ToothGrowth_SLR Analysis

Introduction

This is a statistical study of the `ToothGrowth` data set. In particular we will try to fit a `simple linear model` to the data and perform the most important tests for the statistical inference on the data set.

```r
data("ToothGrowth")
db <- ToothGrowth
str(db)
```

```
'data.frame':   60 obs. of  3 variables:
 $ len : num  4.2 11.5 7.3 5.8 6.4 10 11.2 11.2 5.2 7 ...
 $ supp: Factor w/ 2 levels "OJ","VC": 2 2 2 2 2 2 2 2 2 2 ...
 $ dose: num  0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 ...
```

BEFORE -> Simple Linear Regression Model

Assumptions

In order to create a simple linear regression model on our data set we need to state the assumptions and check for their respectability. We are going to use "len" as the dependent variable and "dose" as the independent one. Let's write down the assumptions:

1. **EXISTENCE:** for any fixed value of the variable X, Y is a random variable with a certain probability distribution having `finite mean` and `finite variance`.
2. **LINEARITY:** the mean value of Y (or a transformation of Y), _Y|X =E(Y), is a `straight-line function` of X (or a transformation of X).
3. **INDEPENDENCE:** the errors, _i, are `independent` (i.e. Y-values are statistically independent of one another).
4. **HOMOSCEDAISTICITY:** the errors, _i, at each value of the predictor, x_i, have `equal variance` (i.e., the variance of Y is the same for any X).

5. **NORMALITY:** the errors, _i, at each value of the predictor, x_i, are `normally distributed` (i.e., for any fixed value of X, Y has a normal distribution).

By the design of study we know that the Y values, namely the observations are independent to each other (3rd assumption met) and that for any fixed value of X, the sub-population of Y has a finite variance and finite mean (1st assumption met).

We are going to check if there really exist a linear relationship between the 2 variables and we are going to investigate the normality of the residuals and their homoscedaisticity.

Residuals

First of all we need to compute the residuals "Y-Y_hat" where "Y_hat" is the predicted value of Y by our simple linear model.

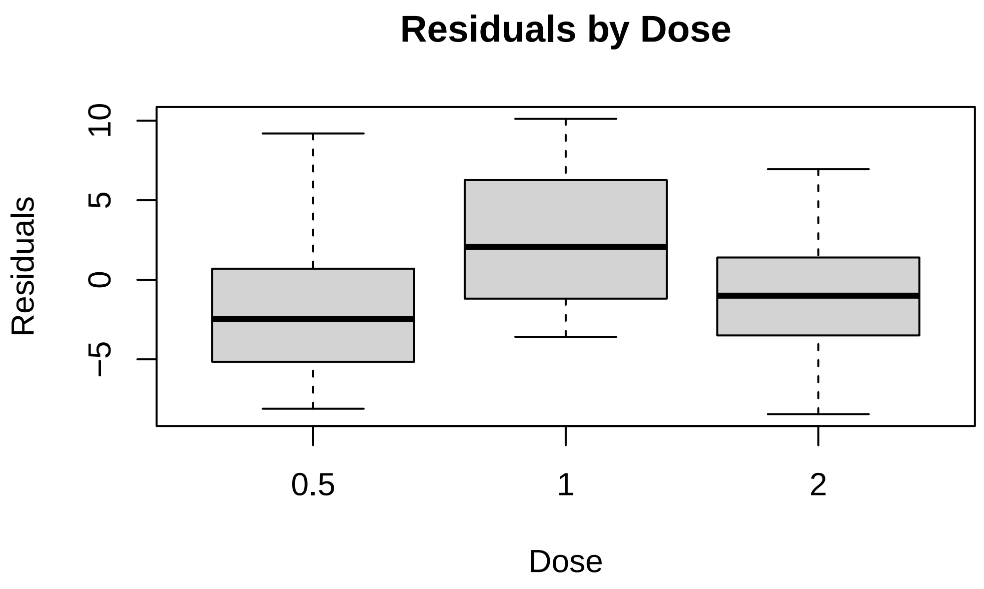
```r
fit <- lm(len ~ dose, data = db)
res <- residuals (fit)
db$residuals_dose <- res
head(db$residuals_dose)
```

```
[1] -8.1042857 -0.8042857 -5.0042857 -6.5042857 -5.9042857 -2.3042857
```

