm_no8_no1and7)
```

```

Call:
lm(formula = Y ~ x3 + x4 + x5 + x7 + x3:x4 + x3:x7 + x4:x7, data
= training_data)

Residuals:
    Min      1Q  Median      3Q     Max 
-73.147 -13.568   3.008  15.413  51.262 

Coefficients:
             Estimate Std. Error t value Pr(>|t|)    
(Intercept) 109.91931   18.00760   6.104 2.43e-08 ***
x3           -2.55139    1.83527  -1.390  0.16782    
x41         -42.49427   19.64092  -2.164  0.03309 *  
x5            2.51343    2.51828   0.998  0.32086    
x7           -0.41734    0.18833  -2.216  0.02916 *  
x3:x41        7.76604    2.92479   2.655  0.00934 ** 
x3:x7         0.05678    0.02221   2.556  0.01222 *  
x41:x7        0.82827    0.17209   4.813 5.80e-06 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 23.35 on 92 degrees of freedom
Multiple R-squared:  0.3875,	Adjusted R-squared:  0.3409 
F-statistic: 8.316 on 7 and 92 DF,  p-value: 7.74e-08
```

```
AIC(train_lm_no8_no1and7)
```

```
[1] 923.536
```

```
BIC(train_lm_no8_no1and7)
```

```
[1] 946.9826
```

另一方面，我用運立一個 R 套件幫我尋找出前十五個較佳模型，他可以用 adjust $R^2$, Mallos Cp 和 BIC 的指標幫我做判准，但一樣我這邊也只考慮到二階交互作用項而已。其結果為

```
train_lm_temp <- regsubsets(Y ~ (x1+x2+x3+x4+x5+x6+x7+x8)^2,
                            training_data,
                            nvmax=15,
                            method='backward')

(train_lm_temp_smry <- summary(train_lm_temp))
```

```
Subset selection object
Call: regsubsets.formula(Y ~ (x1 + x2 + x3 + x4 + x5 + x6 + x7 +
x8)^2,
    training_data, nvmax = 15, method = "backward")
36 Variables  (and intercept)
       Forced in Forced out
x1         FALSE      FALSE
x2         FALSE      FALSE
x3         FALSE      FALSE
x41        FALSE      FALSE
x5         FALSE      FALSE
x6         FALSE      FALSE
x7         FALSE      FALSE
x8         FALSE      FALSE
x1:x2      FALSE      FALSE
x1:x3      FALSE      FALSE
x1:x41     FALSE      FALSE
x1:x5      FALSE      FALSE
x1:x6      FALSE      FALSE
x1:x7      FALSE      FALSE
x1:x8      FALSE      FALSE
x2:x3      FALSE      FALSE
x2:x41     FALSE      FALSE
x2:x5      FALSE      FALSE
x2:x6      FALSE      FALSE
x2:x7      FALSE      FALSE
x2:x8      FALSE      FALSE
x3:x41     FALSE      FALSE
x3:x5      FALSE      FALSE
x3:x6      FALSE      FALSE
x3:x7      FALSE      FALSE
x3:x8      FALSE      FALSE
x41:x5     FALSE      FALSE
x41:x6     FALSE      FALSE
x41:x7     FALSE      FALSE
x41:x8     FALSE      FALSE
x5:x6      FALSE      FALSE
x5:x7      FALSE      FALSE
x5:x8      FALSE      FALSE
x6:x7      FALSE      FALSE
x6:x8      FALSE      FALSE
x7:x8      FALSE      FALSE
1 subsets of each size up to 15
Selection Algorithm: backward
          x1  x2  x3  x41 x5  x6  x7  x8  x1:x2 x1:x3 x1:x41 x1:
x5 x1:x6 x1:x7
1  ( 1 )  " " " " " " " " " " " " " " " " " "   " "   " "    " "
```

