Summary

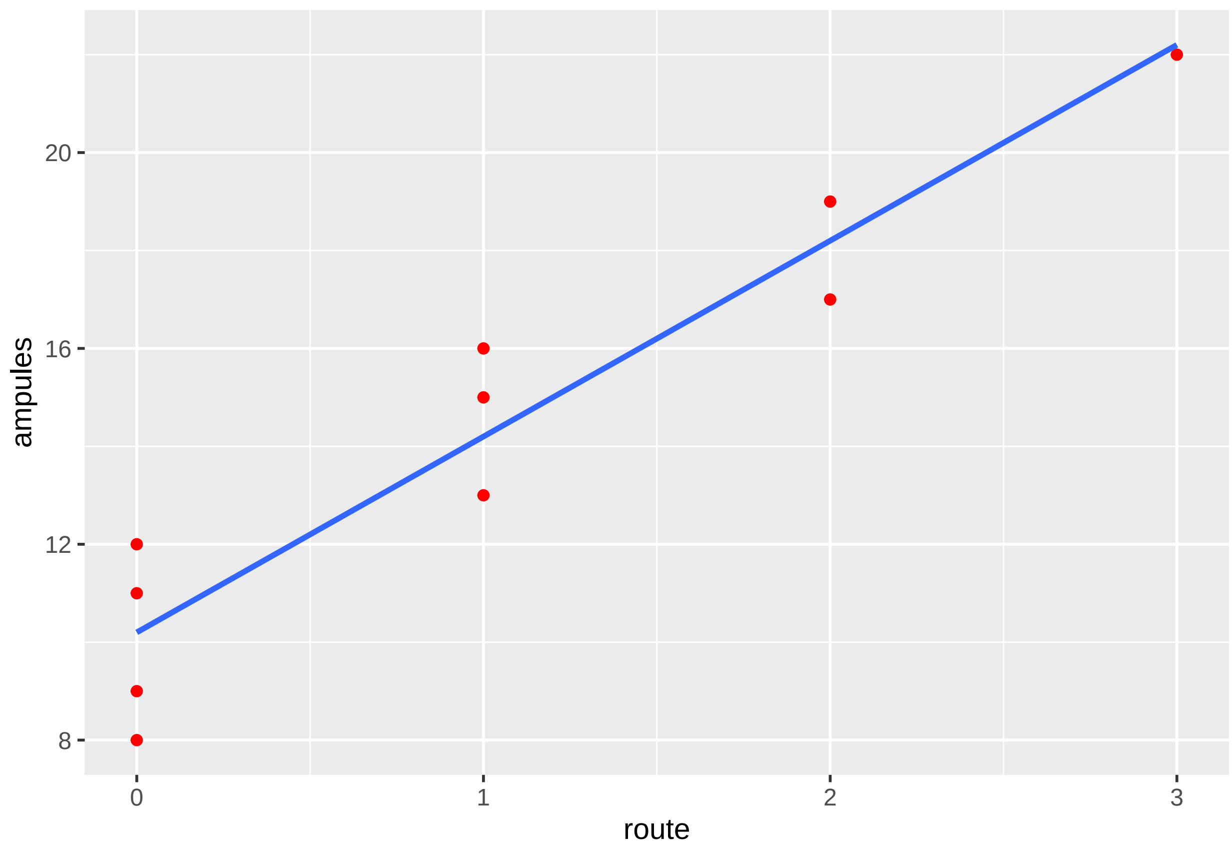
Johnson ODEJIDE

2023-03-05

Contents

Plot the estimated regression function

```
route <- c(1, 0, 2, 0, 3, 1, 0, 1, 2, 0)
ampules <- c(16, 9, 17, 12, 22, 13, 8, 15, 19, 11)

df = tibble(route, ampules)

df %>%
  ggplot(aes(x = route, y = ampules)) +
  geom_point(color = "red") +
  geom_smooth(method = "lm", se = F, formula = y ~ x)
```



