Test 03 - Regression diagnostics and tidy regression results

Richard

2024-01-01

Load required packages

```
library(tidyverse)
# if you're using macOS, you can run: library(dplyr)
library(skimr)
library(broom)
library(modelr)
```

Prepare Data

```
Hsb = read_csv("data/raw/hsb.csv")
Hsb = Hsb %>%
  mutate(
    race = as.factor(race),
    schtyp = as.factor(schtyp),
    prog = as.factor(prog)
  )
```

A regression

Recall that we run a regerssion between `write` score on `read` score and `female` (equal 1 for female students):

```r
ols_reg_fit = lm(formula = write ~ read + female, data = Hsb)
summary(ols_reg_fit)
```

```
Call:
lm(formula = write ~ read + female, data = Hsb)

Residuals:
    Min      1Q  Median      3Q     Max 
-17.523  -5.658   0.168   5.043  15.175 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept) 20.22837    2.71376   7.454 2.80e-12 ***
read         0.56589    0.04938  11.459  < 2e-16 ***
female       5.48689    1.01426   5.410 1.82e-07 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Tidy the coefficients

- The about regressions results are in text format, which is time-consuming to copy to a report
- How about we transform it into a dataframe to easy to manipulate later: use function `tidy` from `broom` package
- For example, if we want to get the coefficient of `read`, we can easy `filter` and `select` to get the coefficient, rather than copy-and-paste:

```
tidy(ols_reg_fit)
```

```
# A tibble: 3 × 5
  term        estimate std.error statistic  p.value
  <chr>          <dbl>     <dbl>     <dbl>    <dbl>
1 (Intercept)   20.2      2.71        7.45 2.80e-12
```

Get predictions and residuals

- Recall that in a regression

$$Y = a + bX + e$$

- So the prediction is:

$$\hat{Y} = \hat{a} + \hat{b}X$$

- and residuals:

$$\hat{e} = Y - \hat{Y}$$

- We have several ways to get the predictions and residuals

1st way: manual calculation

The fitted Y is the product of estimated coefficients and the corresponding X.

```
write_hat = 20.2283684 + 0.5658869*Hsb$read + 5.4868940*Hsb$female
head(write_hat)
```

```
[1] 52.48392 64.19557 45.12739 55.87924 46.82505 45.12739
```

The residuals is the difference between Y and fitted Y:

```
head(Hsb$write - write_hat)
```

```
[1]  -0.4839217  -5.1955716 -12.1273920 -11.8792431   5.1749473   6.8726080
```

2nd way: use `tidy::augment`

This function added several new columns, including the fitted and residuals to the original data. Compare the results to the manual calculation above.

```r
Hsb = augment(ols_reg_fit, Hsb)
Hsb %>%
  select(.fitted:.std.resid) %>%
  head()
```

```
# A tibble: 6 × 6
  .fitted  .resid   .hat .sigma   .cooksd .std.resid
    <dbl>   <dbl>  <dbl>  <dbl>     <dbl>      <dbl>
1    52.5  -0.484 0.0118   7.15 0.0000186    -0.0683
2    64.2  -5.20  0.0219   7.14 0.00404      -0.737
3    45.1 -12.1   0.0147   7.10 0.0146       -1.71
4    55.9 -11.9   0.0160   7.10 0.0152       -1.68
5    46.8   5.17  0.0126   7.14 0.00227       0.730
6    45.1   6.87  0.0147   7.13 0.00469       0.971
```

Training vs Test sample

- We often split our data into training vs test sample:
  - Training sample: to train historical data
  - Test sample: new data to make prediction
- e.g., Netflix uses our historical watched movies to recommend our next movies to watch

Re-train our case

```
# train: use the first 150 obs
ols_reg_fit = lm(formula = write ~ read + female, data = Hsb[1:150,
# test: use the last 50 obs
augment(ols_reg_fit, newdata = Hsb[151:200,])
```

