GLM HW2 Jieqi

Jieqi Tu

2/27/2021

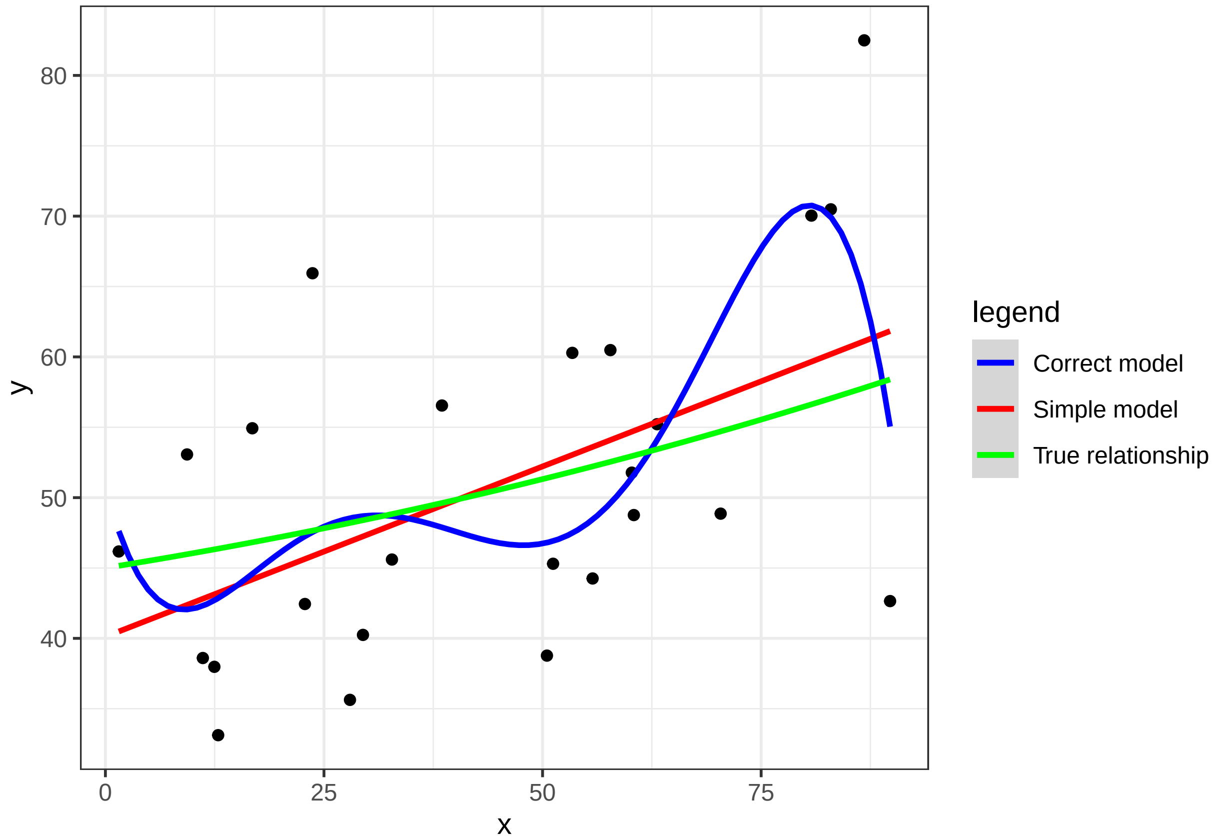
4.36

```r
set.seed(3)
n <- 25
x <- runif(n, 0, 100)
sigma <- 10
error <- rnorm(n, 0, sd = sigma)
y <- 45 + 0.1*x + 5e-4*x^2 + 5e-7*x^3 + 5e-10*x^4 + 5e-13*x^5 + error
true_y <- y - error
## simple model
simple_model <- lm(y~x)
## correct model
correct_model <- lm(y~poly(x, 5, raw = T))
## visualization
xy <- data.frame(x,y,true_y=true_y)

xy %>% ggplot() +
  aes(x = x, y = y) +
  geom_point() +
  geom_smooth(method = "lm", formula = y~x, se = F, aes(color="Simple model")) +
  geom_smooth(method = "lm", formula = y~poly(x, 5, raw = T), se = F, aes(color="Correct model")) +
  geom_smooth(aes(x = x, y = true_y, se = F, color="True relationship"), size = 1) +
  scale_colour_manual(name="legend", values=c("blue", "red", "green")) + theme_bw()
```

```
## Warning: Ignoring unknown aesthetics: se
```

```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```