Fit a regression line

```
lm.fit <- lm(ampules ~ route)
summary(lm.fit)
```

```
##
## Call:
## lm(formula = ampules ~ route)
##
## Residuals:
##    Min     1Q Median     3Q    Max
##   -2.2   -1.2    0.3    0.8    1.8
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  10.2000     0.6633  15.377 3.18e-07 ***
## route         4.0000     0.4690   8.528 2.75e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.483 on 8 degrees of freedom
## Multiple R-squared:  0.9009, Adjusted R-squared:  0.8885
## F-statistic: 72.73 on 1 and 8 DF,  p-value: 2.749e-05
```

Estimate a slope

Estimate the increase in the expected number of ampules broken when there are 2 transfers as compared to 1 transfer. (Hint: Use the slope of the model)

```
# The formula is given as
# y(hat) = 10.2 + 4x

# when x = 1
yhat_1 <- 10.2 + 4*1

yhat_1
```

```
## [1] 14.2
```

```
# Calculate for when X = 2
yhat_2 <- 10.2 + 4*2

# Estimate the increase
slope <- yhat_2 - yhat_1
slope
```

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

Find specific residual for Xi = 3. Show all of your work . Determine if the observed value of 22 is above or below average.

```
# For x = 3
# y_hat = 10.2 + 4*3
# y_hat = 10.2 + 12 = 22.2
# residual for X3 = 22.2 - 22 = 0.2 (Below average because it is negative)

yhat <- 10.2 + 4 * 3
residual <- 22 - yhat # Where 22 is the observed value
residual
```

```
## [1] -0.2
```

Conclusion: Since the residual is negative, we conclude that it is below average

Verify that your fitted regression line goes through the point (Xbar,Ybar)

```
Xbar = mean(route)
Ybar = mean(ampules)
# To verify that the fitted regression line goes through the point, we substitute x in the equation for

yhat <- 10.2 + 4 * 1
print(paste("yhat = ", yhat))
```

```
## [1] "yhat =  14.2"
```

```
print(paste("Xbar = ", Xbar, ", Ybar = ", Ybar, "Yhat at points(1, 14.2) = ", yhat))
```

```
## [1] "Xbar =  1 , Ybar =  14.2 Yhat at points(1, 14.2) =  14.2"
```

Remark:

The fitted regression line goes through the point at xbar = 1 and ybar = 14.2 since yhat and ybar are the same value.

Interpretation of the Multiple R-Squared

Which value in the R summary output table determines if your model is doing a good job explaining the variation in the dependent variable produced by the model. In this case identify this specific proportion of variation.

The Multiple R-squared does the job of explaining the variation in the dependent variable produced by the model.

In this case, the Multiple R-Squared is 0.9009 meaning that about 90% of the variability in the dependent variable (ampules) can be explained by the model.

Computation of confidence interval in R

```
X = c(1, 0, 2, 0, 3, 1, 0, 1, 2, 0)
Y = c(16, 9, 17, 12, 22, 13, 8, 15, 19, 11)

d.data <- tibble(X, Y)

conf.95 <- qt(p=.025, df=8, lower.tail = FALSE)

# conf.95

lm.fit <- lm(Y ~ X, data = d.data)
# lm.fit

# summary(lm.fit)

# qt(p=.025, df=8, lower.tail = FALSE)
# 4.00 +/- 2.306(0.469)

upper_bound <- 4.00 + 2.306 * 0.469
lower_bound <- 4.00 - 2.306 * 0.469

conf.int <- c(lower_bound, upper_bound)
conf.int
```

```
## [1] 2.918486 5.081514
```

Raw computation of confidence interval

se = 0.4690

df = 8

b +/- t(se)

t = 2.306004

4.0 +/- 2.306004(0.4690)

confidence interval = (4.0 - 2.306004(0.4690), 4.0 + 2.306004(0.4690))

= (2.9185, 5.0815)

Difference between confidence interval and prediction interval

While the prediction interval predicts in what range a future observation will fall, the confidence interval shows in what range of values the prediction falls based on some data provided already. In summary, confidence interval predicts what is available within the limits of the data while prediction interval is able to predict the future.