```
" "    " "
2  ( 1 )  " " " " " " " " " " " " " " " " " "   " "   "*"    " "
" "    " "
3  ( 1 )  " " " " " " " " " " " " " " " " " "   " "   "*"    " "
" "    " "
4  ( 1 )  " " " " " " " " " " " " " " " " " "   " "   "*"    "*"
" "    " "
5  ( 1 )  " " " " " " " " " " " " " " " " " "   " "   "*"    "*"
" "    " "
6  ( 1 )  " " " " " " " " " " " " " " " " " "   " "   "*"    "*"
"*"    " "
7  ( 1 )  " " " " " " " " " " "*" " " " " " "   " "   "*"    "*"
"*"    " "
8  ( 1 )  " " " " " " " " " " "*" " " " " " "   " "   "*"    "*"
"*"    " "
9  ( 1 )  " " " " " " " " " " "*" "*" " " " "   " "   "*"    "*"
"*"    " "
10  ( 1 ) " " " " " " " " " " "*" "*" " " " "   "*"   "*"    "*"
"*"    " "
11  ( 1 ) "*" " " " " " " " " "*" "*" " " " "   "*"   "*"    "*"
"*"    " "
12  ( 1 ) "*" " " " " " " " " "*" "*" " " " "   "*"   "*"    "*"
"*"    " "
13  ( 1 ) "*" " " " " " " " " "*" "*" " " " "   "*"   "*"    "*"
"*"    " "
14  ( 1 ) "*" " " " " " " " " "*" "*" " " " "   "*"   "*"    "*"
"*"    " "
15  ( 1 ) "*" " " " " " " "*" "*" "*" " " " "   "*"   "*"    "*"
"*"    " "
          x1:x8 x2:x3 x2:x41 x2:x5 x2:x6 x2:x7 x2:x8 x3:x41 x3:x
5 x3:x6 x3:x7
1  ( 1 )  " "   " "   " "    " "   " "   " "   " "   " "    " "
" "    " "
2  ( 1 )  " "   " "   " "    " "   " "   " "   " "   " "    " "
" "    " "
3  ( 1 )  " "   " "   " "    " "   " "   " "   " "   "*"    " "
" "    " "
4  ( 1 )  " "   " "   " "    " "   " "   " "   " "   "*"    " "
" "    " "
5  ( 1 )  " "   " "   " "    " "   " "   " "   " "   "*"    " "
" "    " "
6  ( 1 )  " "   " "   " "    " "   " "   " "   " "   "*"    " "
" "    " "
7  ( 1 )  " "   " "   " "    " "   " "   " "   " "   "*"    " "
" "    " "
8  ( 1 )  " "   " "   " "    " "   " "   " "   " "   "*"    " "
" "    "*"
```

```
9  ( 1 )  " "   " "   " "    " "   " "   " "   " "   "*"    " "
" "   "*"
10  ( 1 ) " "   " "   " "    " "   " "   " "   " "   "*"    " "
" "   "*"
11  ( 1 ) " "   " "   " "    " "   " "   " "   " "   "*"    " "
" "   "*"
12  ( 1 ) " "   " "   " "    " "   " "   " "   " "   "*"    " "
" "   "*"
13  ( 1 ) "*"   " "   " "    " "   " "   " "   " "   "*"    " "
" "   "*"
14  ( 1 ) "*"   " "   " "    " "   " "   " "   " "   "*"    " "
" "   "*"
15  ( 1 ) "*"   " "   " "    " "   " "   " "   " "   "*"    " "
" "   "*"
          x3:x8 x41:x5 x41:x6 x41:x7 x41:x8 x5:x6 x5:x7 x5:x8 x
6:x7 x6:x8 x7:x8
1  ( 1 )  " "   " "    " "    "*"    " "    " "   " "   " "   "
"   " "   " "
2  ( 1 )  " "   " "    " "    "*"    " "    " "   " "   " "   "
"   " "   " "
3  ( 1 )  " "   " "    " "    "*"    " "    " "   " "   " "   "
"   " "   " "
4  ( 1 )  " "   " "    " "    "*"    " "    " "   " "   " "   "
"   " "   " "
5  ( 1 )  " "   " "    " "    "*"    " "    " "   " "   "*"   "
"   " "   " "
6  ( 1 )  " "   " "    " "    "*"    " "    " "   " "   "*"   "
"   " "   " "
7  ( 1 )  " "   " "    " "    "*"    " "    " "   " "   "*"   "
"   " "   " "
8  ( 1 )  " "   " "    " "    "*"    " "    " "   " "   "*"   "
"   " "   " "
9  ( 1 )  " "   " "    " "    "*"    " "    " "   " "   "*"   "
"   " "   " "
10  ( 1 ) " "   " "    " "    "*"    " "    " "   " "   "*"   "
"   " "   " "
11  ( 1 ) " "   " "    " "    "*"    " "    " "   " "   "*"   "
"   " "   " "
12  ( 1 ) " "   " "    " "    "*"    "*"    " "   " "   "*"   "
"   " "   " "
13  ( 1 ) " "   " "    " "    "*"    "*"    " "   " "   "*"   "
"   " "   " "
14  ( 1 ) "*"   " "    " "    "*"    "*"    " "   " "   "*"   "
"   " "   " "
15  ( 1 ) "*"   " "    " "    "*"    "*"    " "   " "   "*"   "
"   " "   " "
```

