Multi-Linear Regression

Rui Zhou

目录

```
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

0.1 这章在讲什么

多元线性回归（multiple linear regression, MLR）用多个解释变量 $X_1, \dots, X_k$ 一起预测一个连续因变量 Y。心理学最常见的两种用法：(1) 控制混杂变量，比如想看「干预 → 学习成绩」的净效应，就把 IQ、动机、家庭背景一起塞进回归当协变量；(2) 比较解释力，看哪个变量对结果的「贡献」最大（这需要看标准化系数而非原始系数，下文会讲）。

核心模型与 OLS

模型形式：$Y_i = \beta_0 + \beta_1 X_{1i} + \cdots + \beta_k X_{ki} + \varepsilon_i$。OLS 估计的本质是最小化残差平方和 $\sum_i \hat{\varepsilon}_i^2$，闭式解 $\hat{\beta} = (X'X)^{-1}X'Y$。可识别需要 $X'X$ 可逆——这就是为什么**多重共线性**（某列能被其他列线性表出）会让 OLS 崩。

斜率 β_j 的解读：「在保持其他变量不变的情况下，X_j 增加 1 单位，Y 平均改变 β_j」。注意「保持其他变量不变」这件事经常做不到——这正是**偏相关**和后面要讲的 ANCOVA、Mediation 的出发点。

下面这份 `df` 是 SPSS 自带的雇员信息数据，含工资 `salary`、起薪 `salbegin`、受教育年数 `educ`、工龄 `jobtime`、过往工作经验 `prevexp` 等。我们想用其他变量预测 `salary`。

```
df = readxl::read_excel("data02-01.xlsx", sheet = 1)
attach(df)
```

1 Pre-Inspection

1.1 linear relationship

Use `cor()` to carry the correlation test, the default method of which is `method = 'pearson'`. The result will be presented as a correlation matrix.

However, obviously the correlation matrix won't give you information about p-value (or say, significance).

Good thing is that the check of linear relationship is the first and rough step for further regression, maybe other more information (about p-value or CIs) is unnecessary. Another good thing is that the `cor()` is from base R, no extra requirement to library packages.

```
df |> select(salary,educ,salbegin,jobtime,prevexp) |> cor()
```

```
              salary        educ    salbegin      jobtime      prevexp
salary    1.00000000  0.66055891  0.88011747  0.084092267 -0.097466926
educ      0.66055891  1.00000000  0.63319565  0.047378777 -0.252352521
salbegin  0.88011747  0.63319565  1.00000000 -0.019753475  0.045135627
jobtime   0.08409227  0.04737878 -0.01975347  1.000000000  0.002978134
prevexp  -0.09746693 -0.25235252  0.04513563  0.002978134  1.000000000
```

Well, it's clumsy to compress the target data (variables) into a matrix to take the test and get the correlation matrix. Just fine to know a bit about that.

```
# first create an empty matrix, primarily filled with NA
m = matrix(nrow=length(salary),ncol=5)
m[,1]=salary
m[,2]=educ
```

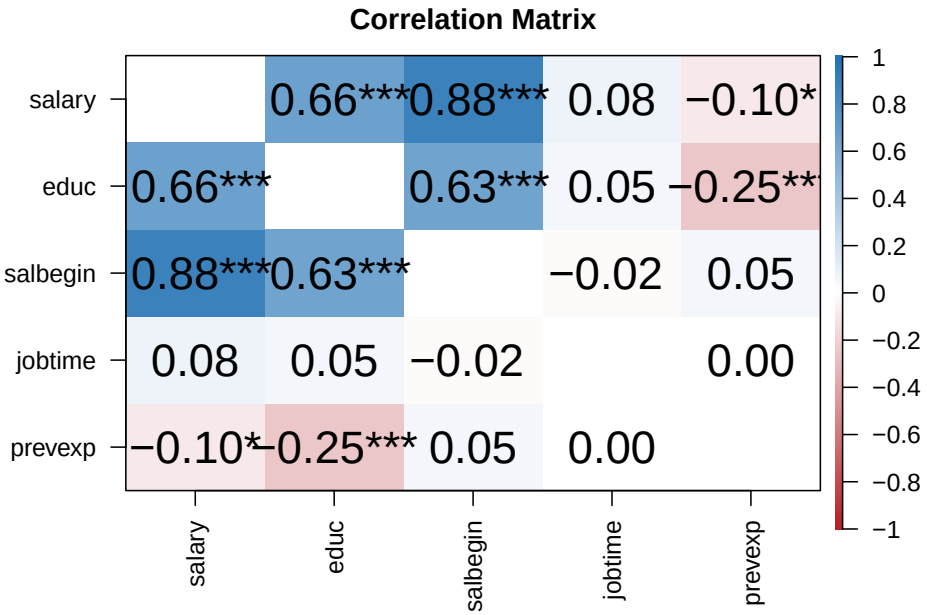
```
m[,3] = salbegin
m[,4]=jobtime
m[,5]=prevexp
#check the correlation matrix
cor(m)
```

```
             [,1]        [,2]        [,3]         [,4]         [,5]
[1,]  1.00000000  0.66055891  0.88011747  0.084092267 -0.097466926
[2,]  0.66055891  1.00000000  0.63319565  0.047378777 -0.252352521
[3,]  0.88011747  0.63319565  1.00000000 -0.019753475  0.045135627
[4,]  0.08409227  0.04737878 -0.01975347  1.000000000  0.002978134
[5,] -0.09746693 -0.25235252  0.04513563  0.002978134  1.000000000
```

Additionally, I'd like to use `bruceR::Corr()` for the correlation matrix, which is visible for both in amount and in significance. Moreover, it's fancy!

For simplicity, the two decimals by default is well enough. If more accuracy is needed, you can set the parameter `digits =`.