Prediction Interval in R

```
new_df <- data.frame(X = 19)

predict(object = lm.fit, newdata = new_df, interval = "prediction") %>%
cbind(new_df)
```

```
##    fit      lwr      upr  X
## 1 86.2 66.40325 105.9967 19
```

Confidence Interval in Prediction

```
predict(object = lm.fit, newdata = new_df, interval = "confidence") %>%
cbind(new_df)
```

```
##    fit      lwr     upr  X
## 1 86.2 66.70097 105.699 19
```

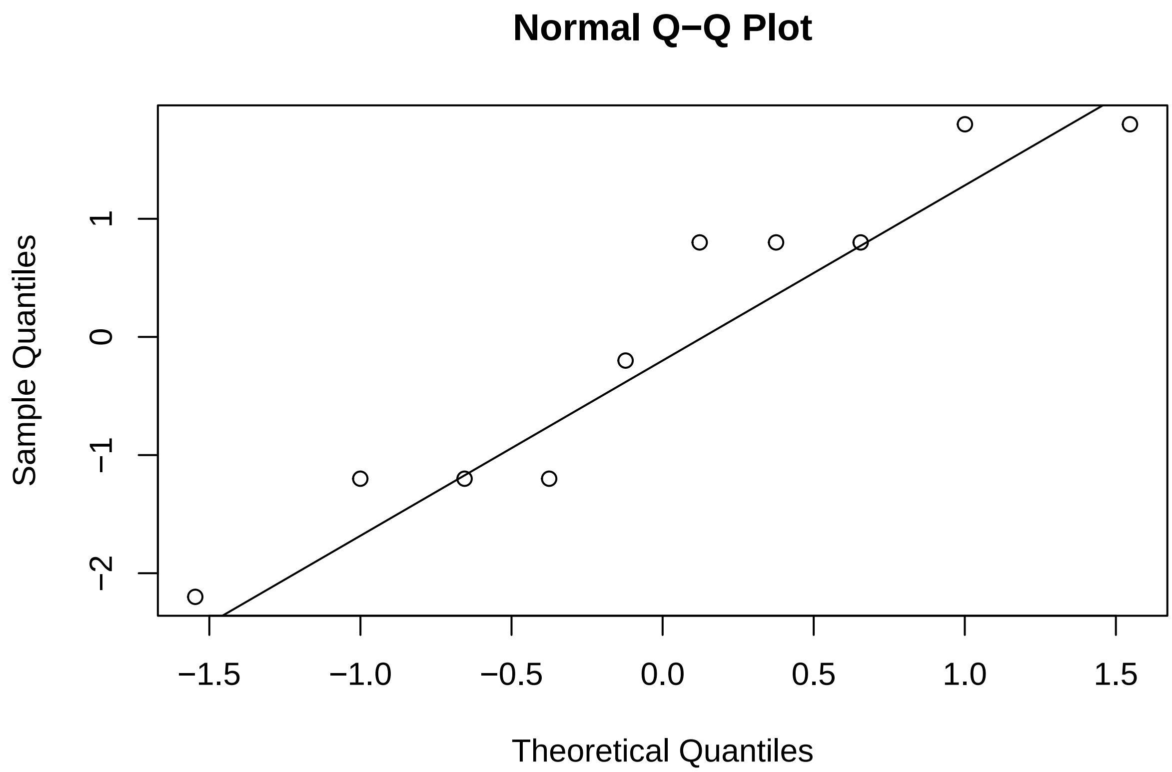
Remarks on Prediction and Confidence Interval

Prediction Interval when X = 19 (66.4033, 105.9967)

Confidence Interval when X = 19 (66.701, 105.699)

