Lab 13

Yunting Chiu

2021-04-12

Fitting a polynomial regression

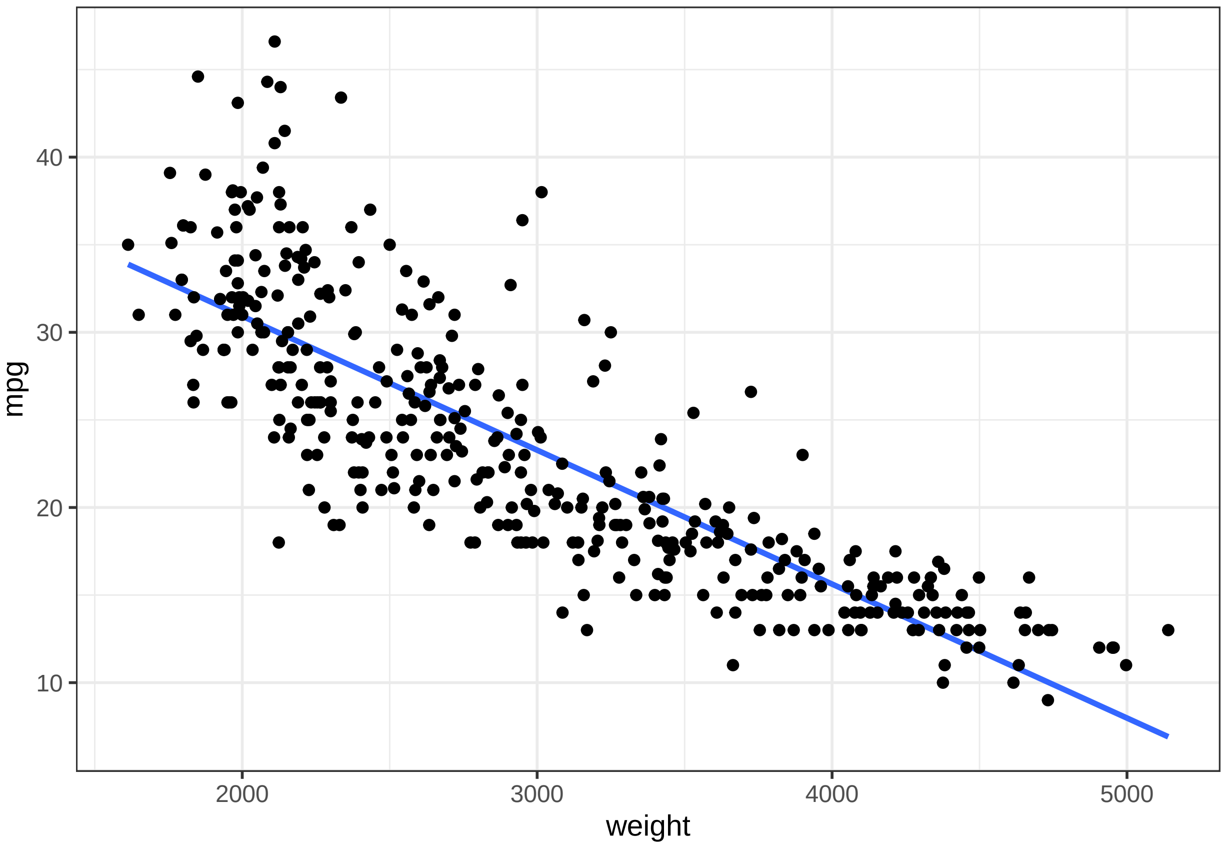
Because the distribution of observed data does not match the linear regression line at the beginning. We decide not to use the linear regression model.

```r
library(ISLR)
attach(Auto)
```

```
## The following object is masked from package:ggplot2:
##
##     mpg
```

```r
reg <- lm(mpg~weight, data = Auto)

Auto %>%
  ggplot(aes(weight, mpg))+
  geom_smooth(method = lm, se = FALSE) +
  geom_point() +
  theme_bw()
```

```
## `geom_smooth()` using formula 'y ~ x'
```