```
df |> select(salary,educ,salbegin,jobtime,prevexp) |> bruceR::Corr(digits = 2)
```



```
Correlation matrix is displayed in the RStudio `Plots` Pane.

Pearson's r and 95% confidence intervals:

                      r       [95% CI]     p       N

salary-educ        0.66 [ 0.61,  0.71] <.001 *** 474
salary-salbegin    0.88 [ 0.86,  0.90] <.001 *** 474
salary-jobtime     0.08 [-0.01,  0.17]  .067 .   474
salary-prevexp    -0.10 [-0.19, -0.01]  .034 *   474
educ-salbegin      0.63 [ 0.58,  0.68] <.001 *** 474
educ-jobtime       0.05 [-0.04,  0.14]  .303     474
educ-prevexp      -0.25 [-0.33, -0.17] <.001 *** 474
salbegin-jobtime  -0.02 [-0.11,  0.07]  .668     474
salbegin-prevexp   0.05 [-0.05,  0.13]  .327     474
jobtime-prevexp    0.00 [-0.09,  0.09]  .948     474
```

1.2 multicollinearity

Use `car::vif(model)` and get the result directly.

For the convenience of narration, check of multicollinearity is presented here. Most of the time, the check is a "post-hoc" one, after obtaining the fitted model.

VIF less than 2 is ideal, and less than 5 is the bottom line.

```
fit = lm(salary ~ educ + salbegin + jobtime + prevexp,data = df)
car::vif(fit)
```

```
    educ salbegin  jobtime  prevexp
1.937010 1.813582 1.007613 1.155814
```

2 MLR

2.1 basic summary

Firstly, use `lm()` to get the linear model.

Subsequently, use `summary()` to get the detailed result.

Period.

```
fit = lm(salary ~ educ + salbegin + jobtime + prevexp,data = df)
summary(fit)
```

```
Call:
lm(formula = salary ~ educ + salbegin + jobtime + prevexp, data = df)

Residuals:
   Min     1Q Median     3Q    Max
-29600  -4119  -1246   2642  46079

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept) -1.615e+04  3.255e+03  -4.961 9.84e-07 ***
educ         6.699e+02  1.656e+02   4.045 6.11e-05 ***
salbegin     1.768e+00  5.873e-02  30.111  < 2e-16 ***
jobtime      1.615e+02  3.425e+01   4.715 3.19e-06 ***
prevexp     -1.730e+01  3.528e+00  -4.904 1.30e-06 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 7465 on 469 degrees of freedom
Multiple R-squared:  0.8105,    Adjusted R-squared:  0.8089
F-statistic: 501.5 on 4 and 469 DF,  p-value: < 2.2e-16
```

It's noteworthy that `bruceR::print_table(model)` can spring up the re-

gression result as a decent table, and can be exported to a `.doc` file.

However, fine with `summary()`, decency doesn't matter much here, and the `summary()` can give more information about the model itself that is essential, such F value and multiple R squared.

2.2 coefficients

Use `coefficients(model)` from base R to see the coefficients for each IV.

Furthermore, use `confint(model,level = 0.95)` from base R to see the CIs (95% by default) for each coefficients.

```
coefficients(fit)
```

```
##   (Intercept)          educ      salbegin       jobtime       prevexp
## -16149.671204    669.913570      1.768427    161.485642    -17.303251
```

```
confint(fit, level=0.95)
```

```
##                     2.5 %       97.5 %
## (Intercept) -22546.784635 -9752.557773
## educ           344.511104   995.316035
## salbegin         1.653019     1.883835
## jobtime         94.190179   228.781104
## prevexp        -24.236663   -10.369839
```

Multiple R squared is reported through `summary()`.

By definition, Multiple R squared is defined as the square of the correlation between estimated (fitted) value and real values.

```
cor(salary,fit$fitted.values)
```

```
[1] 0.9002721
```

2.3 model comparison

Compare the model with the "zero" model, formally called the null model. The null model has no IV but with intercept, which is the mean. The null model uses the mean to predict the data or explain the variation of values.

Get the null model by formula `DV~1`.

```
fit2 = lm(salary ~ 1 )
```

Then use `anova(model1,model2)` to compare the two models.

```
anova(fit,fit2)
```

```
Analysis of Variance Table

Model 1: salary ~ educ + salbegin + jobtime + prevexp
Model 2: salary ~ 1
  Res.Df        RSS Df   Sum of Sq      F    Pr(>F)
1    469 2.6137e+10
2    473 1.3792e+11 -4 -1.1178e+11 501.45 < 2.2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

If we think further, the F test for the MLR model is to test with null hypothesis that all the coefficients of all the IVs are zero. Therefore, the model comparison by essence coincides with the F test! Obviously, the F value "copies" that in the `summary()` result.

3 Standard MLR

Transform the formula with each variable embedded into the function `scale()`, no other adjustments!