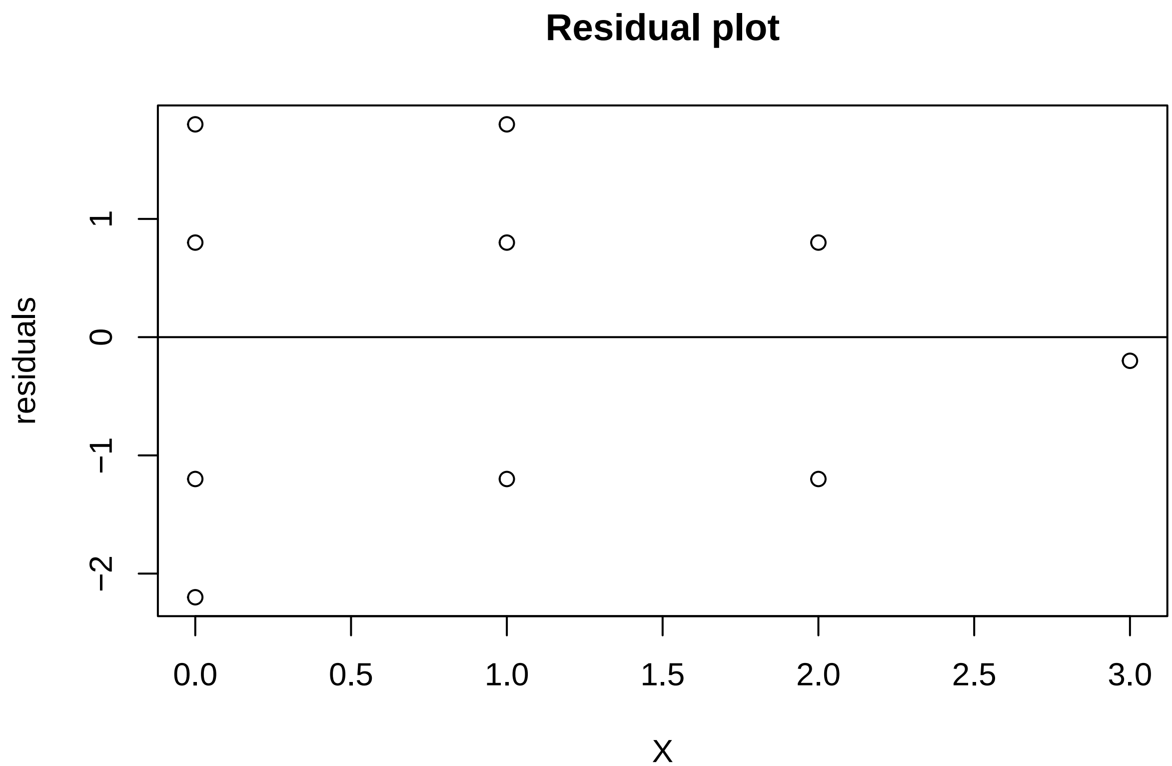
Plotting Residuals using QQplot

```
qqnorm(lm.fit$residuals)
qqline(lm.fit$residuals)
```