```
which.min(train_lm_temp_smry$cp)
```

```
[1] 9
```

```
train_lm_temp_smry$cp[9]
```

```
[1] -0.347286
```

```
which.max(train_lm_temp_smry$adjr2)
```

```
[1] 15
```

```
train_lm_temp_smry$adjr2[15]
```

```
[1] 0.3957109
```

```
which.min(train_lm_temp_smry$bic)
```

```
[1] 3
```

```
train_lm_temp_smry$bic[3]
```

```
[1] -22.90053
```

然而這邊比較可惜的是，雖然有找到幾格不錯的模型，但是他會都保瞭二階交互作用項而忽略掉低階項，因此在這邊我就不先考慮了。

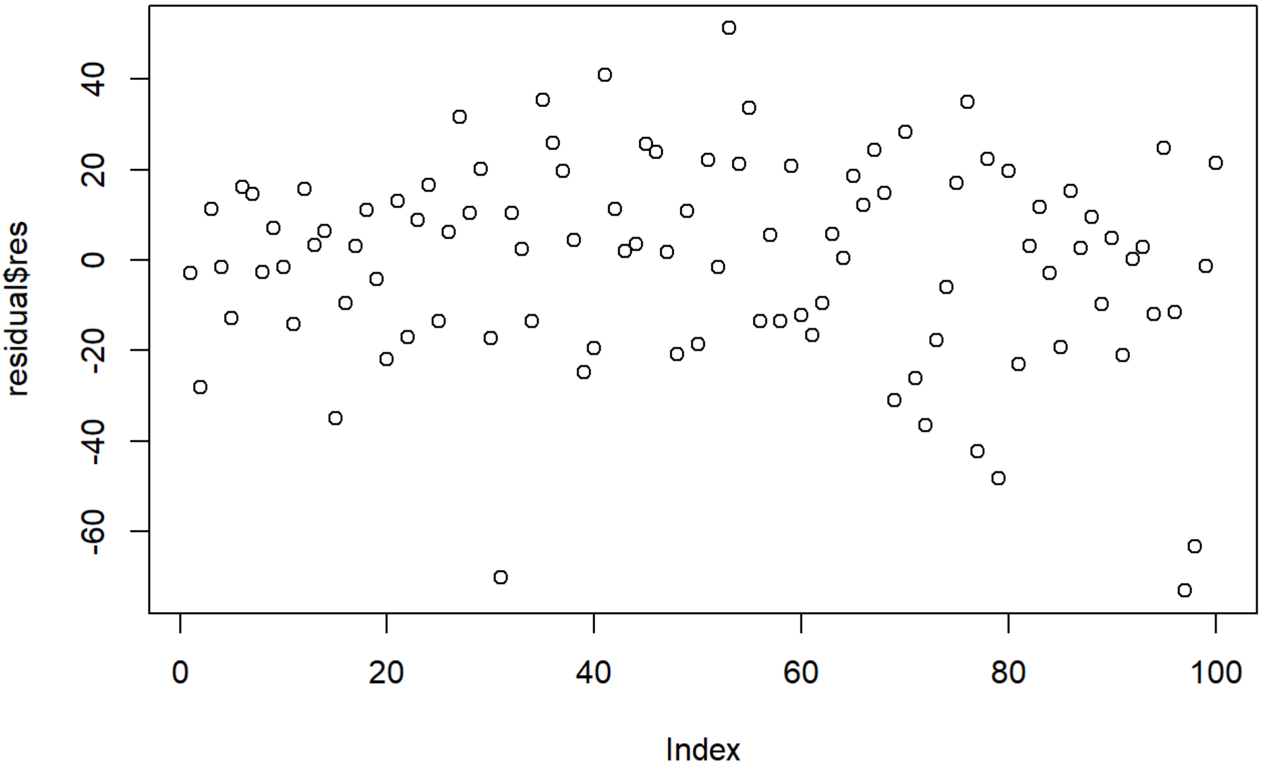
### 2. Show the validity of your final model by demonstrating the residual analysis for model check.