Under standard MLR, the coefficients are standardized ones.

```
fit_std = lm(scale(salary) ~ scale(educ) + scale(salbegin) + scale(jobtime) + scale(pre
```

```
summary(fit_std)
```

```
Call:
lm(formula = scale(salary) ~ scale(educ) + scale(salbegin) +
    scale(jobtime) + scale(prevexp), data = df)

Residuals:
     Min       1Q   Median       3Q      Max
-1.73349 -0.24121 -0.07298  0.15475  2.69850

Coefficients:
                  Estimate Std. Error t value Pr(>|t|)
(Intercept)     -1.129e-16  2.008e-02   0.000        1
scale(educ)      1.132e-01  2.798e-02   4.045 6.11e-05 ***
scale(salbegin)  8.151e-01  2.707e-02  30.111  < 2e-16 ***
scale(jobtime)   9.515e-02  2.018e-02   4.715 3.19e-06 ***
scale(prevexp)  -1.060e-01  2.161e-02  -4.904 1.30e-06 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.4372 on 469 degrees of freedom
Multiple R-squared:  0.8105,    Adjusted R-squared:  0.8089
F-statistic: 501.5 on 4 and 469 DF,  p-value: < 2.2e-16
```

3.1 (semi-) partial for x

(Semi-) partial correlation indicates the variance accounted for by the certain IV specifically, which is especially intriguing and important in standard MlR.

Use `pcor()` from the package `ppcor` to compute the partial correlation for

IVs.

Use `spcor()` from the package `ppcor` to compute the semi-partial correlation for IVs.

Throw the regressed variables (both IVs & DV) into the functions above, and period.

The results will include estimated value of correlation and its p-value. However, the drawback is that the returned matrix does not include rownames and colnames, making it reading-unfriendly. However, we only care about the (semi-) partial correlation between DV and all IVs, which means only the very one row or column is enough.

Therefore, DV goes first! Only check the first row or column!

```
# partial correlation
df |> select(salary,educ,salbegin,jobtime,prevexp) |>
  ppcor::pcor()
```

```
$estimate
             salary        educ   salbegin     jobtime     prevexp
salary    1.0000000  0.18362578  0.8118323  0.21275206 -0.22085427
educ      0.1836258  1.00000000  0.2347785  0.04216897 -0.30912243
salbegin  0.8118323  0.23477855  1.0000000 -0.21340370  0.33602452
jobtime   0.2127521  0.04216897 -0.2134037  1.00000000  0.07981028
prevexp  -0.2208543 -0.30912243  0.3360245  0.07981028  1.00000000

$p.value
                salary         educ      salbegin      jobtime      prevexp
salary    0.000000e+00 6.105604e-05 1.163309e-111 3.186348e-06 1.296712e-06
educ      6.105604e-05 0.000000e+00  2.550384e-07 3.611647e-01 6.903840e-12
salbegin 1.163309e-111 2.550384e-07  0.000000e+00 2.967847e-06 6.775217e-14
jobtime   3.186348e-06 3.611647e-01  2.967847e-06 0.000000e+00 8.358673e-02
prevexp   1.296712e-06 6.903840e-12  6.775217e-14 8.358673e-02 0.000000e+00
```

```
$statistic
            salary       educ  salbegin    jobtime   prevexp
salary    0.000000  4.0454629 30.110719  4.7153987 -4.904006
educ      4.045463  0.0000000  5.230663  0.9140415 -7.039248
salbegin 30.110719  5.2306629  0.000000 -4.7305296  7.726346
jobtime   4.715399  0.9140415 -4.730530  0.0000000  1.733935
prevexp  -4.904006 -7.0392480  7.726346  1.7339351  0.000000

$n
[1] 474

$gp
[1] 3

$method
[1] "pearson"
```

```
# part or semi-partial correlation
df |> select(salary,educ,salbegin,jobtime,prevexp) |>
  ppcor::spcor()
```

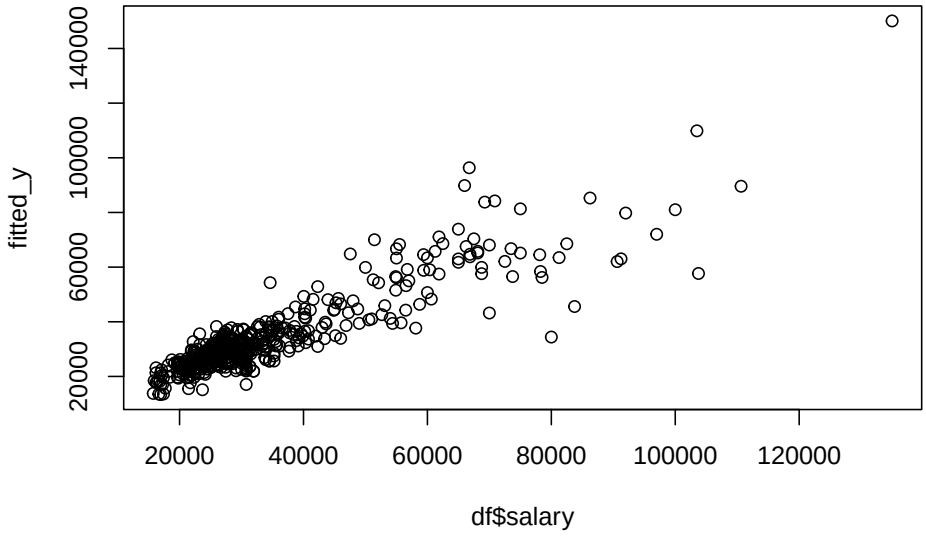
```
$estimate
             salary        educ   salbegin     jobtime     prevexp
salary    1.0000000  0.08132013  0.6052725  0.09478688 -0.09857818
educ      0.1319374  1.00000000  0.1705911  0.02981024 -0.22957564
salbegin  0.6028339  0.10472088  1.0000000 -0.09470792  0.15468589
jobtime   0.2119468  0.04108416 -0.2126269  1.00000000  0.07793657
prevexp  -0.2054291 -0.29487448  0.3236570  0.07263463  1.00000000

$p.value
               salary         educ     salbegin    jobtime      prevexp
salary   0.000000e+00 7.788628e-02 2.067811e-48 0.03975543 3.244186e-02
educ     4.126643e-03 0.000000e+00 1.994428e-04 0.51867959 4.739749e-07
salbegin 6.161045e-48 2.303088e-02 0.000000e+00 0.03992148 7.559174e-04
```

```
jobtime  3.477648e-06 3.736610e-01 3.230041e-06 0.00000000 9.112418e-02
prevexp  6.973073e-06 6.655125e-11 6.011570e-13 0.11543038 0.000000e+00

$statistic
            salary       educ  salbegin    jobtime   prevexp
salary    0.000000  1.7669539 16.466993  2.0620275 -2.145298
educ      2.882488  0.0000000  3.749348  0.6458697 -5.108220
salbegin 16.362654  2.2804165  0.000000 -2.0602942  3.390753
jobtime   4.696710  0.8904871 -4.712493  0.0000000  1.692976
prevexp  -4.545809 -6.6830784  7.407988  1.5771712  0.000000

$n
[1] 474

$gp
[1] 3

$method
[1] "pearson"
```

