GU4205/5205-Linear Regression Models-Lab2b

Authors: Dr. Banu Baydil, Dr. Ronald Neath

Fall 2022

Section 1: WLS Model

In this section we will work with the physics data set.

```
library(alr4)
attach(physics)
dim(physics)
```

```
## [1] 10  3
```

```
names(physics)
```

```
## [1] "x"  "y"  "SD"
```

```
physics
```

```
##        x   y SD
## 1  0.345 367 17
## 2  0.287 311  9
## 3  0.251 295  9
## 4  0.225 268  7
## 5  0.207 253  7
## 6  0.186 239  6
## 7  0.161 220  6
## 8  0.132 213  6
## 9  0.084 193  5
## 10 0.060 192  5
```

Weights are given through the SD variable.

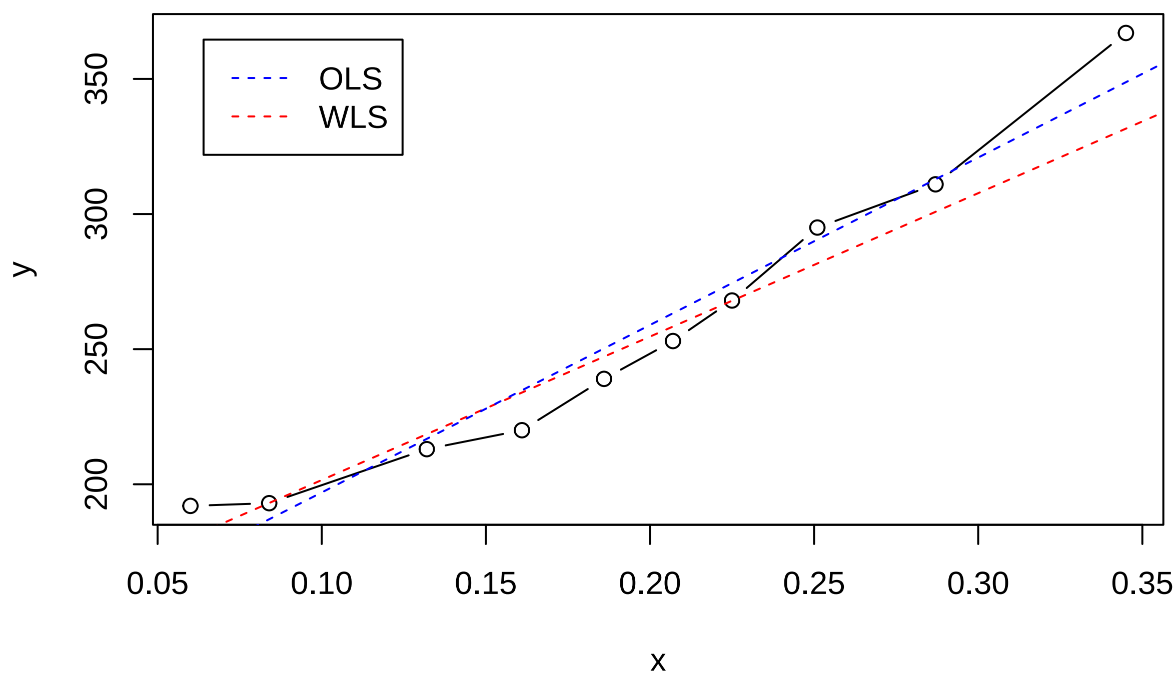
```
m1 <- lm(y ~ x, weights=1/SD^2, data=physics)
summary(m1)
```

```
##
## Call:
## lm(formula = y ~ x, data = physics, weights = 1/SD^2)
##
## Weighted Residuals:
##     Min      1Q  Median      3Q     Max
## -2.3230 -0.8842  0.0000  1.3900  2.3353
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  148.473      8.079   18.38 7.91e-08 ***
## x            530.835     47.550   11.16 3.71e-06 ***
```

```
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.657 on 8 degrees of freedom
## Multiple R-squared:  0.9397, Adjusted R-squared:  0.9321
## F-statistic: 124.6 on 1 and 8 DF,  p-value: 3.71e-06
```

Let us compare WLS with OLS:

```
plot(y ~ x, data=physics, type="b")
abline(m1, lty=2, col="red")
abline(lm(y~x,data=physics), lty=2, col="blue")
legend("topleft", inset=.05, legend=c("OLS", "WLS"),
       lty=c(2,2), col=c("blue", "red"))
```



Note that since WLS since the point x=.35 point has high variance, WLS is less concerned with fitting.