Homoscedaisticity

Let's check the equal variance of the residuals, namely for any fixed value of X the variances of Y, so ^2_Y|X, must be equal. We can check this assumption by using a `visual interpretation` (box-plot) or by using a statistical test like `studentized Breusch-Pagan test`.

```r
boxplot(resid(fit) ~ db$dose,
        xlab = "Dose",
        ylab = "Residuals",
        main = "Residuals by Dose")
```



By observing the box-plots we are not able to deduce the homoscedaisticity of the data but we can observe a pretty equal distribution of the data around the mean for each group.

Let's go through the Breusch-Pagan test which aim is to identify the presence of heteroscedaisticity inj a regression analysis.

```r
library(lmtest)
bptest(fit)
```

```

	studentized Breusch-Pagan test

data:  fit
BP = 1.3602, df = 1, p-value = 0.2435
```

Since the **p-value** $=$ **0.2435** is greater than **0.05**, we fail to reject the null hypothesis. This means there is `no significant evidence` of heteroscedasticity in the model.

Normality

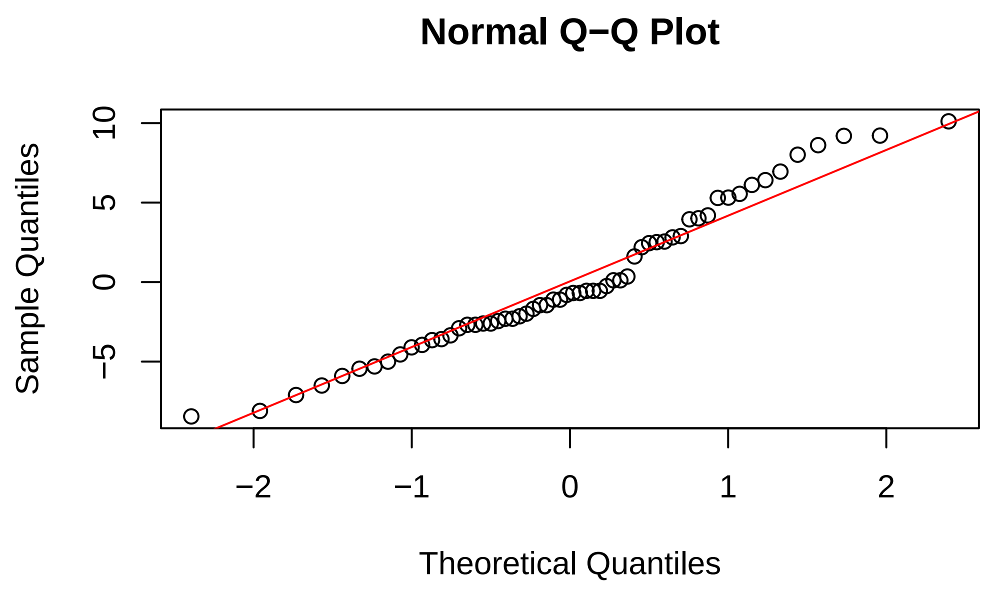
Check the 5th assumption, namely the normality of our residuals through a the `Shapiro-Wilk` statistical test. Since the size of our samples are < 50 observations this is the most suited test for our purpose.

```
res_by_dose <- split(db, db$dose)
for (group in res_by_dose){
  result <- shapiro.test(group$residuals_dose)
  print(paste(" '",unique(group$dose),"' -> p-value: ", result$p.value," ", sep=""))
}
```

```
[1] " '0.5' -> p-value: 0.246601486196049 "
[1] " '1' -> p-value: 0.163882141233334 "
[1] " '2' -> p-value: 0.901911496439856 "
```

For all the fixed value of X, Y has a normal distribution, since the `p-value > 0.05` for every population of Y given X analyzed.

```
qqnorm(db$residuals_dose)
qqline(db$residuals_dose, col = "red")
```



RUN -> Simple Linear Regression Model

Now that we have checked most of the assumptions let's build the linear model and check for linearity.