4 Goodness of Fit

4.1 fitted values & real values

```
fitted_y = fitted(fit) # predicted values
plot(df$salary,fitted_y,'p')
```

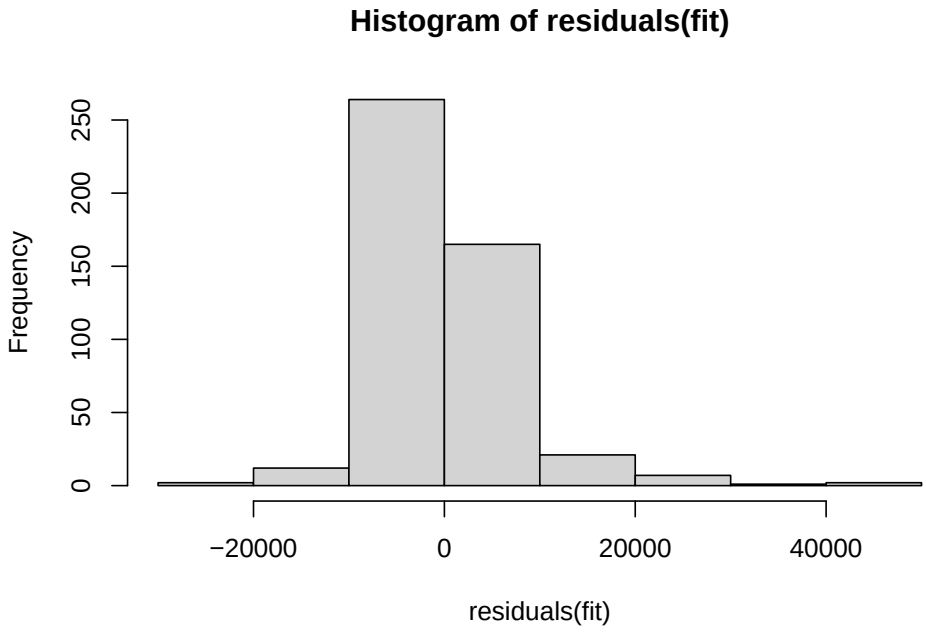


4.2 residual

If the model fits well, then the residual should follow a normal distribution.

Fitted model's residuals can be obtained through `residuals(model)` or `model$residuals`. This works the same for other attributes of a linear model, such as coefficients.

```
hist(residuals(fit))
```




```
# hist(fit$residuals)
```

4.3 plot more

Use `plot(model)` to get multiple plots reflecting the goodness of fit from many perspectives.

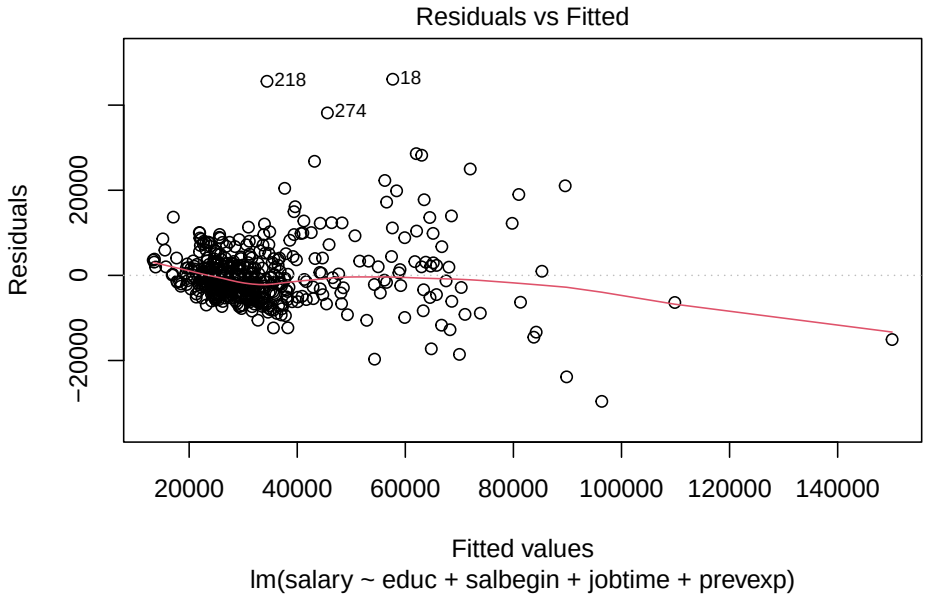
Specify parameter `which = vector_of_range` to designate the plots to show, 6 the most. `which = 1:4` by default.