```
residual <- data.frame(
  res = residuals(train_best),
  stand_res = rstandard(train_best),
  student_res = rstudent(train_best),
  fit = fitted(train_best)
)
```

#### Independent

首先我們先檢查 residual 會不會隨著資料的排序有系統性的關係

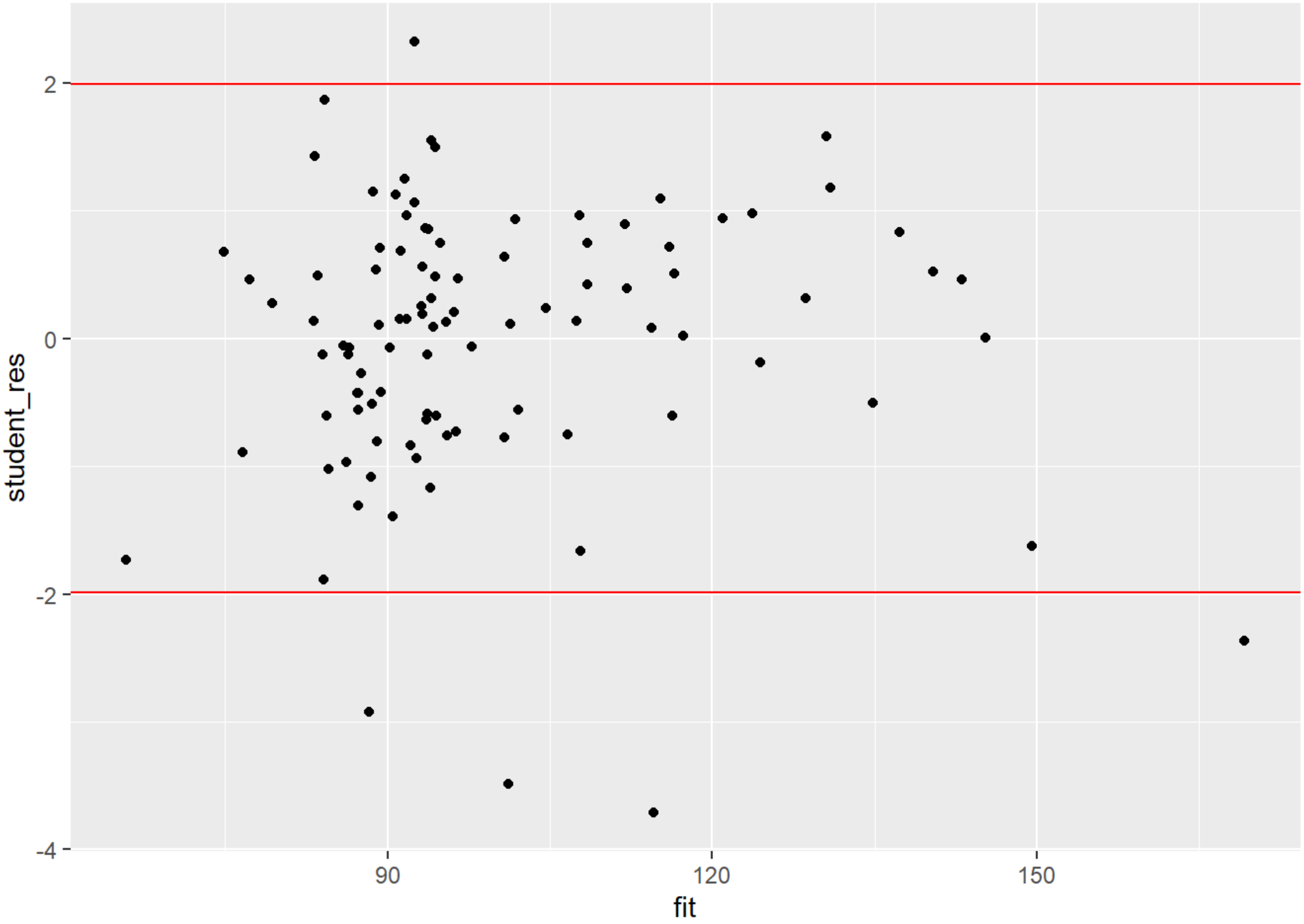
```
plot(residual$res)
```



上圖中，沒有發顯 residual 有任何不獨立或自回歸的狀況。

**Outlier**

