A fact about linear regression with a factorial predictor

Xulong Wang

October 19, 2015

A factorial predictor (x) and responses (y) by simulation. The predictor has 3 levels (A, B, C), each level has 3 replicates. "A" was the reference level.

```
set.seed(1)

(x <- rep(c("A", "B", "C"), each = 3) %>% as.factor)
```

```
## [1] A A A B B B C C C
## Levels: A B C
```

```
(y <- as.numeric(x) * 3)
```

```
## [1] 3 3 3 6 6 6 9 9 9
```

```
(y <- y + rnorm(length(y), 0, 1))
```

```
## [1] 2.373546 3.183643 2.164371 7.595281 6.329508 5.179532 9.487429 9.738325
## [9] 9.575781
```

```
(fit = lm(y ~ x) %>% summary)
```

```
##
## Call:
## lm(formula = y ~ x)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -1.1886 -0.2003 -0.0386  0.1378  1.2272
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   2.5739     0.4430   5.810 0.001141 **
## xB            3.7943     0.6265   6.057 0.000918 ***
## xC            7.0267     0.6265  11.216    3e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.7673 on 6 degrees of freedom
## Multiple R-squared:  0.9546, Adjusted R-squared:  0.9394
## F-statistic: 63.04 on 2 and 6 DF,  p-value: 9.376e-05
```

Another linear regression by using only two levels of the predictor.

```r
(x2 <- rep(c("A", "B"), each = 3) %>% as.factor)
```

```
## [1] A A A B B B
## Levels: A B
```

```r
(y2 <- y[1:6])
```

```
## [1] 2.373546 3.183643 2.164371 7.595281 6.329508 5.179532
```

```r
(fit2 = lm(y2 ~ x2) %>% summary)
```

```
##
## Call:
## lm(formula = y2 ~ x2)
##
## Residuals:
##       1       2       3       4       5       6
## -0.2003  0.6098 -0.4095  1.2272 -0.0386 -1.1886
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   2.5739     0.5400   4.766  0.00887 **
## x2B           3.7943     0.7637   4.968  0.00766 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.9354 on 4 degrees of freedom
## Multiple R-squared:  0.8605, Adjusted R-squared:  0.8257
## F-statistic: 24.68 on 1 and 4 DF,  p-value: 0.007662
```

Std. Error for "xB" in the two models are different, which leads to different t and p values. Parameter estimates are the same.

```r
fit$coefficients
```

```
##             Estimate Std. Error   t value     Pr(>|t|)
## (Intercept) 2.573854  0.4429817  5.810293 1.140871e-03
## xB          3.794253  0.6264708  6.056552 9.181545e-04
## xC          7.026658  0.6264708 11.216258 2.999648e-05
```

```r
fit2$coefficients
```

```
##             Estimate Std. Error  t value    Pr(>|t|)
## (Intercept) 2.573854  0.5400462 4.765988 0.008866135
## x2B         3.794253  0.7637407 4.967986 0.007662451
```

Parameter estimates were computed with LSE (least squared estimation):

$$\hat{\beta} = (X^T X)^{-1} X^T y$$

```r
(X <- model.matrix( ~ x))
```

```
##   (Intercept) xB xC
## 1           1  0  0
## 2           1  0  0
## 3           1  0  0
## 4           1  1  0
## 5           1  1  0
## 6           1  1  0
## 7           1  0  1
## 8           1  0  1
## 9           1  0  1
## attr(,"assign")
## [1] 0 1 1
## attr(,"contrasts")
## attr(,"contrasts")$x
## [1] "contr.treatment"
```

```r
(X2 <- model.matrix( ~ x2))
```

```
##   (Intercept) x2B
## 1           1   0
## 2           1   0
## 3           1   0
## 4           1   1
## 5           1   1
## 6           1   1
## attr(,"assign")
## [1] 0 1
## attr(,"contrasts")
## attr(,"contrasts")$x2
## [1] "contr.treatment"
```

```r
(beta = solve(crossprod(X)) %*% crossprod(X, y))
```

```
##                 [,1]
## (Intercept) 2.573854
## xB          3.794253
## xC          7.026658
```

```r
(beta2 = solve(crossprod(X2)) %*% crossprod(X2, y2))
```

```
##                 [,1]
## (Intercept) 2.573854
## x2B         3.794253
```

Given $var(Ay) = Avar(y)A^T$, by plugging in the LSE equation for $\hat{\beta}$, standard errors were computed as follow:

$$var(\hat{\beta}) = var((X^TX)^{-1}X^Ty) = var(y)(X^TX)^{-1}$$

Given var(y) is a diagnol,

$$var(y)(X^TX)^{-1} = var(y)diag((X^TX)^{-1})$$

Let's compute $diag((X^TX)^{-1})$ for the two models.

```
solve(crossprod(X))
```

```
##             (Intercept)         xB         xC
## (Intercept)   0.3333333 -0.3333333 -0.3333333
## xB           -0.3333333  0.6666667  0.3333333
## xC           -0.3333333  0.3333333  0.6666667
```

```
solve(crossprod(X2))
```

```
##             (Intercept)        x2B
## (Intercept)   0.3333333 -0.3333333
## x2B          -0.3333333  0.6666667
```

So, $diag((X^TX)^{-1})$ for the two models's shared parameters are the same. var(y) was the one who made all the differences. How was var(y) computed?

In a given linear regression model, var(y) is the variation of the model residules given fixed covariates.

```
(res = y - X %*% beta)
```

```
##          [,1]
## 1 -0.20030744
## 2  0.60978969
## 3 -0.40948225
## 4  1.22717407
## 5 -0.03859896
## 6 -1.18857511
## 7 -0.11308265
## 8  0.13781300
## 9 -0.02473035
```

```
(res2 = y2 - X2 %*% beta2)
```

```
##          [,1]
## 1 -0.20030744
## 2  0.60978969
## 3 -0.40948225
## 4  1.22717407
## 5 -0.03859896
## 6 -1.18857511
```

We noticed model residuals for shared data are also the same because of the same parameter estimates.

To compute residual variation, R::lm() use: $SSR/(N-p)$\$, where SSR is residual sum of squares, N is sample size, p is degree of freedom.

```r
N = 9
p = 3

(var_res = var(res) * (N - 1) / (N - p))
```

```
##           [,1]
## [1,] 0.5886985
```

```r
var_beta = var_res * diag(solve(crossprod(X)))
(se_beta = sqrt(var_beta))
```

```
## [1] 0.4429817 0.6264708 0.6264708
```

```r
fit$coefficients[, "Std. Error"]
```

```
## (Intercept)          xB          xC
##   0.4429817   0.6264708   0.6264708
```

```r
N2 = 6
p2 = 2

(var_res2 = var(fit2$residuals) * (N2 - 1) / (N2 - p2))
```

```
## [1] 0.8749498
```

```r
var_beta2 = var_res2 * diag(solve(crossprod(X2)))
(se_beta2 = sqrt(var_beta2))
```

```
## (Intercept)         x2B
##   0.5400462   0.7637407
```

```r
fit2$coefficients[, "Std. Error"]
```

```
## (Intercept)         x2B
##   0.5400462   0.7637407
```

```r
y[7:9] <- y[7:9] + rnorm(3, 0, 10)

fit_new <- lm(y ~ x) %>% summary

fit_new$coefficients[, "Std. Error"]
```

```
## (Intercept)          xB          xC
##    3.129934    4.426396    4.426396
```

```r
fit2$coefficients[, "Std. Error"]
```

```
## (Intercept)         x2B
##   0.5400462   0.7637407
```