Next, let us try a quadratic mean function:

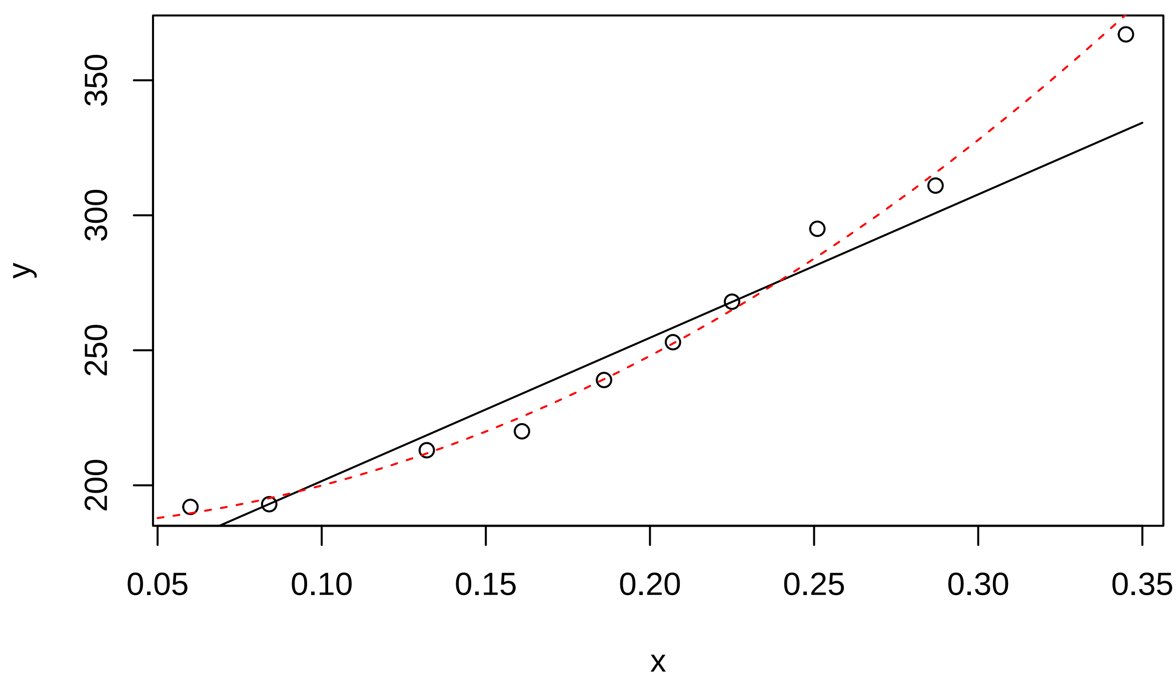
```
m2 <- lm(y ~ x + I(x^2), weights=1/SD^2, data=physics)
summary(m2)
```

```
##
## Call:
## lm(formula = y ~ x + I(x^2), data = physics, weights = 1/SD^2)
##
## Weighted Residuals:
##      Min       1Q   Median       3Q      Max
## -0.89928 -0.43508  0.01374  0.37999  1.14238
```

```
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  183.8305     6.4591  28.461  1.7e-08 ***
## x              0.9709    85.3688   0.011 0.991243
## I(x^2)      1597.5047   250.5869   6.375 0.000376 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.6788 on 7 degrees of freedom
## Multiple R-squared:  0.9911, Adjusted R-squared:  0.9886
## F-statistic: 391.4 on 2 and 7 DF,  p-value: 6.554e-08
```

Let us compare the two WLS models:

```
plot(y ~ x, data=physics)
x <- seq(.05, .35, .01)
lines(x, predict(m1, data.frame(x=x)))
lines(x, predict(m2, data.frame(x=x)), lty=2, col="red")
```



We can estimate mean response given x = .215:

```
yhat <- predict(m2, newdata=data.frame(x=.215), se.fit=T)
as.numeric( yhat$fit)
```

```
## [1] 257.8839
```

The standard error of the estimate is given by:

```
yhat$se.fit
```

```
## [1] 1.988875
```

Confidence interval for mean response at x=.215 is given by:

```
predict(m2, data.frame(x=.215), interval="confidence")
```

```
##        fit      lwr      upr
## 1 257.8839 253.1809 262.5868
```

Prediction interval for single response x=.215 is given by:

```
predict(m2, data.frame(x=.215), weights=1/49, interval="prediction")
```

```
##        fit      lwr      upr
## 1 257.8839 245.7033 270.0644
```

Note that above weight for x=.215 needs to be specified as discussed in class.