```
ggplot(residual, aes(x = fit, y = student_res)) +
  geom_point() +
  geom_hline(yintercept = qt(c(0.025, 0.975), train_best$df.residua
l - 1), color = "red")
```



這邊檢查了 studentized residual，發現有少數幾點可能是 outliers 的傾向，但在這邊沒有資料點的詳細資訊，因此判定為 outlier 還有帶保留

**Homoscedasticity**

在此檢查 residuals 有沒有違反 homoscedasticity 的假設

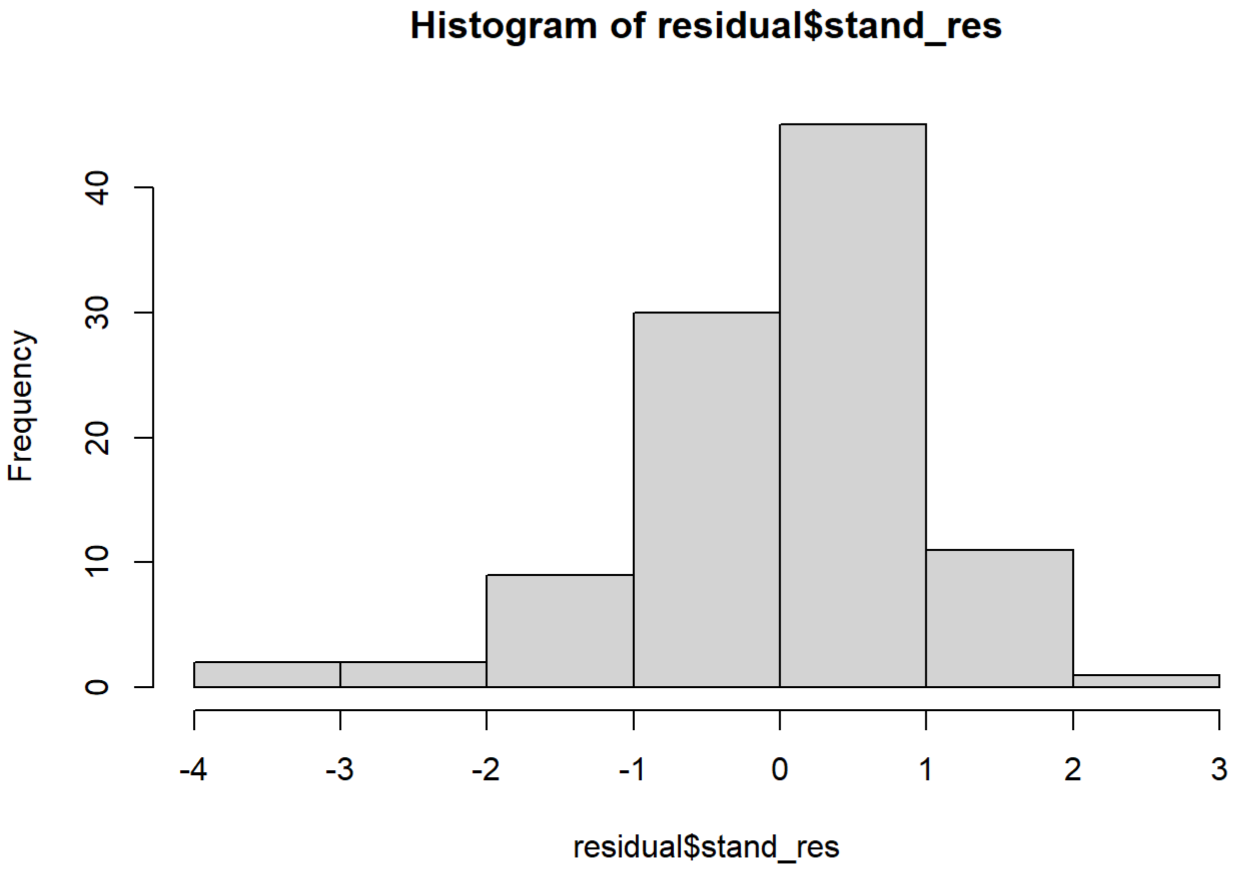
```
library(lmtest)
bptest(train_best, studentize = TRUE)
```