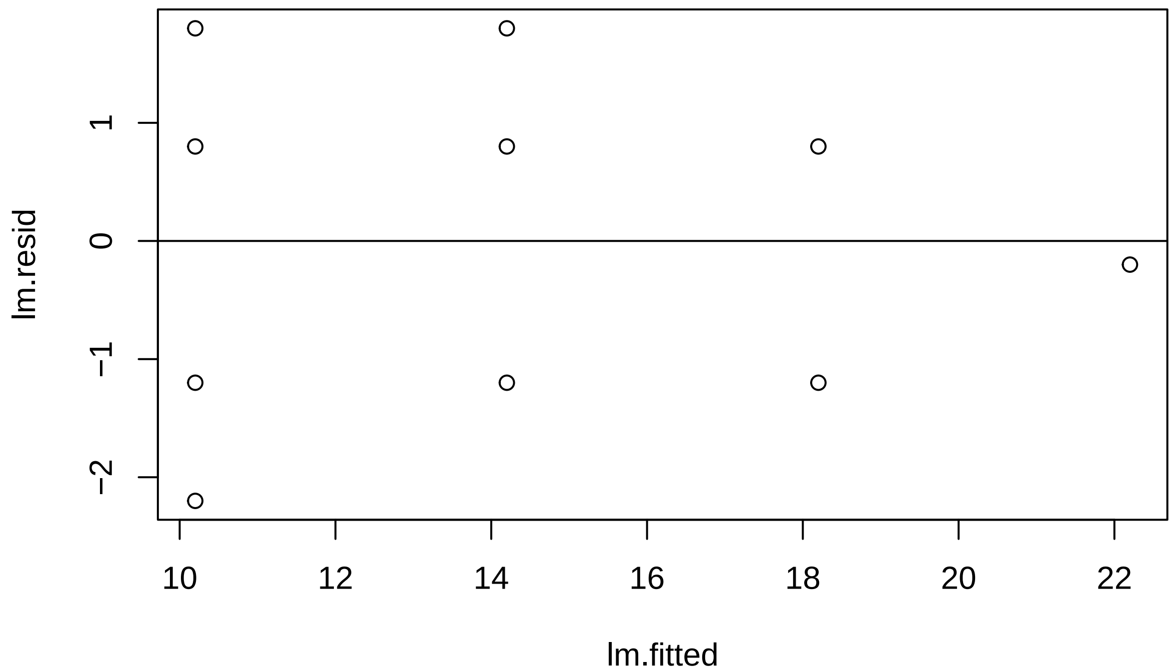
Residual plots (Scatter plot)

```
lm.resid <- resid(lm.fit)

plot(d.data$X , lm.resid,
     xlab = "X",
     ylab = "residuals",
     main = "Residual plot")
abline(0, 0)
```



Residual against Fitted Plot

```
lm.fitted <- fitted(lm.fit)

plot(lm.fitted, lm.resid)
abline(0, 0)
```



To reject null hypotheses or not?

```
x <- c(9,9,9,7,7,7,5,5,5,3,3,3,1,1,1)
y<- c(.07,.09,.08,.16,.17,.21,.49,.58,.53,1.22,1.15,1.07,2.84,2.57,3.10)

linear <- lm(y ~ x)
summary(linear)
```

```
##
## Call:
## lm(formula = y ~ x)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -0.5333 -0.4043 -0.1373  0.4157  0.8487
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   2.5753     0.2487  10.354 1.20e-07 ***
## x            -0.3240     0.0433  -7.483 4.61e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.4743 on 13 degrees of freedom
## Multiple R-squared:  0.8116, Adjusted R-squared:  0.7971
## F-statistic: 55.99 on 1 and 13 DF,  p-value: 4.611e-06
```

```r
anova(linear)
```

```
## Analysis of Variance Table
##
## Response: y
##           Df  Sum Sq Mean Sq F value    Pr(>F)
## x          1 12.5971  12.597  55.994 4.611e-06 ***
## Residuals 13  2.9247   0.225
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
f.value <- 55.994

f.critical <- qf(p = 0.025, df1 = 1, df2 = 13, lower.tail = FALSE)

print(paste("F value = ", f.value, "F Critical = ", f.critical))
```

```
## [1] "F value =  55.994 F Critical =  6.41425430025058"
```

```r
ifelse(f.value > f.critical, "Reject the null hypothesis", "Fail to reject the null hypothesis")
```