```r
## quality of fit
mean(abs(simple_model$fitted.values - y))
```

```
## [1] 8.801169
```

```r
mean(abs(correct_model$fitted.values - y))
```

```
## [1] 7.385279
```

Correct model achieved a better fit but has a problem of overfitting, since there are only 25 samples but it used 6 parameters to describe the relationship, where the true coefficients of high-order terms are almost zero. So this resulted in high complexity of the model Complex model has a better fit but performs bad in prediction. Simple model may has higher bias, but does better in prediction. Therefore, model parsimony sometimes provide similar prediction performance but much more interpretability.

4.37

```r
summary(simple_model)
```

```
##
## Call:
## lm(formula = y ~ x)
```

```
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -19.186  -7.205  -2.909   7.247  21.373
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  40.1169     4.1806   9.596 1.66e-09 ***
## x             0.2420     0.0815   2.969  0.00687 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 10.68 on 23 degrees of freedom
## Multiple R-squared:  0.277,  Adjusted R-squared:  0.2456
## F-statistic: 8.814 on 1 and 23 DF,  p-value: 0.006875
```

```r
summary(correct_model)
```

```
##
## Call:
## lm(formula = y ~ poly(x, 5, raw = T))
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -13.659  -4.808  -1.615   8.657  18.395
##
## Coefficients:
##                        Estimate Std. Error t value Pr(>|t|)
## (Intercept)           5.078e+01  1.333e+01   3.811  0.00118 **
## poly(x, 5, raw = T)1 -2.366e+00  2.677e+00  -0.884  0.38775
## poly(x, 5, raw = T)2  2.089e-01  1.785e-01   1.170  0.25632
## poly(x, 5, raw = T)3 -6.792e-03  4.935e-03  -1.376  0.18476
## poly(x, 5, raw = T)4  9.252e-05  5.978e-05   1.548  0.13817
## poly(x, 5, raw = T)5 -4.395e-07  2.629e-07  -1.672  0.11099
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 10.5 on 19 degrees of freedom
## Multiple R-squared:  0.4231, Adjusted R-squared:  0.2713
## F-statistic: 2.787 on 5 and 19 DF,  p-value: 0.04726
```

From the results, we could see that the estimated coefficients for x^2 and higher orders were not significant, and the standard error of these estimated coefficients were large. This inflated variation was caused by collinearity.

```r
x2 = x^2
x3 = x^3
x4 = x^4
x5 = x^5
data = data.frame(x, x2, x3, x4, x5)
cor(data)
```

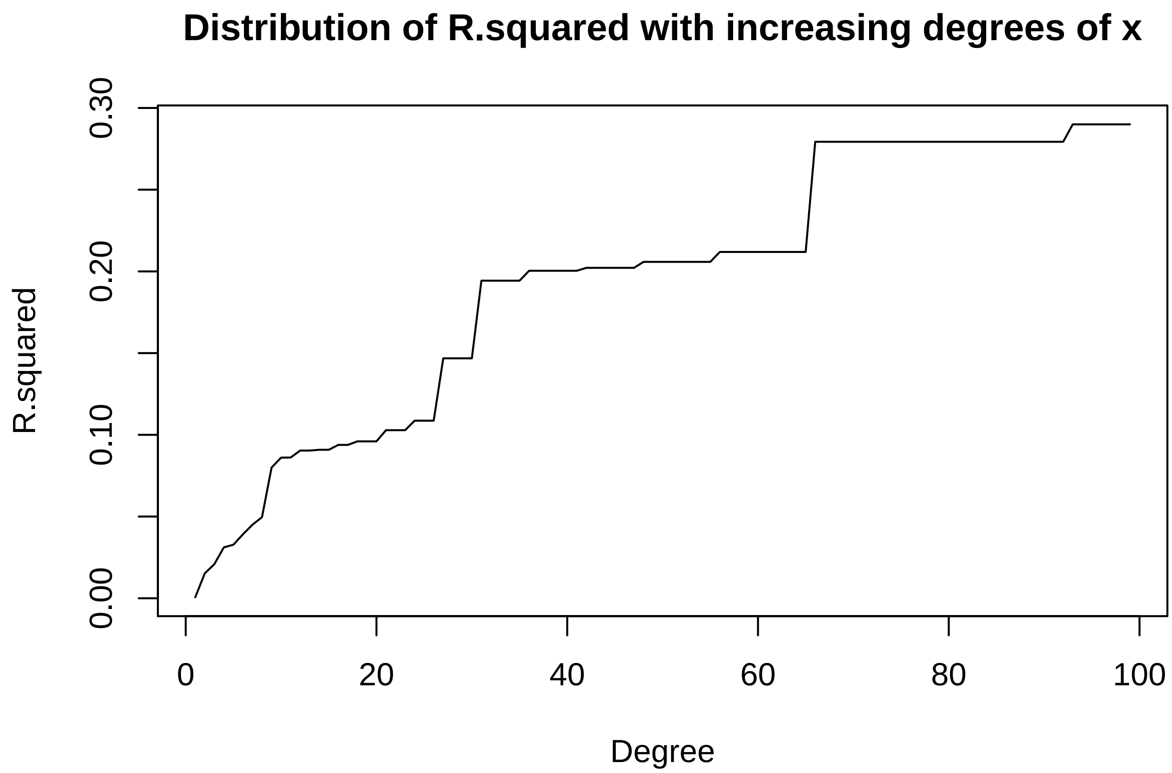
```
##             x        x2        x3        x4        x5
```