```


	studentized Breusch-Pagan test

data:  train_best
BP = 9.2908, df = 7, p-value = 0.2324
```

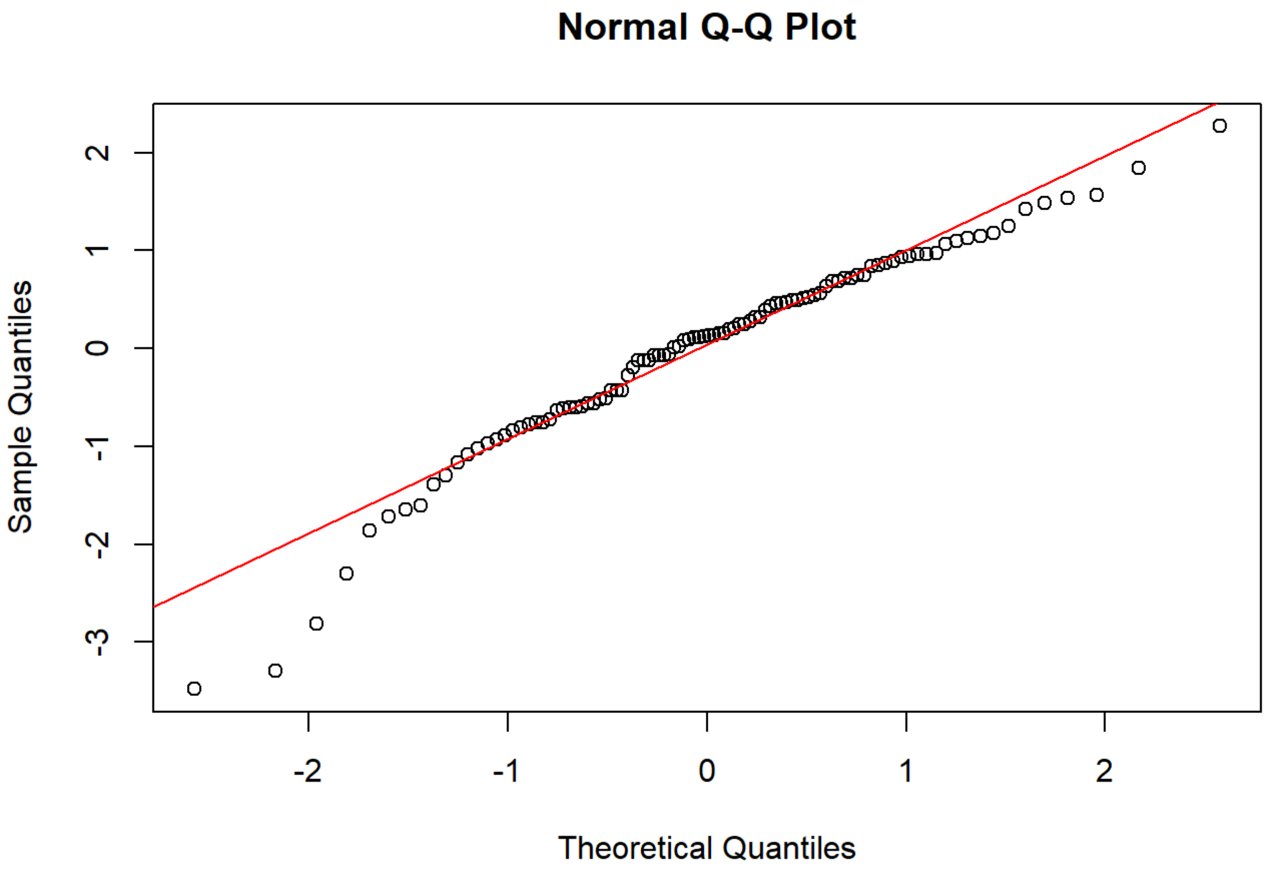
透過 Breusch–Pagan test 得到 p-value > .05，沒有拒絕虛無假設，說明此筆資料應該是沒有違反 homoscedasticity 的假設。

**Normality**