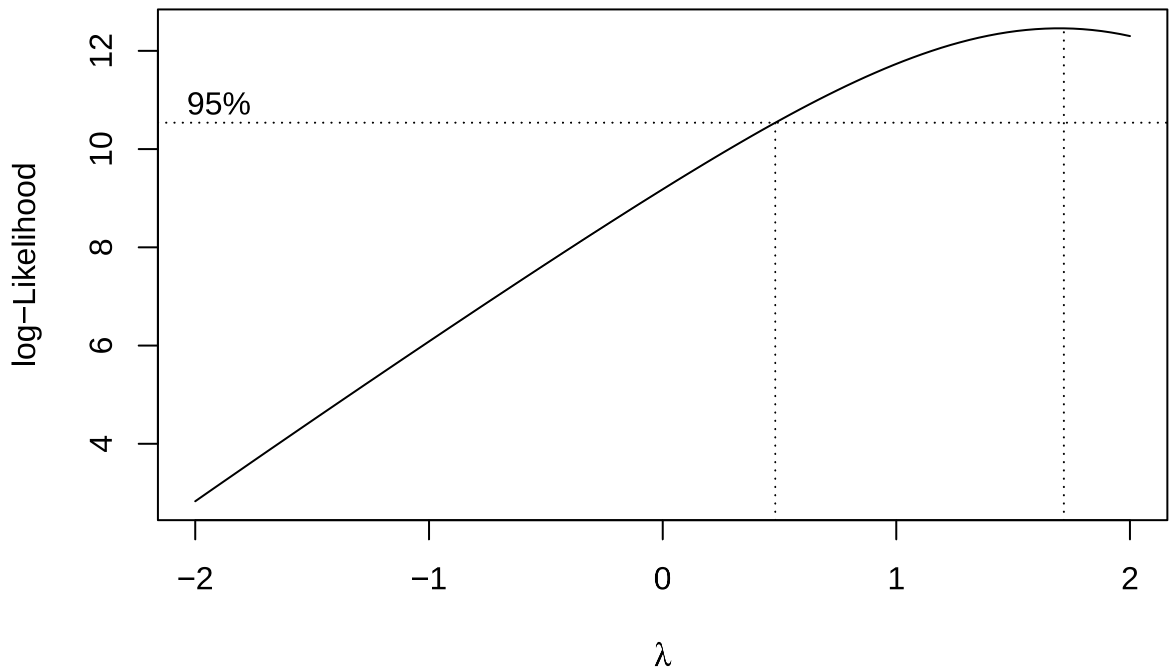
```
## [1] "Reject the null hypothesis"
```

Boxcox

```r
library(MASS)
```

```
##
## Attaching package: 'MASS'
## The following object is masked from 'package:dplyr':
##
##     select
```

```r
box_cox <- boxcox(Y ~ X)
```




```r
# Optimum lambda
lambda <- box_cox$x[which.max(box_cox$y)]
lambda
```

```
## [1] 1.717172
```

```r
new_model <- lm(((Y^lambda-1)/lambda) ~ X)
new_model
```

```
##
## Call:
## lm(formula = ((Y^lambda - 1)/lambda) ~ X)
##
## Coefficients:
## (Intercept)            X
##       29.99        27.88
```

```r
lambda.3 <- lm(((Y^.3 - 1) / .3) ~ X)
anova(lambda.3)
```

```
## Analysis of Variance Table
##
## Response: ((Y^0.3 - 1)/0.3)
##           Df Sum Sq Mean Sq F value    Pr(>F)
## X          1 3.8197  3.8197  47.646 0.0001242 ***
## Residuals  8 0.6413  0.0802
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
lambda.5 <- lm(((Y^.5 - 1) / .5) ~ X)
anova(lambda.5)
```

```
## Analysis of Variance Table
##
## Response: ((Y^0.5 - 1)/0.5)
##           Df  Sum Sq Mean Sq F value    Pr(>F)
## X          1 11.0405 11.0405  54.078 7.965e-05 ***
## Residuals  8  1.6333  0.2042
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Remark: An Appropriate transformation of Y would be Y^(0.5050505)

SSE from lambda results

SSE = 0.584 when lamda = .3 suggested transformation is Y^0.3

SSE = 1.51 when lamda = .4 suggested transformation is Y^0.4

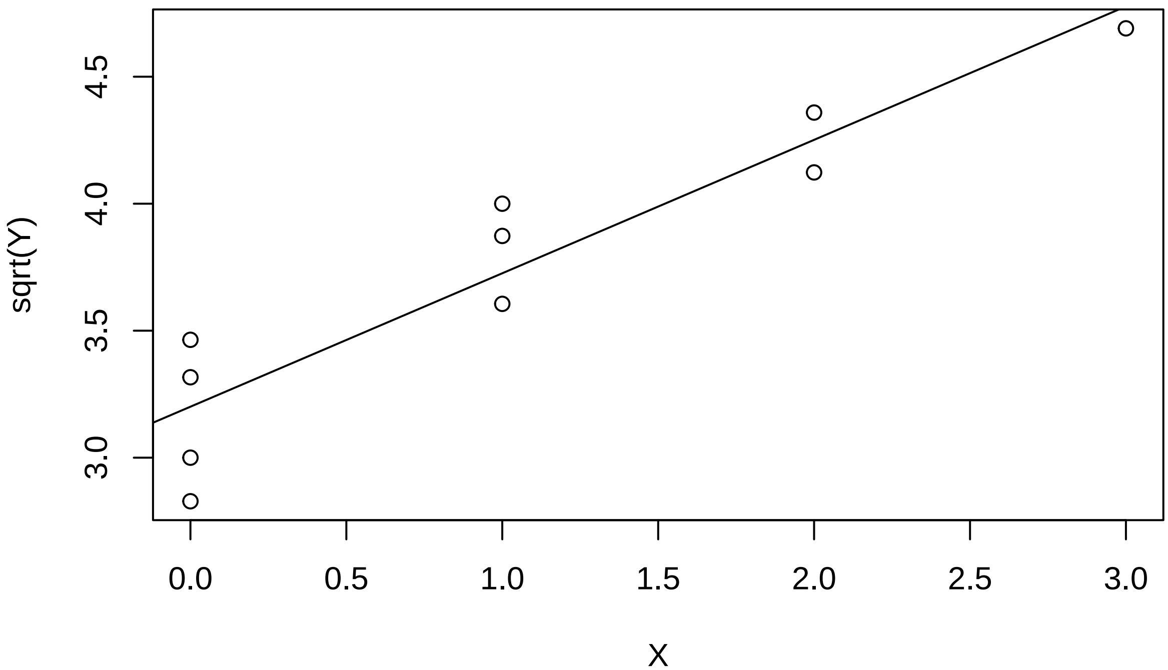
SSE = 4.20 when lamda = .5 suggested transformation is Y^0.5