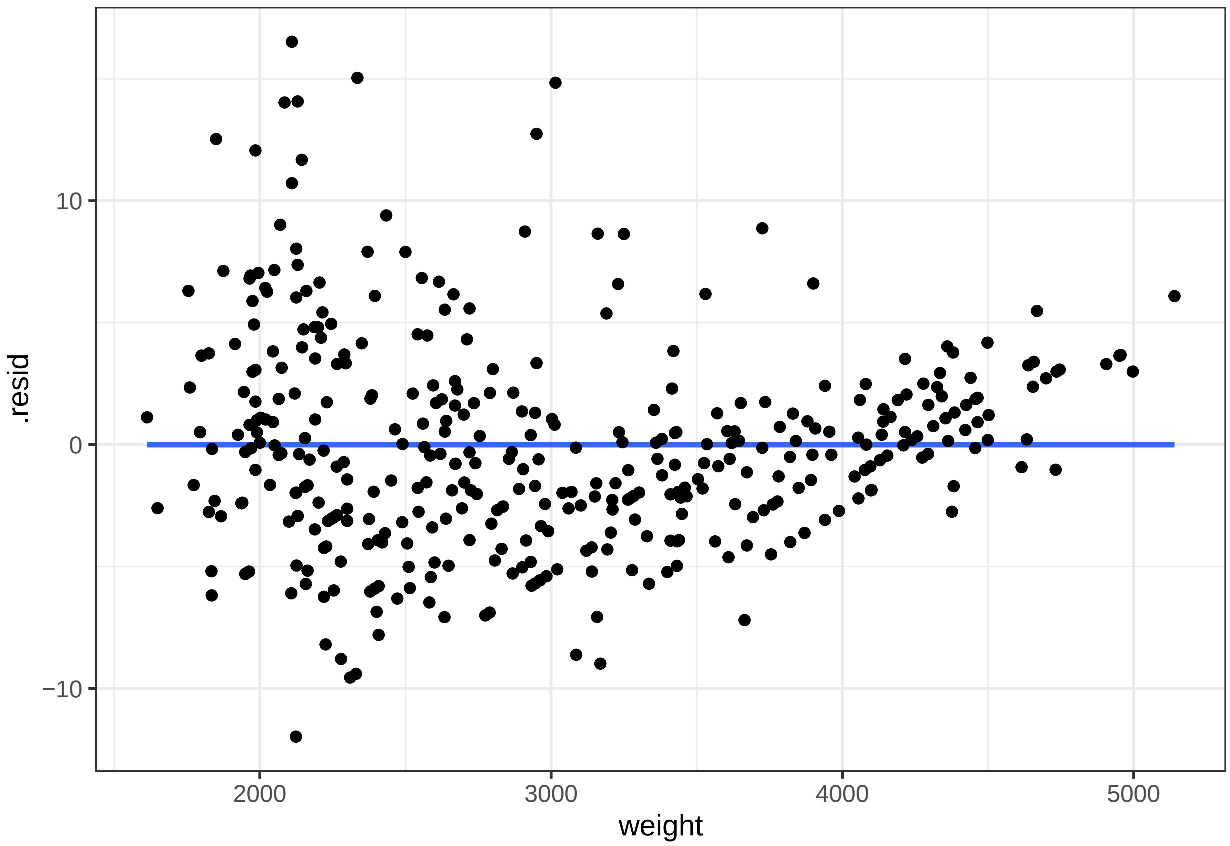


look at the residual plot

Examine the pattern in the residuals vs weight plot, which indicates a bad bit (but not really bad). To achieve this goal, consider using a more reliable regression model.

```
# plot( weight, residuals(reg))
augment(reg) %>%
  ggplot(aes(x = weight, y = .resid)) +
  geom_smooth(method = lm, se = FALSE) +
  geom_point() +
  theme_bw()
```

```
## `geom_smooth()` using formula 'y ~ x'
```



summary Table

```r
summary(reg)
```

```
##
## Call:
## lm(formula = mpg ~ weight, data = Auto)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -11.9736  -2.7556  -0.3358   2.1379  16.5194
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 46.216524   0.798673   57.87   <2e-16 ***
## weight      -0.007647   0.000258  -29.64   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.333 on 390 degrees of freedom
## Multiple R-squared:  0.6926, Adjusted R-squared:  0.6918
## F-statistic: 878.8 on 1 and 390 DF,  p-value: < 2.2e-16
```