```
hist(residual$stand_res, bins = 30)
```



```
qqnorm(residual$stand_res)
qqline(residual$stand_res, col = "red")
```



大部分的點坐落在紅色的斜直線上，顯示與 normal distribution 接近。然而在左尾的部分可能要比真實的常態分配要來的厚，因此 normality assumption 可能需要注意！

### 3. Calculate the leave-one-out cross-validation prediction errors based on your final model.

我們知道 leave-one-out cross-validation prediction errors 的公式為：

$$CV = \frac{1}{n}\sum_{i=1}^{n}(\frac{y_i - \hat{y}_i}{1 - h_{ii}})^2$$

```
h <- hatvalues(train_best)
Y <- training_data$Y
Y_hat <- fitted(train_best)
n <- nrow(training_data)

CV <- 1/n * sum( ( (Y-Y_hat)/(1-h) )^2)
CV
```

```
[1] 680.9999
```

得到的 CV 為 $CV \approx 681$。

### 4. Use the test data set to calculate the squared prediction errors (sum) based on your final model. Display the prediction outcome by plotting predicted vs. observed.

```
test_predict <- data.frame(
  predict_value = predict(train_best, testing_data),
  true_value = testing_data$Y
)

test_predict <- test_predict %>%
  mutate(predict_error = predict_value - true_value)