```
# A tibble: 50 × 17
      id female race    ses schtyp prog   read write  math science socst
.fitted
   <dbl>  <dbl> <fct> <dbl> <fct>  <fct> <dbl> <dbl> <dbl>   <dbl> <dbl>
<dbl>
 1    43      1 3         1 1      2        47    37    43      42    46
51.1
 2    96      1 4         3 1      2        65    54    61      58    56
61.7
 3   138      1 4         2 1      3        43    57    40      50    51
48.8
 4    10      1 1         2 1      1        47    54    49      53    61
51.1
 5    71      1 4         2 1      1        57    62    56      58    66
57.0
```

Regression diagnostics

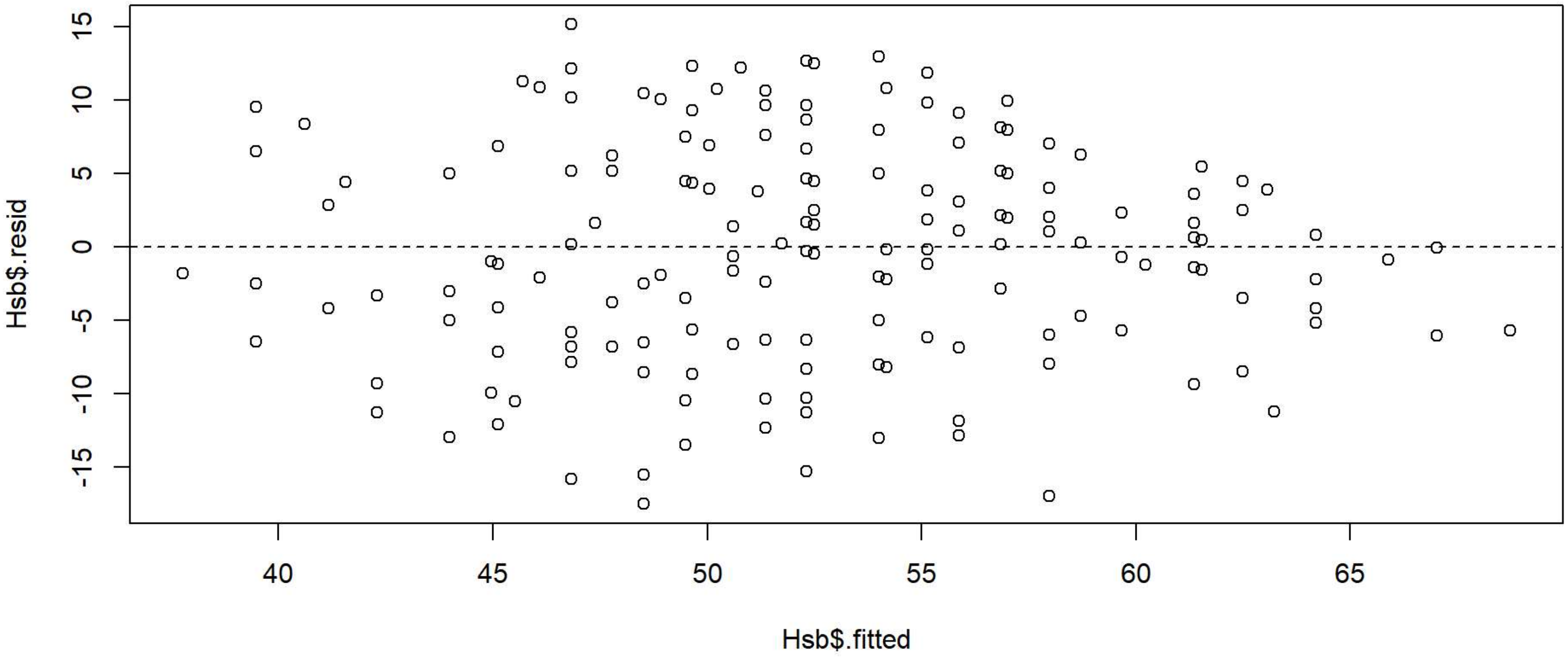
- The OLS regressions have several important assumptions, which we assume the data must be, to make sure the estimation is correct.
- Thus, after running regression, we often need to check these assumptions again to make sure
- This process is called as "regression diagnostics"
- I borrow a lot from this slide note from UCLA

Assumption 1: Homogeneity of variance (homoscedasticity)

- It assumes that the variance of residuals is constant
- If the model is well-fitted, there should be no pattern to the residuals plotted against the fitted values.
- Let’s plot to see:

Plot of residuals