```
## x  1.0000000 0.9683736 0.9119344 0.8560397 0.8084994
## x2 0.9683736 1.0000000 0.9841512 0.9533765 0.9212481
## x3 0.9119344 0.9841512 1.0000000 0.9915121 0.9745378
## x4 0.8560397 0.9533765 0.9915121 1.0000000 0.9953190
## x5 0.8084994 0.9212481 0.9745378 0.9953190 1.0000000
```

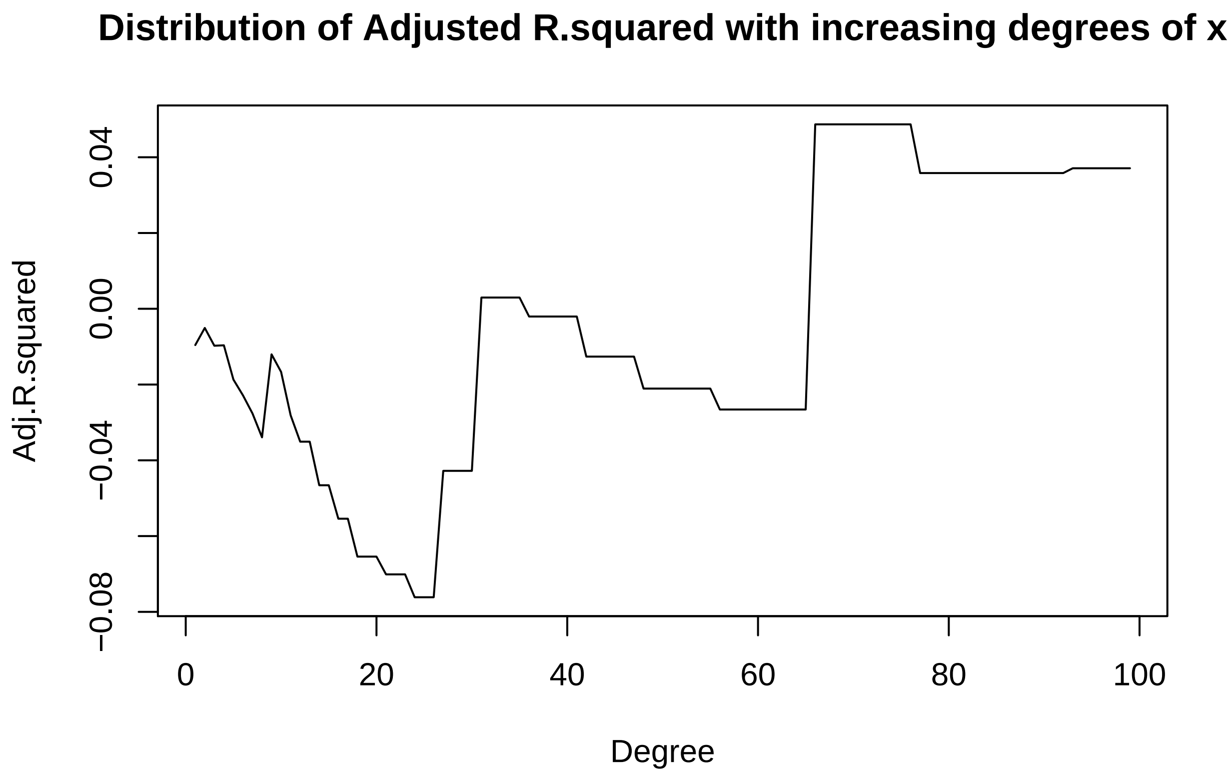
Then we looked into the correlation matrix of predictors in the "correct" model, we could see that the correlation between variables are very high (they are all greater than 0.8, and most of them are greater than 0.9). This can cause collinearity.

4.38

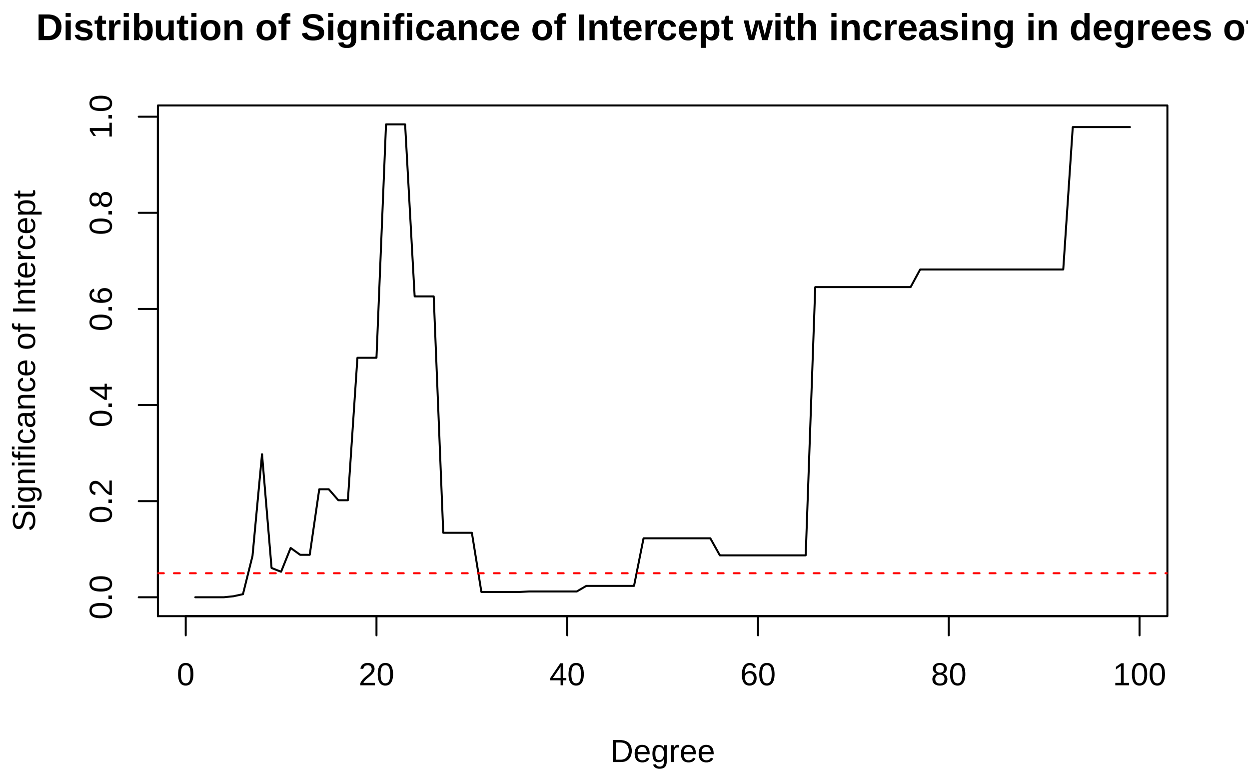
```r
set.seed(1029)
x <- runif(100, 0, 100)
y <- runif(100, 0, 100)
# calculate true R^2
r <- matrix(0, 99, 3)
r[1, 1] <- summary(lm(y ~ x))$'r.squared'
r[1, 2] <- summary(lm(y ~ x))$'adj.r.squared'
r[1, 3] <- summary(lm(y ~ x))$coefficients[1, 4]
for (i in 2:99) {
r[i, 1] <- summary(lm(y ~ poly(x, i, raw = T)))$'r.squared'
r[i, 2] <- summary(lm(y ~ poly(x, i, raw = T)))$'adj.r.squared'
r[i, 3] <- summary(lm(y ~ poly(x, i, raw = T)))$coefficients[1, 4]
}
colnames(r) <- c('r.squared', 'adj.r.squared', 'intercept sig')
plot(r[,1], type = 'l', xlab = 'Degree', ylab = 'R.squared', main = 'Distribution of R.squared with inc
```




```r
plot(r[,2], type = 'l', xlab = 'Degree', ylab = 'Adj.R.squared', main = 'Distribution of Adjusted R.squa
```




```
plot(r[,3], type = 'l', xlab = 'Degree', ylab = 'Significance of Intercept', main = 'Distribution of Si
abline(h = 0.05, col = 'red', lty = 2)
```



From the plot, we could see that, the R^2 increases as the degrees of x increases. However, although the theoretical value of R^2 should achieve 1 when the degree of x is 99, our result only have the value of R^2 less than 0.30 when the degree is 99. This is due to the multicollinearity when p reaches approximately 15. The adjusted R^2 and the p-values for intercept term have more unstable performance. Only a few models have significant intercepts.