(SSE_predict <- sum((test_predict$predict_error)^2))
```

```
[1] 9859.907
```

```
(MSE_predict <- SSE_predict / nrow(testing_data))
```

```
[1] 328.6636
```

```
(RMSE_predict <- sqrt(MSE_predict))
```

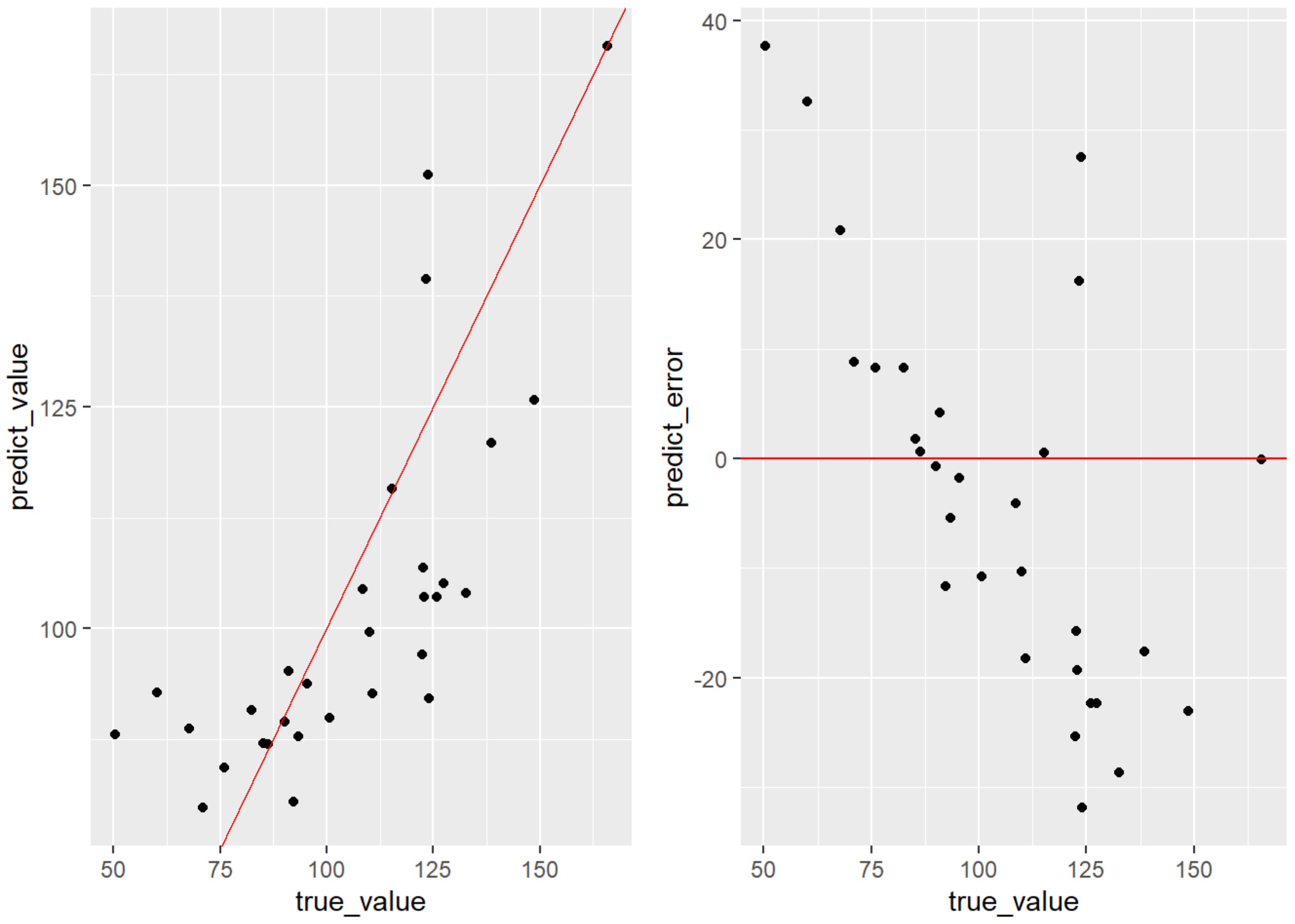
```
[1] 18.12908
```

結果為

- Sum of squared (prediction) errors = 9859.91
- Root mean square (prediction) errors = 18.13

另一方面，

```
g1 <- ggplot(test_predict, aes(x = true_value, y = predict_value))
+
  geom_point() +
  geom_abline(slope = 1, intercept = 0, color = "red")
g2 <- ggplot(test_predict, aes(x = true_value, y = predict_error))
 +
  geom_point() +
  geom_hline(yintercept = 0, color = "red")
gridExtra::grid.arrange(g1, g2, nrow = 1)
```



左圖為 testing data 的真實值 $Y$ 與用我目前模型所得到的預測值 $\hat{Y}$，可觀察到預測值與真實值有相似的趨勢，$Y$ 越高 $\hat{Y}$ 也越高，且大致落在斜率為1的斜直線上。然而我們看到右圖，為 prediction error $\hat{Y} - Y$ 與 $Y$ 的散佈圖，雖然預測誤差坐落在 0 的上下，然而很明顯的可以發現還是有系統信的誤差存在，當 $Y$ 低的時候 $\hat{Y}$ 有高估，而當 $Y$ 高的時候 $\hat{Y}$ 有低估的現象產生。因此推測其實我目前的到的「最佳」模型，其實還是少了一些重要的解釋變項在裡頭，可能被我給忽略了。