Try to use polynomial regression model

model selection

```
polynomial.fit <-  regsubsets(mpg ~ poly(weight, 10), data=Auto )
summary(polynomial.fit)
```

```
## Subset selection object
## Call: regsubsets.formula(mpg ~ poly(weight, 10), data = Auto)
## 10 Variables  (and intercept)
##                      Forced in Forced out
## poly(weight, 10)1       FALSE      FALSE
## poly(weight, 10)2       FALSE      FALSE
## poly(weight, 10)3       FALSE      FALSE
## poly(weight, 10)4       FALSE      FALSE
## poly(weight, 10)5       FALSE      FALSE
## poly(weight, 10)6       FALSE      FALSE
## poly(weight, 10)7       FALSE      FALSE
## poly(weight, 10)8       FALSE      FALSE
## poly(weight, 10)9       FALSE      FALSE
## poly(weight, 10)10      FALSE      FALSE
## 1 subsets of each size up to 8
## Selection Algorithm: exhaustive
##          poly(weight, 10)1 poly(weight, 10)2 poly(weight, 10)3
## 1  ( 1 ) "*"               " "               " "
## 2  ( 1 ) "*"               "*"               " "
## 3  ( 1 ) "*"               "*"               " "
## 4  ( 1 ) "*"               "*"               " "
## 5  ( 1 ) "*"               "*"               " "
## 6  ( 1 ) "*"               "*"               " "
## 7  ( 1 ) "*"               "*"               " "
## 8  ( 1 ) "*"               "*"               " "
##          poly(weight, 10)4 poly(weight, 10)5 poly(weight, 10)6
## 1  ( 1 ) " "               " "               " "
## 2  ( 1 ) " "               " "               " "
## 3  ( 1 ) " "               " "               " "
## 4  ( 1 ) " "               " "               " "
## 5  ( 1 ) " "               "*"               " "
## 6  ( 1 ) " "               "*"               "*"
## 7  ( 1 ) " "               "*"               "*"
## 8  ( 1 ) "*"               "*"               "*"
##          poly(weight, 10)7 poly(weight, 10)8 poly(weight, 10)9
## 1  ( 1 ) " "               " "               " "
## 2  ( 1 ) " "               " "               " "
## 3  ( 1 ) " "               " "               " "
## 4  ( 1 ) "*"               " "               " "
## 5  ( 1 ) "*"               " "               " "
## 6  ( 1 ) "*"               " "               " "
## 7  ( 1 ) "*"               " "               "*"
## 8  ( 1 ) "*"               " "               "*"
##          poly(weight, 10)10
## 1  ( 1 ) " "
## 2  ( 1 ) " "
## 3  ( 1 ) "*"
```

```
## 4  ( 1 ) "*"
## 5  ( 1 ) "*"
## 6  ( 1 ) "*"
## 7  ( 1 ) "*"
## 8  ( 1 ) "*"
```

find out the largest adjusted R squares

```
summary(polynomial.fit)$adjr2
```

```
## [1] 0.6918423 0.7136830 0.7146411 0.7151404 0.7150263 0.7147608 0.7144345
## [8] 0.7140268
```

```
which.max(summary(polynomial.fit)$adjr2)
```

```
## [1] 4
```

So the largest adjusted R squares is

$$\hat{y} = \beta_0 + \beta_1 X + \beta_2 X^2 + \beta_3 X^3 + \beta_4 X^4 + e$$

find out the smallest CP

```
summary(polynomial.fit)$cp
```

```
## [1] 30.079827  1.452349  1.163224  1.498208  2.660492  4.024793  5.468433
## [8]  7.018200
```

```
which.min(summary(polynomial.fit)$cp)
```

```
## [1] 3
```

So the smallest CP is

$$\hat{y} = \beta_0 + \beta_1 X + \beta_2 X^2 + \beta_3 X^3 + e$$

find out the smallest BIC (penalized-likelihood criteria)

```
summary(polynomial.fit)$bic
```

```
## [1] -450.5016 -474.3535 -470.7051 -466.4321 -461.3181 -455.9986 -450.5986
## [8] -445.0903
```

```
which.min(summary(polynomial.fit)$bic)
```

```
## [1] 2
```

so the smallest BIC is

$$\hat{y} = \beta_0 + \beta_1 X + \beta_2 X^2 + e$$

Cross-validation agrees with the quadratic model

Assessing how the results of a statistical analysis will generalize to an independent data set.