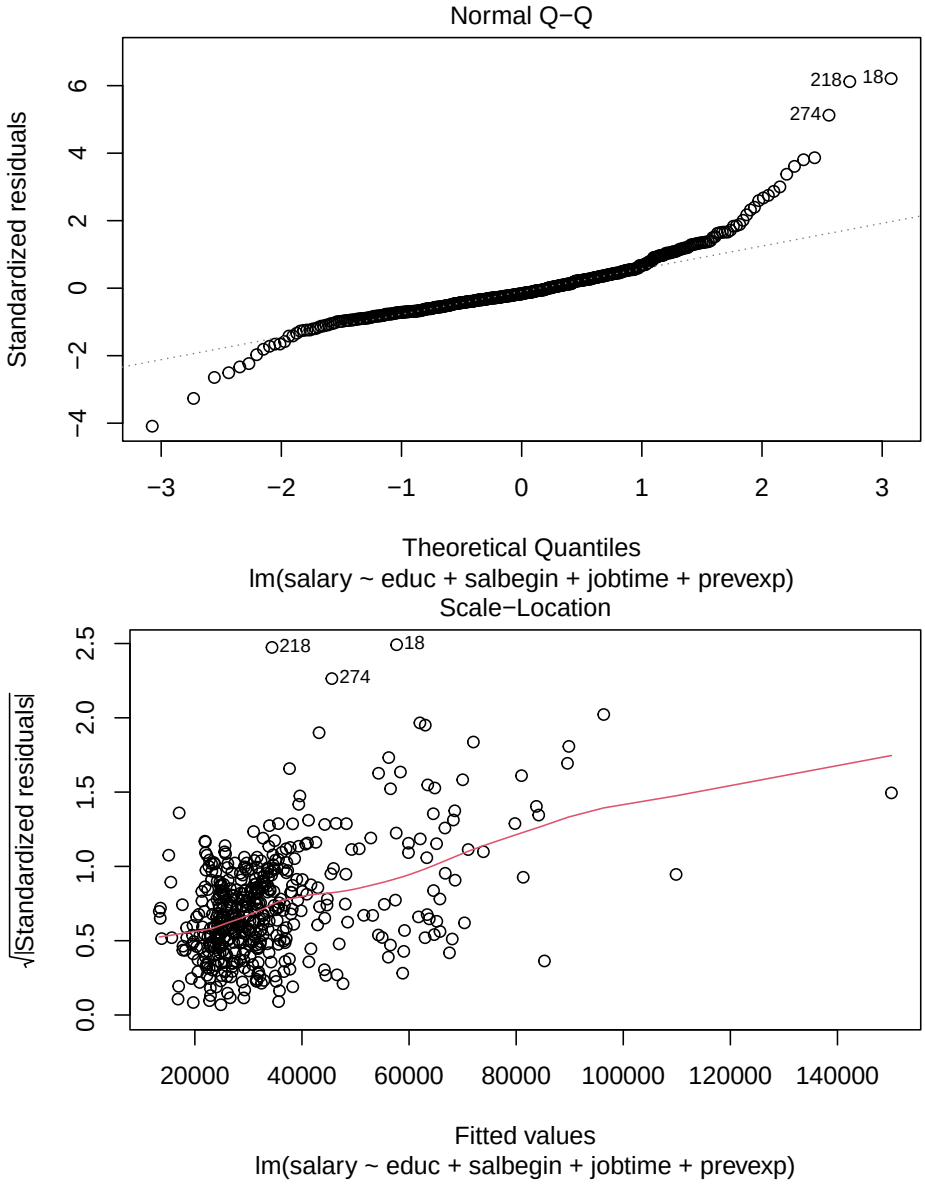
Following the order, the six plots are:

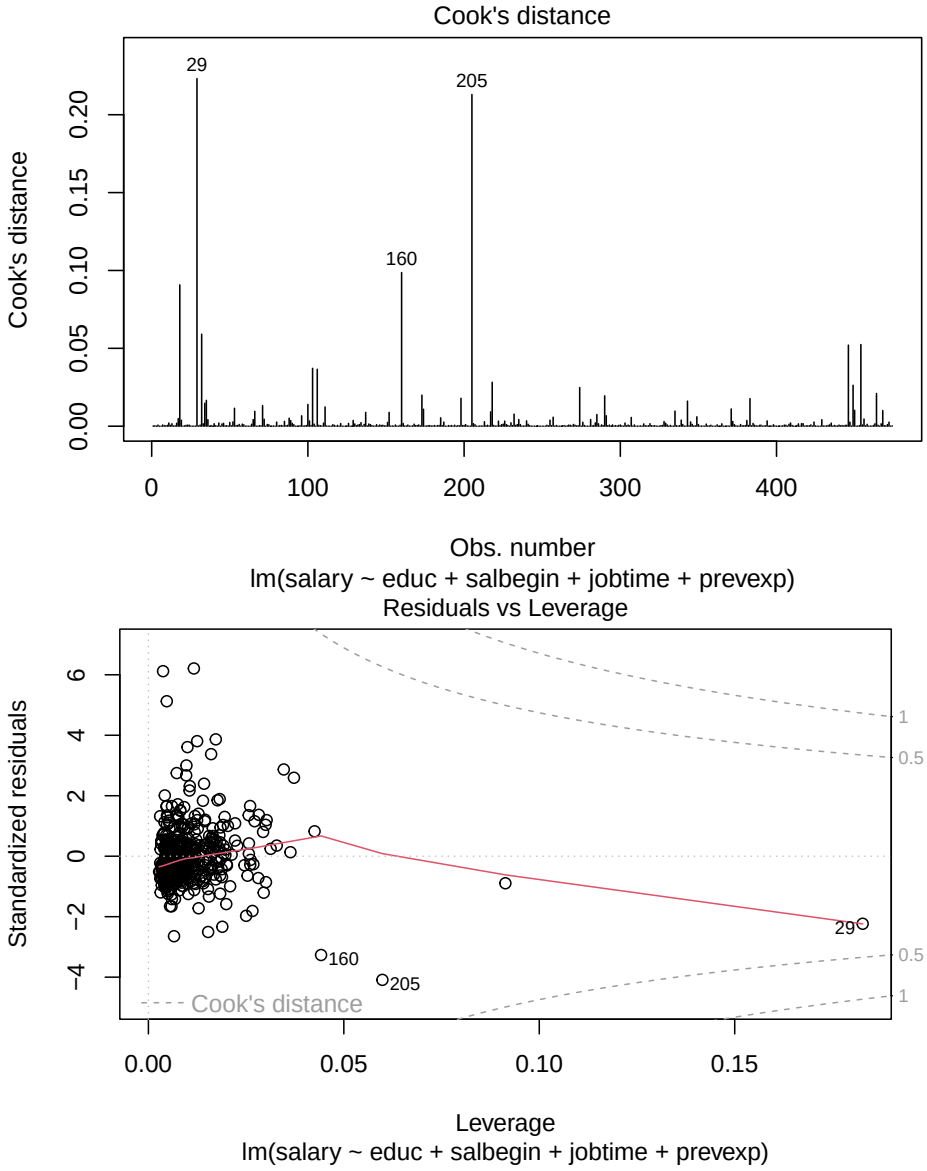
- Residuals & Fitted Values
  - test the homoscedascitiy assumption
- QQ Plot for Residuals
  - test the Gaussian assumption
- Square Root of Standardized Residuals & Fitted Value
  - similar to the first one
- Cook's Distance

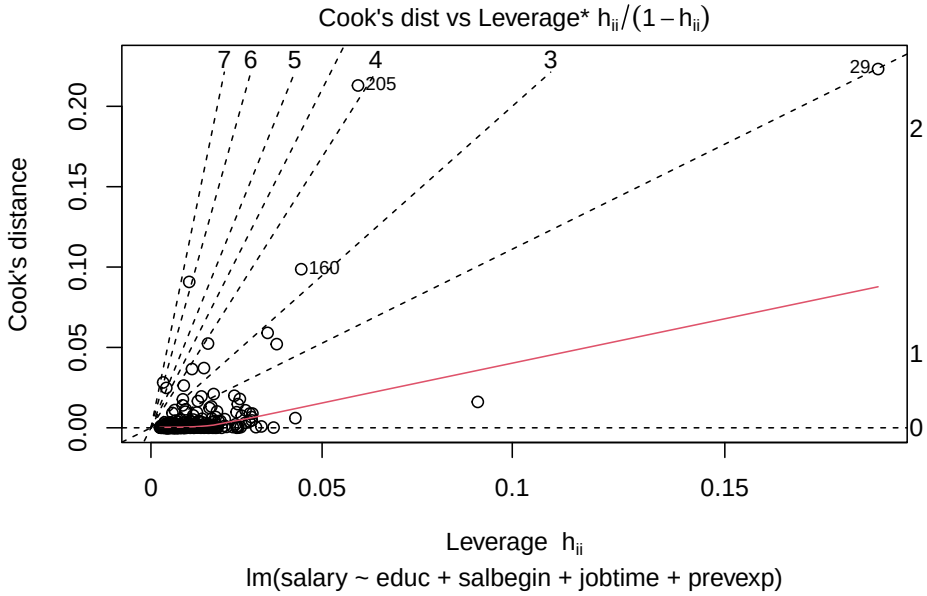
  - check outliers
- Standardized Residuals & Leverage
  - See if influential points have huge residual, if so, possibly outlier!
- Cook's Distance & Leverage
  - Large in both, possibly outlier!

```
plot(fit, which=1:6)
```




Normal Q−Q
Standardized residuals
218
18
274
−4
−2
0
2
4
6
−3
−2
−1
0
1
2
3
Theoretical Quantiles
lm(salary ~ educ + salbegin + jobtime + prevexp)
Scale−Location
√|Standardized residuals|
218
18
274
0.0
0.5
1.0
1.5
2.0
2.5
20000
40000
60000
80000
100000
120000
140000
Fitted values
lm(salary ~ educ + salbegin + jobtime + prevexp)