```
plot(Hsb$.resid ~ Hsb$.fitted)
abline(h = 0, lty = 2)
```

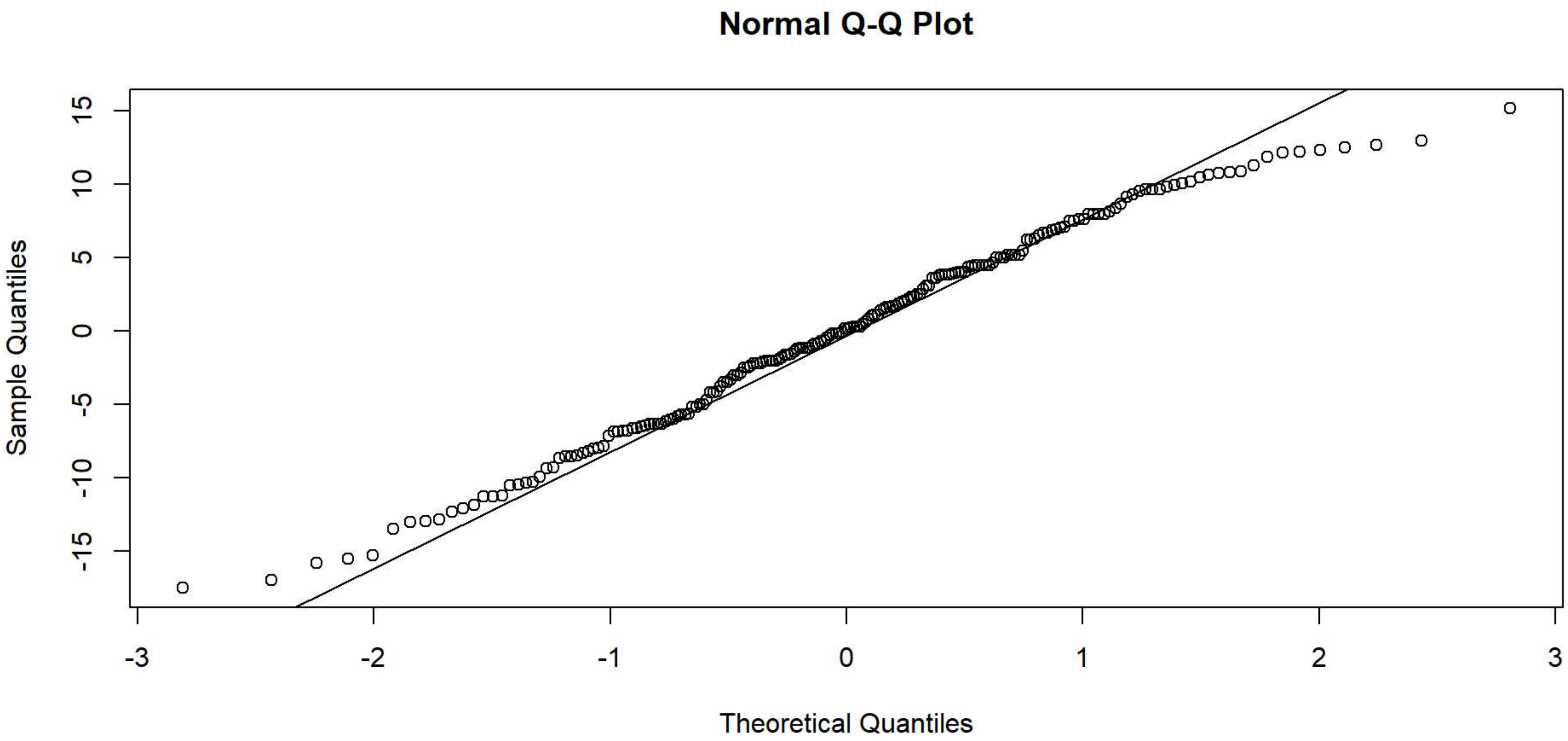


Assumption 2: Normality of residuals

- It assumes that the residuals follow a normal distribution
- Thus, we need to test normality for the residuals

Q-Q plot

```
qqnorm(Hsb$.resid)
qqline(Hsb$.resid)
```



Normality test for residuals

Do you remember we have a test for normality?

Assumption 3: Check for multicolinearity

- The term collinearity implies that two variables are near perfect linear combinations of one another.
- VIF, variance inflation factor, is used to measure the degree of multicollinearity.
- Rule-of-thump: VIF >= 10 means that the variable could be considered as a linear combination of other independent variables.

Multicolinearity check in R

- Install `car` package if not yet

```r
# install.packages("car")
car::vif(ols_reg_fit)
```

```
    read   female 
1.001121 1.001121 
```

- All coefficients have low VIF
  - Less concern on multicolinearity problem

Quiz time

Hmm…