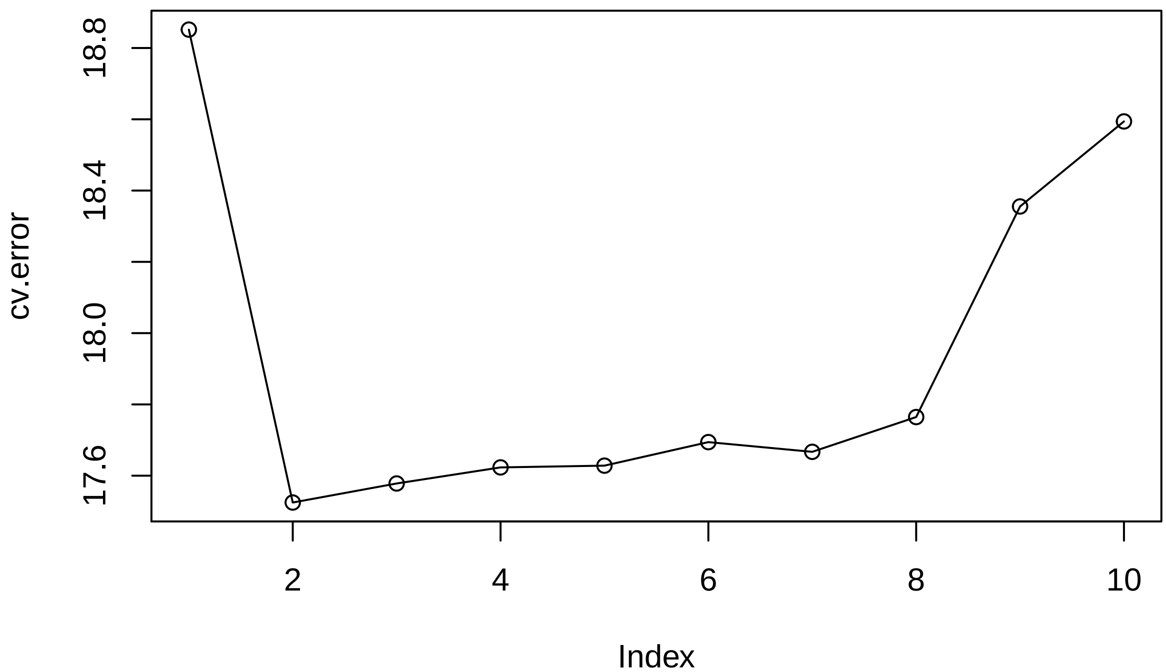
```
library(boot)
cv.error = rep(0,10)
for (p in 1:10){
  polynomial = glm( mpg ~ poly(weight, p) )
  cv.error[p] = cv.glm( Auto, polynomial )$delta[1]}

cv.error
```

```
##  [1] 18.85161 17.52474 17.57811 17.62324 17.62822 17.69418 17.66695 17.76456
##  [9] 18.35543 18.59401
```

Plot the cv.error

- The y index represents the **Mean squared error**. Our goal is to find the polynomial model with the lowest MSE. The quadratic model has the lowest MSE, according to the results.
- So, we choose the quadratic regression – degree 2 polynomial. Its prediction MSE is 17.52474.

```r
plot(cv.error); lines(cv.error)
```




```r
which.min(cv.error)
```

```
## [1] 2
```