Section 2: Sandwich Estimator of Var(beta.hat)

In this section we will work with the sniffer data set.

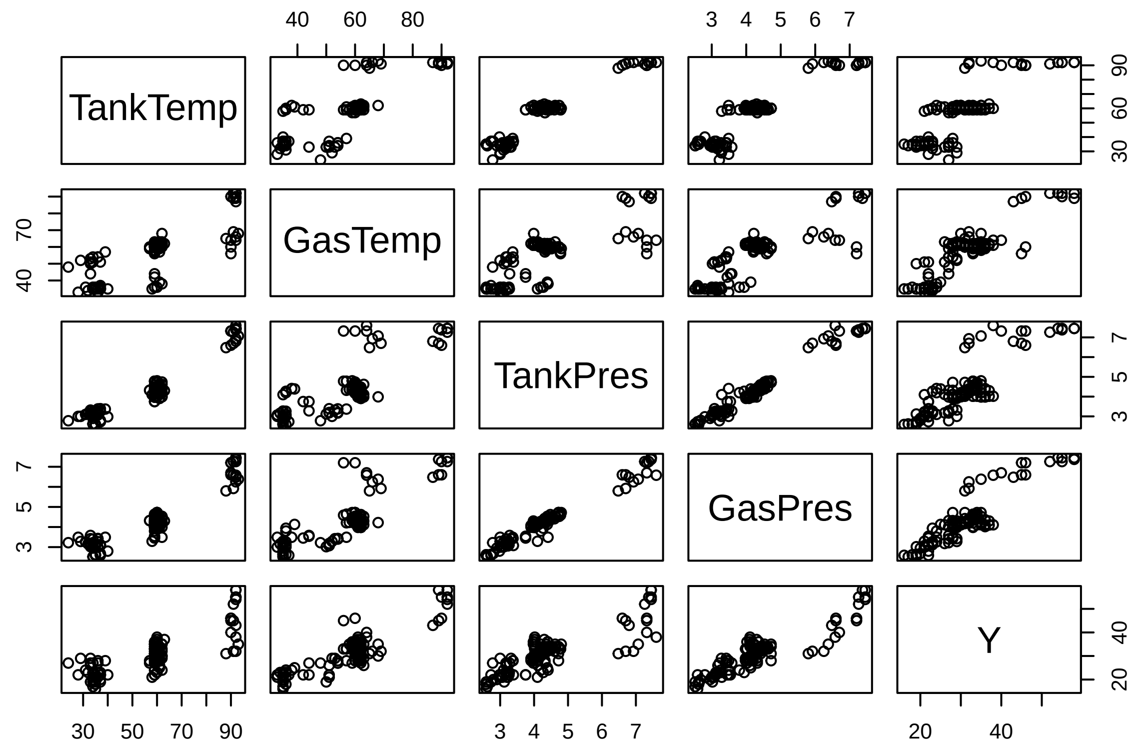
```
rm(list=ls());
library(alr4);
dim(sniffer);
```

```
## [1] 125   5
```

```
names(sniffer);
```

```
## [1] "TankTemp" "GasTemp"  "TankPres" "GasPres"  "Y"
```