We will start with a simple linear model `y =  0 +  1*x +  ` using `dose` as the variable"x".

```r
fit <- lm(len ~ dose, data = db)
```

```r
coefs <- coef(fit)
print(coefs)
```

```
(Intercept)        dose
   7.422500    9.763571
```

```r
confint(fit, level = 0.95)
```

```
               2.5 %    97.5 %
(Intercept) 4.900171  9.944829
dose        7.856870 11.670273
```

0 - C.I.

Is the intercept and in that case it represents the mean value of the length of the odontoblasts at x(dose) = 0. To find this value we are extrapolating since in our data set we do not have observations with a value of dose = 0. Since extrapolation with SLR is not a reliable method we do not care about the meaning of the intercept in the real world.

- 95% Confidence Interval = 7.42 ± 2.52

1 - Linearity

We are going to run an hypothesis testing on the parameter Beta1 to check whether there exist a linear relationship between the dependent variable Y and the independent variable X (dose).

- **H0:** beta1 = 0 -> no linear relationship
- **H1:** beta1 != 0 -> presence of a linear relationship between Y and X

From the theory we know that if all the previous assumptions are met (normality, independence, existence and homoscedaisticity) then the estimate of the parameter 1(namely beta1_hat) is a random variable described by a `t-distribution` with "n-2" degrees of freedom. So, let's compute the t-statistic.

$$t_{\widehat{\beta}} = \frac{\widehat{\beta} - \beta_0}{\mathrm{SE}(\widehat{\beta})}$$

```r
coefs
```

```
(Intercept)        dose 
   7.422500    9.763571 
```

```r
beta1_hat <- as.numeric(coefs[2])
summary(fit)
```

```

Call:
lm(formula = len ~ dose, data = db)

Residuals:
    Min      1Q  Median      3Q     Max 
-8.4496 -2.7406 -0.7452  2.8344 10.1139 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept)   7.4225     1.2601    5.89 2.06e-07 ***
dose          9.7636     0.9525   10.25 1.23e-14 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 4.601 on 58 degrees of freedom
Multiple R-squared:  0.6443,    Adjusted R-squared:  0.6382 
F-statistic: 105.1 on 1 and 58 DF,  p-value: 1.233e-14
```

With a `p-value = 1.23e_14 <<< 0.05` we reject H0 and state that the true population parameter 1 != 0.

F-Test

IMPORTANT NOTE: We could have reached the same result by observing the p-value computed for the F-test applied to the ratio of variances **MSM** (mean square of the model) /

MSE (mean square of the errors), which under the null-hypothesis is equal to 1. In this context the F-test is used to test if the model including covariate(s) (in our case only the covariate "dose") results in a significant reduction of the residual sum squares (SSR) compared to a model containing only an intercept (1=0).

For SLR models the "t-test" and the "F-test" are equivalent for testing H0: 1=0.

```r
summary(fit)$fstatistic
```

```
   value     numdf     dendf 
105.0648   1.0000  58.0000 
```

```r
fstat <- summary(fit)$fstatistic
f_value <- fstat[1]
df1 <- fstat[2]
df2 <- fstat[3]
pf(f_value, df1, df2, lower.tail = FALSE)
```

```
       value 
1.232698e-14 
```

The p-value for the F-statistic is 1.23e-14 $<$ 0.05 so the test is statistically significant and we reject the null hypothesis H0: 1=0.

1 - C.I.