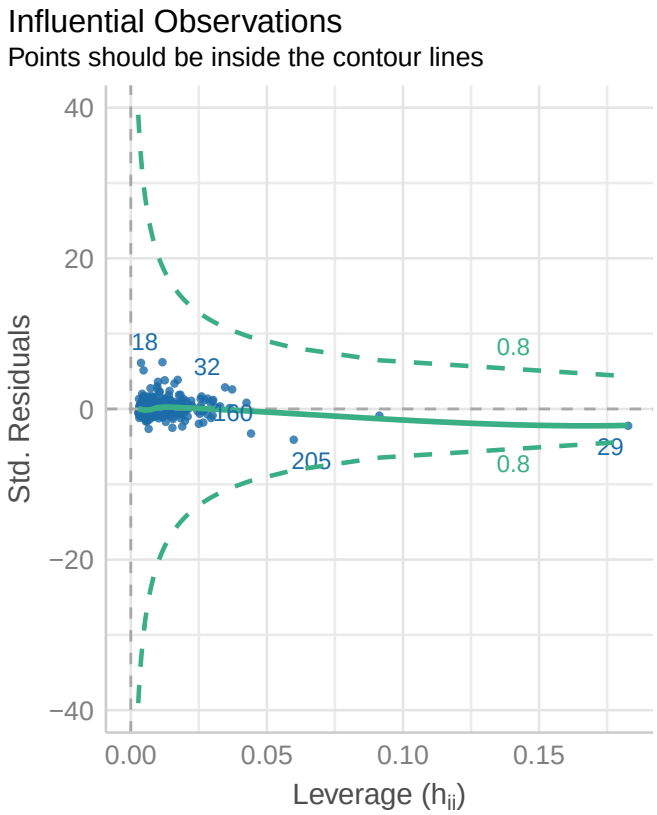

Cook's distance
29
205
160
Cook's distance
0.00
0.05
0.10
0.15
0.20
0
100
200
300
400
Obs. number
lm(salary ~ educ + salbegin + jobtime + prevexp)
Residuals vs Leverage
Standardized residuals
−4
−2
0
2
4
6
1
0.5
0.5
1
29
160
205
Cook's distance
0.00
0.05
0.10
0.15
Leverage
lm(salary ~ educ + salbegin + jobtime + prevexp)



There's another way to do the model checking. Use `performance::check_model(model)`.

The default option is to check all the "vif", "qq", "normality", "linearity", "ncv", "homogeneity", "outliers", "reqq", "pp_check", "binned_residuals" and "overdispersion", which can be specified through `check = specifying_vector`. Here I'll take the check of outliers as an example.

```
performance::check_model(fit,check = 'outliers')
```



Do not forget to `detach()` the attached data.

```
detach(df)
```

5 容易踩的坑

新手容易栽的三件事

1. **多重共线性**（multicollinearity）：解释变量之间相关性太高时 $X'X$ 接近奇异，$\hat{\beta}$ 方差爆炸，单个系数的 t 检验不显著但模型整体 F 检验显著。诊断：`car::vif()`，VIF >10（甚至 >5）就要警惕。常见来源——把同一个东西测了两遍（如「年龄」和「工龄」）、把多分类哑变量没设参照水平、塞进去一堆「派生变量」（如 X、X^2、$X\times Z$ 没中心化）。

2. R^2 **高不等于因果**：$R^2 = 0.9$ 只说明模型解释了 90% 的方差，不代表 $X \to Y$ 是因果关系。教科书最爱举的例子：用「鞋码」预测「数学成绩」，R^2 不低，但因果链是「年龄 → 鞋码 & 数学」，鞋码本身没用。
3. **残差不正态 / 异方差**：OLS 的 t/F 检验和置信区间假设 $\varepsilon \sim \text{Normal}(0, \sigma^2)$。Q-Q 图严重偏离对角线 → 正态性破；Scale-Location 图残差扇形分布 → 异方差。补救：稳健标准误（`sandwich::vcovHC`）、对 Y 做对数变换、或换 GLM。

6 Appendix

6.1 Attach and Detach

We can use the function `attach()` to use variables in the database conveniently. After `attach()`, column variables can be used directly without `$`. (R will do the searching work behind for you!)

However, sometimes names of the variables will conflict with each other and cause problems. Not recommended from my perspective, because you'll not know what database the variable is from, especially when you're trying to check the codes. Combined with tube-friendly and formula-pattern genre, this won't be any more trouble.

```
attach(df)
```

After attaching a database, remember to `detach()` that finally, in case of conflicts with other variables.

6.2 MLR

```
heart = read.csv('heart.csv')
```

6.2.1 Fit the Model

```
mfit = lm(scale(heart.disease)~scale(biking)+scale(smoking),
          data = heart)
```

6.2.2 Summary the Model

```
summary(mfit)
```

```
##
## Call:
## lm(formula = scale(heart.disease) ~ scale(biking) + scale(smoking),
##     data = heart)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.47658 -0.09762  0.00792  0.09671  0.42283
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)    -6.119e-17  6.410e-03    0.00        1
## scale(biking)  -9.403e-01  6.418e-03 -146.53   <2e-16 ***
## scale(smoking)  3.234e-01  6.418e-03   50.39   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1431 on 495 degrees of freedom
## Multiple R-squared:  0.9796, Adjusted R-squared:  0.9795
## F-statistic: 1.19e+04 on 2 and 495 DF,  p-value: < 2.2e-16
```

It's predicted that if `biking` changes for one standard deviation, then `heart.disease` is to change for -0.940 standard deviation, and that if `smoking` changes for one standard deviation, then `heart.disease` is to

change for 0.323 standard deviation. Meanwhile, coefficients of both `biking` and `smoking` are highly significant with $p < .001$.