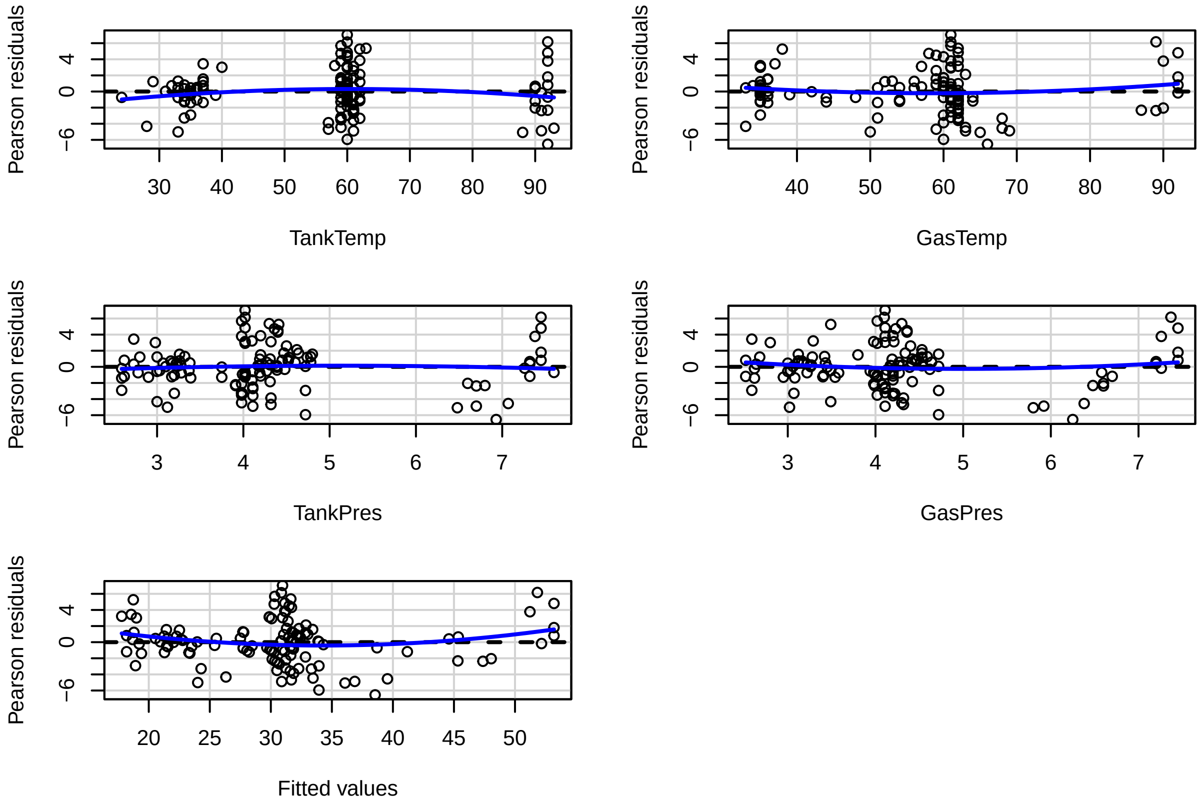
```
pairs(sniffer)
```




```r
m1 <- lm(Y ~ ., data=sniffer)
summary(m1)
```

```
##
## Call:
## lm(formula = Y ~ ., data = sniffer)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -6.5425 -1.2938  0.0495  1.2259  7.0413
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.15391    1.03489   0.149   0.8820
## TankTemp    -0.08269    0.04857  -1.703   0.0912 .
## GasTemp      0.18971    0.04118   4.606 1.03e-05 ***
## TankPres    -4.05962    1.58000  -2.569   0.0114 *
## GasPres      9.85744    1.62515   6.066 1.57e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.758 on 120 degrees of freedom
## Multiple R-squared:  0.8933, Adjusted R-squared:  0.8897
## F-statistic: 251.1 on 4 and 120 DF,  p-value: < 2.2e-16
```

```
residualPlots(m1) #Note that constant variance assumption may not be appropriate for this dataset.
```




```
##            Test stat Pr(>|Test stat|)
## TankTemp     -2.5729          0.01131 *
## GasTemp       1.4691          0.14445
## TankPres     -0.6098          0.54313
## GasPres       1.1789          0.24080
## Tukey test    2.4770          0.01325 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
coef(m1) #OLS estimates are still reasonable
```

```
## (Intercept)    TankTemp     GasTemp    TankPres     GasPres
##  0.15390800 -0.08269487  0.18970676 -4.05961741  9.85744062
```

```
sqrt(diag(vcov(m1))) #But standard errors may not be valid
```

```
## (Intercept)    TankTemp     GasTemp    TankPres     GasPres
##  1.03488948  0.04856801  0.04118370  1.58000429  1.62515157
```