Let's build the confidence interval.

```r
confint(fit, level = 0.95)
```

```
               2.5 %    97.5 %
(Intercept) 4.900171  9.944829
dose        7.856870 11.670273
```

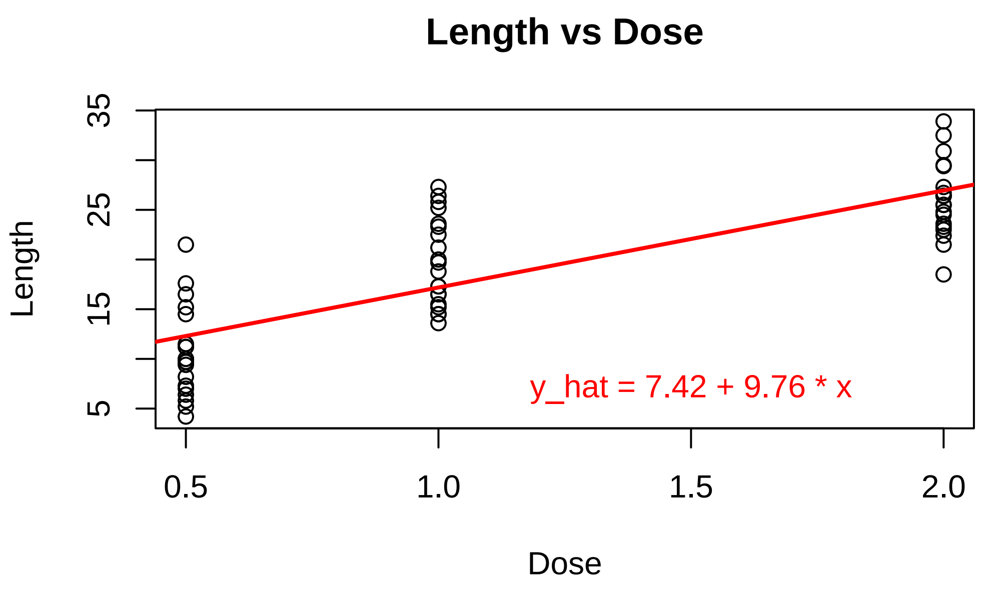
- 95% Confidence Interval $= 9.76 \pm 1.91$

In that way we have both find the best estimates of the parameters 0 and 1 and checked for the linearity of the data.

Resulting fitting line

The resulting fitting line is described by the following equation: **Y__i__hat = 7.42 + 9.76 X__i**

```r
plot(db$dose, db$len,
     main = "Length vs Dose",
     xlab = "Dose",
     ylab = "Length")
abline(fit, col = "red", lwd = 2)
intercept <- round(coefs[1], 2)
slope <- round(coefs[2], 2)
eq_text <- paste0("y_hat = ", intercept, " + ", slope, " * x")
x_pos <- 1.5
y_pos <- intercept + slope * x_pos
text(x = x_pos, y = min(db$len), labels = eq_text, pos = 3, col = "red")
```



Goodness of fit

How much variation of the values of the dependent variable Y ,the independent variable X is able to explain? To answer this question we are going to compute the Adjusted R^2 value.

```r
fit <- lm(len ~ dose, data = db)
summary(fit)
```

```
Call:
lm(formula = len ~ dose, data = db)

Residuals:
    Min      1Q  Median      3Q     Max 
-8.4496 -2.7406 -0.7452  2.8344 10.1139 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept)   7.4225     1.2601    5.89 2.06e-07 ***
dose          9.7636     0.9525   10.25 1.23e-14 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 4.601 on 58 degrees of freedom
Multiple R-squared:  0.6443,	Adjusted R-squared:  0.6382 
F-statistic: 105.1 on 1 and 58 DF,  p-value: 1.233e-14
```

The value that we are interested in is the `"Adjusted R-squared"` which in this case is `0.6382` namely the independent variable "dose" is able to explain the **63.82%** of the variability in Y. A pretty good job for a single independent variable.