```r
Y_trans <- lm(Y^0.5 ~ X)
summary(Y_trans)
```

```
##
## Call:
## lm(formula = Y^0.5 ~ X)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.37222 -0.12632  0.01059  0.13922  0.27399
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  3.20064    0.10103  31.679 1.07e-09 ***
## X            0.52537    0.07144   7.354 7.96e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.2259 on 8 degrees of freedom
## Multiple R-squared:  0.8711, Adjusted R-squared:  0.855
## F-statistic: 54.08 on 1 and 8 DF,  p-value: 7.965e-05
```

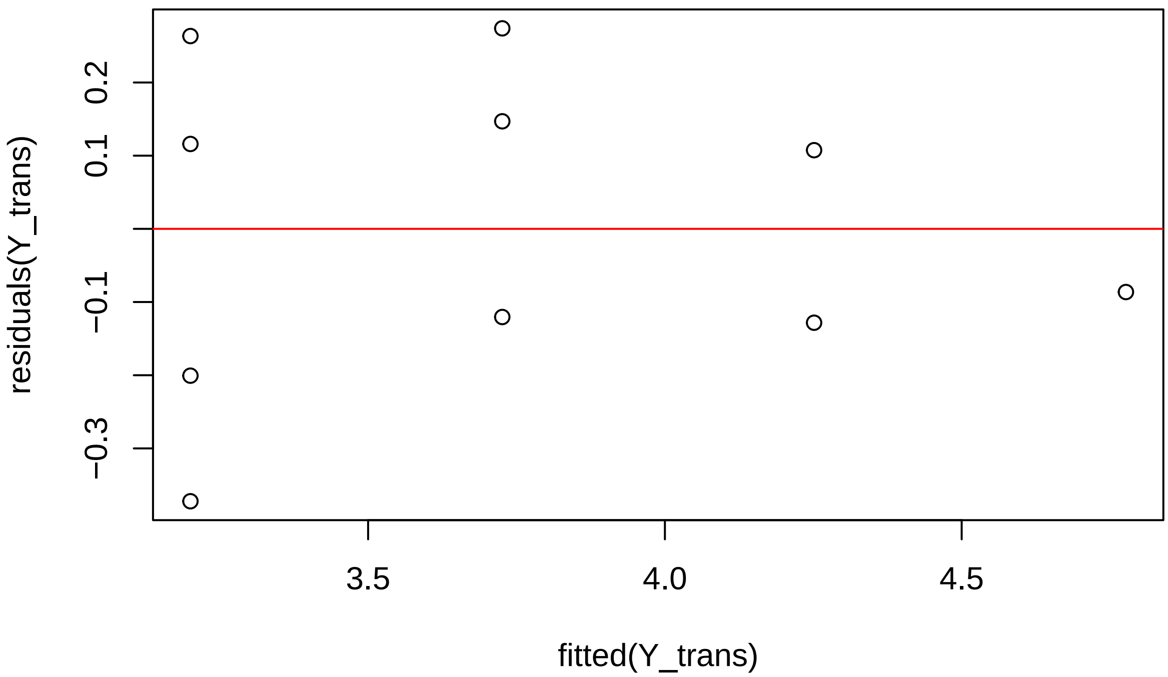
Remark: sqrt(yhat) <- 10.26093 + 1.07629X