We can use a sandwich estimator. The function hccm automates computation of sandwich estimator.

```
sandwich <- hccm(m1)
sqrt(diag(sandwich))
```

```
## (Intercept)    TankTemp     GasTemp    TankPres     GasPres
##  1.04734551  0.04444225  0.03380072  1.97239041  2.05585118
```

We can carry out nonconstant variance test where, NH: Var(Y|X=x) = sigma^2 and AH: Var(Y|X=x) = sigma^2 * exp(lambda*x):

```
ncvTest(m1, ~ TankTemp + GasTemp + TankPres + GasPres)
```

```
## Non-constant Variance Score Test
## Variance formula: ~ TankTemp + GasTemp + TankPres + GasPres
## Chisquare = 13.75993, Df = 4, p = 0.008102
```

We have above fairly strong evidence against the constant variance model.

If instead we test:

```
ncvTest(m1)
```

```
## Non-constant Variance Score Test
## Variance formula: ~ fitted.values
## Chisquare = 4.802652, Df = 1, p = 0.028416
```

then this tests the model Var(Y|X=x) = sigma^2 * exp{E(Y|X=x)}.

Section 3: Delta method

In this section we will work with the cake data set.

```
rm(list=ls());
library(alr4);
cakes
```

```
##    block       X1       X2    Y
## 1      0 33.00000 340.0000 3.89
## 2      0 37.00000 340.0000 6.36
## 3      0 33.00000 360.0000 7.65
## 4      0 37.00000 360.0000 6.79
## 5      0 35.00000 350.0000 8.36
## 6      0 35.00000 350.0000 7.63
## 7      0 35.00000 350.0000 8.12
## 8      1 37.82843 350.0000 8.40
## 9      1 32.17157 350.0000 5.38
## 10     1 35.00000 364.1421 7.00
## 11     1 35.00000 335.8579 4.51
## 12     1 35.00000 350.0000 7.81
## 13     1 35.00000 350.0000 8.44
## 14     1 35.00000 350.0000 8.06
```

```
m12 <- lm(Y ~ X1 + X2 + I(X1^2) + I(X2^2) + I(X1*X2), data=cakes)
summary(m12)
```

```
##
## Call:
## lm(formula = Y ~ X1 + X2 + I(X1^2) + I(X2^2) + I(X1 * X2), data = cakes)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -0.4912 -0.3080  0.0200  0.2658  0.5454
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
```

```
## (Intercept) -2.204e+03  2.416e+02  -9.125 1.67e-05 ***
## X1           2.592e+01  4.659e+00   5.563 0.000533 ***
## X2           9.918e+00  1.167e+00   8.502 2.81e-05 ***
## I(X1^2)     -1.569e-01  3.945e-02  -3.977 0.004079 **
## I(X2^2)     -1.195e-02  1.578e-03  -7.574 6.46e-05 ***
## I(X1 * X2)  -4.163e-02  1.072e-02  -3.883 0.004654 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.4288 on 8 degrees of freedom
## Multiple R-squared:  0.9487, Adjusted R-squared:  0.9167
## F-statistic:  29.6 on 5 and 8 DF,  p-value: 5.864e-05
```

We can estimate optimal baking time when temperature is 350 degrees:

```
beta.hat <- coef(m12)
beta.hat
```

```
##  (Intercept)           X1           X2      I(X1^2)      I(X2^2)   I(X1 * X2)
## -2204.484987    25.917558     9.918267    -0.156875    -0.011950    -0.041625
```

```
beta.hat <- as.vector(beta.hat)
b1 <- beta.hat[2];
b11 <- beta.hat[4];
b12 <- beta.hat[6];

x1_opt.hat <- -(b1 + 350*b12) / (2*b11)
x1_opt.hat   #Optimal baking time is estimated to be 36 minutes and 10 seconds
```

```
## [1] 36.1715
```

Next we can approximate its standard error by delta method and get a confidence interval. The car function deltaMethod() automates this process. The syntax takes a bit getting of used to.

```
Names <- c("b0","b1","b2","b11","b22","b12")
g <- " -(b1 + 350*b12) / (2*b11) "
deltaMethod(m12, g=g, parameterNames=Names)
```

```
##                              Estimate       SE    2.5 % 97.5 %
## -(b1 + 350 * b12)/(2 * b11) 36.17150  0.38096 35.42482 36.918
```