Pearson Correlation coefficient

From the theory behind SLR models we know that the `R^2` is equal to the square of the `Pearson Correlation coefficient`.

```r
pearson <- cor(db$dose, db$len, method = "pearson")
pearson^2
```

```
[1] 0.6443133
```

```r
summary(fit)$r.squared
```

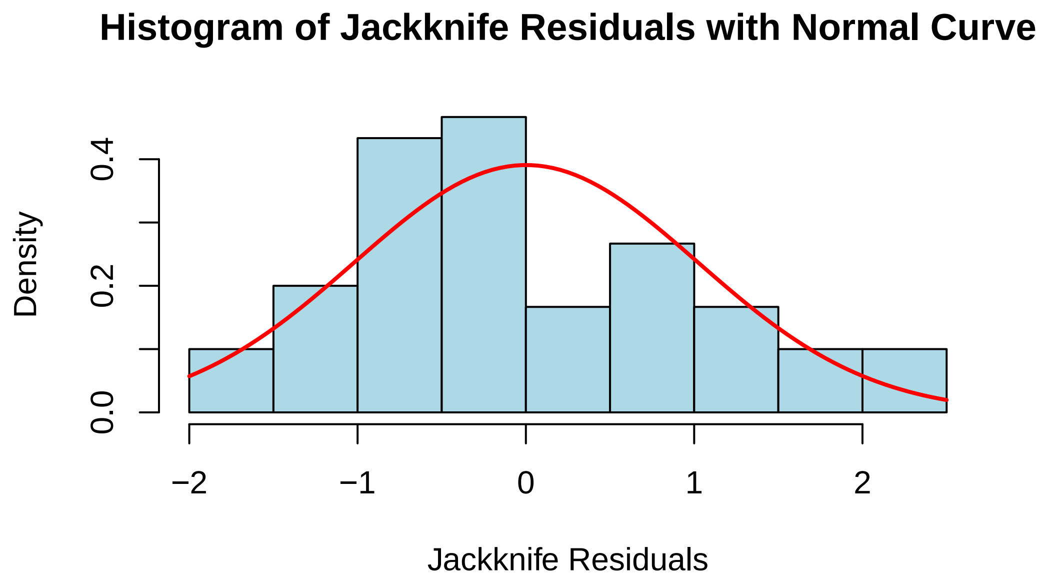
```
[1] 0.6443133
```

Jackknife residuals

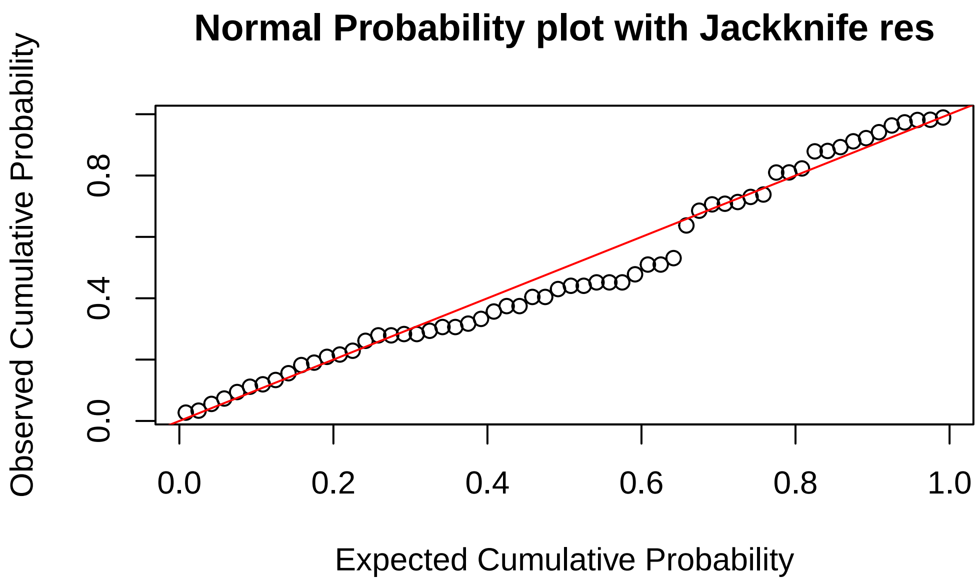
Jackknife residuals are a type of residual used in regression diagnostics to better detect outliers and influential points. They are also used to evaluate the violations of the assumptions of SLR.

```r
jk_res <- rstudent(fit)
hist_data <- hist(jk_res,
                  probability = TRUE,  # scales y-axis to density
                  main = "Histogram of Jackknife Residuals with Normal Curve",
                  xlab = "Jackknife Residuals",
                  col = "lightblue",
                  border = "black")

curve(dnorm(x, mean = mean(jk_res), sd = sd(jk_res)),
      col = "red",
      lwd = 2,
      add = TRUE)
```