```r
plot(X, sqrt(Y))
abline(lm(sqrt(Y)~X))
```



Residual Plot

```
plot(fitted(Y_trans), residuals(Y_trans))
abline(0,0, col = "red")
```



Initializing Matrices

```
A <- matrix(c(1, 4,
              2, 6,
              3, 8), ncol = 2, byrow = TRUE)


B <- matrix(c(1, 3,
              1, 4,
              2, 5), ncol = 2, byrow = TRUE)


C <- matrix(c(3, 8, 1,
              5, 4, 0), ncol = 3, byrow = TRUE)

D <- matrix(c(5, 3,
              15, 6), ncol = 2, byrow = TRUE)
D
```

```
##      [,1] [,2]
## [1,]    5    3
## [2,]   15    6
```

Transpose of Matrices

```r
t(B)
```

```
##      [,1] [,2] [,3]
## [1,]    1    1    2
## [2,]    3    4    5
```

Inverse of Matrices

Remark: To inverse a matrix, it has to square!

```r
solve(D)
```

```
##      [,1]       [,2]
## [1,] -0.4  0.2000000
## [2,]  1.0 -0.3333333
```

Multiplication of uneven dimensions

```r
A %*% C -> mult.res
mult.res
```

```
##      [,1] [,2] [,3]
## [1,]   23   24    1
## [2,]   36   40    2
## [3,]   49   56    3
```

Getting the dimensions of Matrices

```r
dim(mult.res)
```

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

Intercept and Slope using Matrices

```r
Y <- matrix(c(124,
              95,
              71,
              45,
              18), ncol = 1, byrow = TRUE)

X <- matrix(c(1, 49,
              1, 69,
              1, 89,
              1, 99,
              1, 109), ncol = 2, byrow = TRUE)

t(X) -> transposeX
# transposeX

dim(X)
```

```
## [1] 5 2
```

```r
transposeX%*%X -> Product2
# Product2

det(Product2)
```

```
## [1] 11600
```

```r
solve(Product2)
```

```
##             [,1]          [,2]
## [1,]  3.16939655 -0.0357758621
## [2,] -0.03577586  0.0004310345
```

```r
interceptandslope <- solve(Product2)%*%transposeX%*%Y

interceptandslope
```

```
##            [,1]
## [1,] 211.270690
## [2,]  -1.694828
```

Remark: Intercept = 211.27, and Slope = -1.6948

Fitted values using Matrices

```r
X %*% interceptandslope
```

```
##           [,1]
## [1,] 128.22414
## [2,]  94.32759
## [3,]  60.43103
## [4,]  43.48276
## [5,]  26.53448
```

Residuals using Matrices

```r
Y - X %*% interceptandslope
```

```
##            [,1]
## [1,] -4.2241379
## [2,]  0.6724138
## [3,] 10.5689655
## [4,]  1.5172414
## [5,] -8.5344828
```