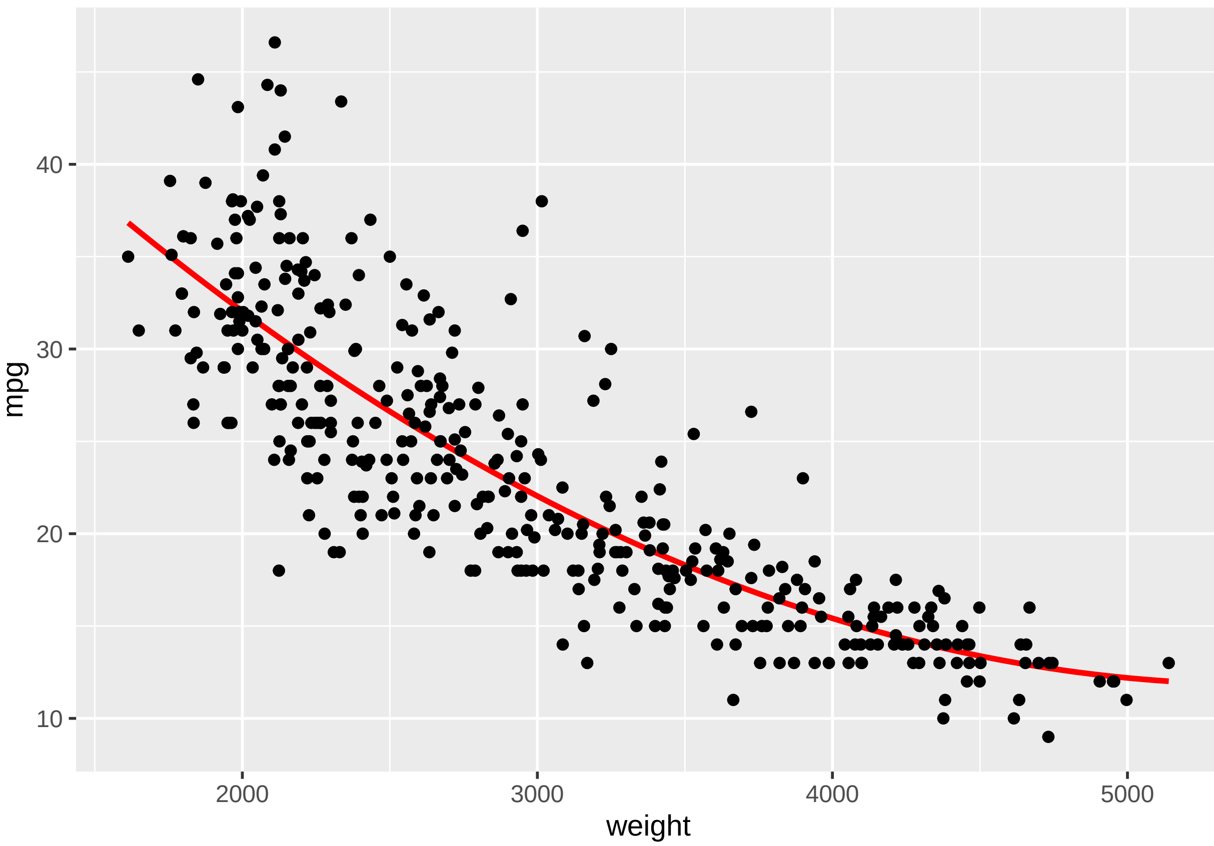
```r
poly2 <- lm( mpg ~ poly(weight,2), data = Auto)
summary(poly2)
```

```
##
## Call:
## lm(formula = mpg ~ poly(weight, 2), data = Auto)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -12.6246  -2.7134  -0.3485   1.8267  16.0866
##
## Coefficients:
##                    Estimate Std. Error t value Pr(>|t|)
## (Intercept)         23.4459     0.2109 111.151  < 2e-16 ***
## poly(weight, 2)1 -128.4436     4.1763 -30.755  < 2e-16 ***
## poly(weight, 2)2   23.1589     4.1763   5.545 5.43e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
```

```
## Residual standard error: 4.176 on 389 degrees of freedom
## Multiple R-squared:  0.7151, Adjusted R-squared:  0.7137
## F-statistic: 488.3 on 2 and 389 DF,  p-value: < 2.2e-16
```

Plot

```
Auto %>%
  ggplot(aes(x = weight, y = mpg)) +
  stat_smooth(method="lm", se=TRUE, fill=NA,
              formula= y ~ poly(x, 2, raw = TRUE), colour="red") +
  geom_point()
```



The quadratic regression best fits the data in this lab. We can conclude that this model is good because it has a higher Adjusted R-squared (0.7137) and each independent variable is significant.

Reference

- https://stats.stackexchange.com/questions/95939/how-to-interpret-coefficients-from-a-polynomial-model-fit