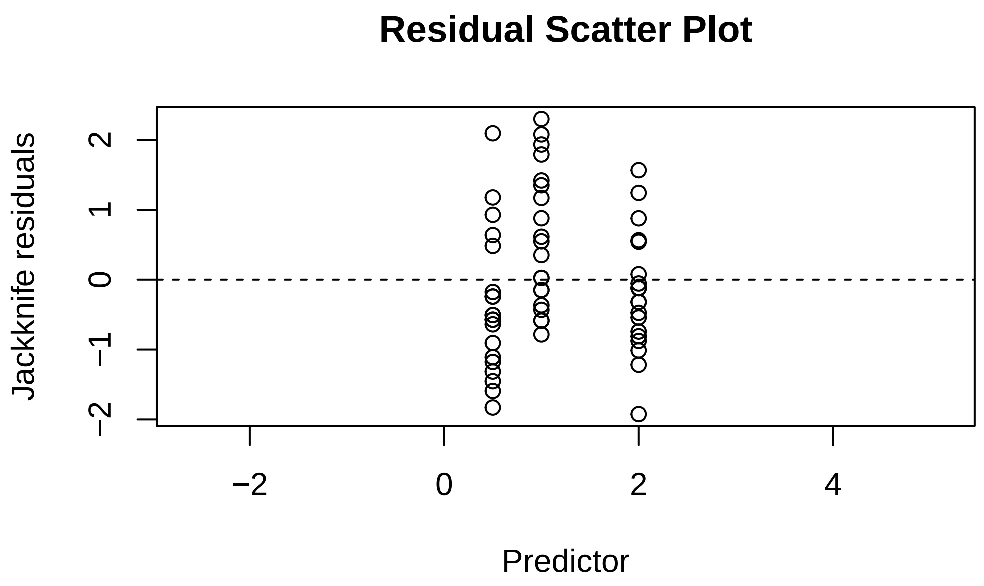


```r
plot(ppoints(length(jk_res)),
     sort(pnorm(jk_res)),
     ylab = "Observed Cumulative Probability",
     xlab = "Expected Cumulative Probability",
     main = "Normal Probability plot with Jackknife res")
abline(a=0, b=1, col='red', lwd=1)
```



Analyzing the graphs it can be stated that the jackknife residuals are `approximately normal distributed` since the data points in the middle of the "Normal Probability plot" does not fall exactly on the identity line.

```
plot(db$dose, jk_res,
     ylab = "Jackknife residuals",
     xlab = "Predictor",
     main = "Residual Scatter Plot",
     xlim = c(min(db$dose)-5*sd(db$dose), max(db$dose)+5*sd(db$dose)))
abline(a=0, b=0, lty=2)
```



This kind of diagnostic plot is generally used to evaluate the homoscedaisticty of a variable. In that case, due to the poor number of values for the predictor it is not easy to detect an heteroscedaisticity. Analyzing this graph is not enough to asses something about the variance of Y given X.

Linear Algebra in

We know that a general simple linear regression equation like `y =  0 +  1*x1 + ... +  _n*xn +  ` can be written using linear algebra notation: `Y = X* _hat + ` . In particular:

- `Y`: is the column matrix composed of the observed value in our data set
- `X`: is the design matrix, namely a m*(n+1) matrix where "m" is the number of observations in our data set and "n" is the number of independent variables in our linear regression model. In the case of SLR we have only 1 independent variable so X is a "m by 2" matrix.
- ` _hat` : is the column vector composed of the estimate of the true population parameters.