6.2.3 Goodness of Fit

After qualitative analysis, now move on to quantitative works.

```
result = summary(mfit)
result$adj.r.squared
```

```
[1] 0.9795351
```

Adjusted R^2 of 0.9795 indicates a highly awesome goodness of fit and its ability to explain the variation of the data!

```
cor(x = fitted(mfit), y = heart$heart.disease)
```

```
[1] 0.9897563
```

The correlation between the fitted values and the observed (real) values is as high as 0.9897563, which is exactly the square root of R^2. Jointly speaking, the model has great credibility in prediction.

6.2.4 VIF

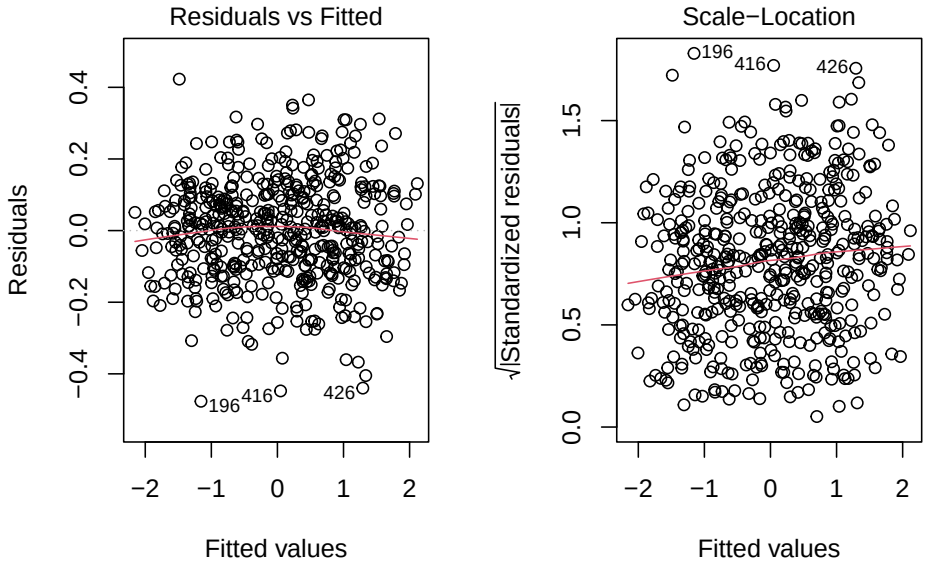
```
car::vif(mfit)
```

```
 scale(biking) scale(smoking)
      1.000229       1.000229
```

VIFs of `biking` and `smoking` are 1.000229 and 1.000229 respectively, far less than 5, showing that there should be no worries about multicollinearity.

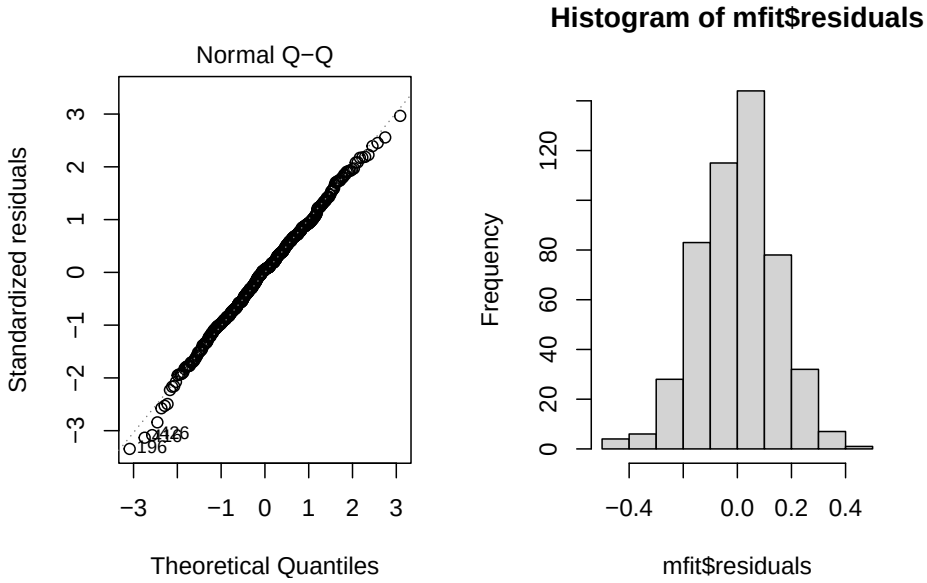
6.2.5 Plots of Residuals

```
par(mfrow = c(1,2))
plot(mfit,which = c(1,3))
```



From the first plot, it's shown that there's no obvious tendency between the fitted values and the residuals.

```
par(mfrow = c(1,2))
plot(mfit,which = 2)
hist(mfit$residuals)
```



In QQ plot, the scatter points of residuals closely center around the line offered by normal distribution. In the histogram of residuals, a clear prototype of normal distribution is depicted. What's more, move on from qualitative analysis to the quantitative.

```
shapiro.test(mfit$residuals)
```

```
    Shapiro-Wilk normality test

data:  mfit$residuals
W = 0.997, p-value = 0.4935
```

```
mfit$residuals |> mean()
```

```
[1] 2.305236e-18
```

From the Shapiro test, $W = 0.9969963, p = 0.4935143$, indicating that the residuals from the fitted model follow a normal distribution. Moreover, the mean is computed to be (extremely close to) zero.

Therefore, we can safely say that the residuals of the fitted model do follow

a normal distribution.

Based on what we've discussed, we can conclude confidently that the model fits the data well!