Y

```r
head(db)
```

```
   len supp dose residuals_dose
1  4.2   VC  0.5     -8.1042857
2 11.5   VC  0.5     -0.8042857
3  7.3   VC  0.5     -5.0042857
4  5.8   VC  0.5     -6.5042857
5  6.4   VC  0.5     -5.9042857
6 10.0   VC  0.5     -2.3042857
```

```r
len_values <- db$len
Y <- matrix(len_values)
print(Y)
```

```
      [,1]
 [1,]  4.2
 [2,] 11.5
 [3,]  7.3
 [4,]  5.8
 [5,]  6.4
 [6,] 10.0
 [7,] 11.2
 [8,] 11.2
 [9,]  5.2
[10,]  7.0
[11,] 16.5
[12,] 16.5
[13,] 15.2
[14,] 17.3
[15,] 22.5
[16,] 17.3
[17,] 13.6
[18,] 14.5
[19,] 18.8
[20,] 15.5
[21,] 23.6
[22,] 18.5
[23,] 33.9
[24,] 25.5
[25,] 26.4
```

```
[26,] 32.5
[27,] 26.7
[28,] 21.5
[29,] 23.3
[30,] 29.5
[31,] 15.2
[32,] 21.5
[33,] 17.6
[34,]  9.7
[35,] 14.5
[36,] 10.0
[37,]  8.2
[38,]  9.4
[39,] 16.5
[40,]  9.7
[41,] 19.7
[42,] 23.3
[43,] 23.6
[44,] 26.4
[45,] 20.0
[46,] 25.2
[47,] 25.8
[48,] 21.2
[49,] 14.5
[50,] 27.3
[51,] 25.5
[52,] 26.4
[53,] 22.4
[54,] 24.5
[55,] 24.8
[56,] 30.9
[57,] 26.4
[58,] 27.3
[59,] 29.4
[60,] 23.0
```

Design matrix X

```r
dose_values <- db$dose
X <- matrix(c(rep(1, length(Y)), dose_values), ncol = 2)
print(X)
```

```
      [,1] [,2]
 [1,]    1  0.5
 [2,]    1  0.5
 [3,]    1  0.5
 [4,]    1  0.5
 [5,]    1  0.5
 [6,]    1  0.5
 [7,]    1  0.5
 [8,]    1  0.5
 [9,]    1  0.5
[10,]    1  0.5
[11,]    1  1.0
[12,]    1  1.0
[13,]    1  1.0
[14,]    1  1.0
[15,]    1  1.0
[16,]    1  1.0
[17,]    1  1.0
[18,]    1  1.0
[19,]    1  1.0
[20,]    1  1.0
[21,]    1  2.0
[22,]    1  2.0
[23,]    1  2.0
[24,]    1  2.0
[25,]    1  2.0
[26,]    1  2.0
[27,]    1  2.0
[28,]    1  2.0
[29,]    1  2.0
[30,]    1  2.0
[31,]    1  0.5
[32,]    1  0.5
[33,]    1  0.5
[34,]    1  0.5
[35,]    1  0.5
[36,]    1  0.5
[37,]    1  0.5
[38,]    1  0.5
[39,]    1  0.5
[40,]    1  0.5
[41,]    1  1.0
[42,]    1  1.0
```

```
[43,]    1  1.0
[44,]    1  1.0
[45,]    1  1.0
[46,]    1  1.0
[47,]    1  1.0
[48,]    1  1.0
[49,]    1  1.0
[50,]    1  1.0
[51,]    1  2.0
[52,]    1  2.0
[53,]    1  2.0
[54,]    1  2.0
[55,]    1  2.0
[56,]    1  2.0
[57,]    1  2.0
[58,]    1  2.0
[59,]    1  2.0
[60,]    1  2.0
```

Beta hat

```
beta_hat_vec <- solve(crossprod(X)) %*% crossprod(X,Y)
beta_hat_vec
```

```
         [,1]
[1,] 7.422500
[2,] 9.763571
```

```
print(beta_hat_vec[1]) # is the estimate of the parameter Beta0
```

```
[1] 7.4225
```

```
print(beta_hat_vec[2]) # is the estimate of the parameter Beta1
```

```
[1] 9.763571
```

Let's check with the "lm()" function

```r
fit <- lm(len ~ dose, data = db)
summary(fit)$coefficients
```

```
            Estimate Std. Error   t value     Pr(>|t|)
(Intercept) 7.422500  1.2600826  5.890487 2.064211e-07
dose        9.763571  0.9525329 10.250114 1.232698e-14
```

MSE

$$\frac{SSE}{n-p-1} = \frac{\mathbf{Y}^\top\mathbf{Y} - \hat{\beta}^\top\mathbf{X}^\top\mathbf{Y}}{n-p-1}.$$

Using this formula we can compute the MSE, namely the estimate of the variance of Y | X.

```r
mse <- (crossprod(Y) - t(beta_hat_vec) %*% crossprod(X,Y)) / (length(Y)-1-1)
mse
```

```
         [,1]
[1,] 21.17078
```

Variance Covariance matrix

We need to compute the variance covariance matrix for the parameters Beta in order to get their variances and then compute the t statistic for the hypothesis testing on them.

$$Var(\hat{\beta}) = \hat{\sigma}^2_{Y|X}(\mathbf{X}^\top\mathbf{X})^{-1}$$

```r
sigma_mat <- as.numeric(mse) *  solve(crossprod(X))
print(sigma_mat)
```

```
          [,1]      [,2]
[1,]  1.587808 -1.058539
[2,] -1.058539  0.907319
```

```r
print(sigma_mat[1,1]) # Variance of the parameter Beta0
```

```
[1] 1.587808
```

```r
print(sigma_mat[2,2]) # Variance of the parameter Beta1
```

```
[1] 0.907319
```

Variance of Beta1

We need to extract the variance of the estimate of beta1 from the variance covariance matrix

```r
c_vec <- matrix(c(0,1), nrow = 1)
beta1_var <- as.numeric(c_vec %*% sigma_mat %*% t(c_vec))
print(beta1_var)
```

```
[1] 0.907319
```

The same variance can be computed using this formula where "c_vec" is the vector used to extract the beta1 values.

$$Var(\hat{\beta}_1) = \mathbf{c}(\mathbf{X}^\top \mathbf{X})^{-1}\mathbf{c}^\top \hat{\sigma}^2_{Y|X}$$

```r
c_vec %*% solve(crossprod(X)) %*% t(c_vec) %*% as.numeric(mse)
```

```
         [,1]
[1,] 0.907319
```

T - statistic

Once we have the variance of the estimated parameter Beta1 we can compute the `t-statistic`.

```r
t_stat <- beta1_hat / sqrt(beta1_var)
print(t_stat)
```

```
[1] 10.25011
```

Compute the `p-value` for that t-statistic.

```
2 * (1 - pt(t_stat, df = length(Y)-1-1))
```

```
[1] 1.24345e-14
```

```
summary(fit)$coefficients
```

```
            Estimate Std. Error   t value     Pr(>|t|)
(Intercept) 7.422500  1.2600826  5.890487 2.064211e-07
dose        9.763571  0.9525329 10.250114 1.232698e-14
```

By comparison with the t-statistic (t-value) and the p-value (Pr(>|t|)) we can see that we have reached the same conclusion nonetheless we have used linear algebra operations.

C.I. for Beta1

```
lci <- beta1_hat - qt(0.975, df = length(Y)-2) * sqrt(beta1_var)
uci <- beta1_hat + qt(0.975, df = length(Y)-2) * sqrt(beta1_var)
print(paste(lci, " - ", uci))
```

```
[1] "7.85686959872824  -  11.6702732584146"
```

The same confidence interval that we have observed using the “lm()” function.

```
confint(fit, level = 0.95)[2,] #confidence interval computed for the parameter Beta1
```

```
   2.5 %   97.5 %
 7.85687 11.67027
```