Lab 2

Adam Shen

September 23, 2020

Packages

Install required packages

```
install.packages(c("EnvStats", "car", "nortest"))
```

Load packages

```
library(EnvStats)
library(car)
library(nortest)
```

Tomato data

Load the data

```
tomato <- read.table("./tomato.txt", header=TRUE)
```

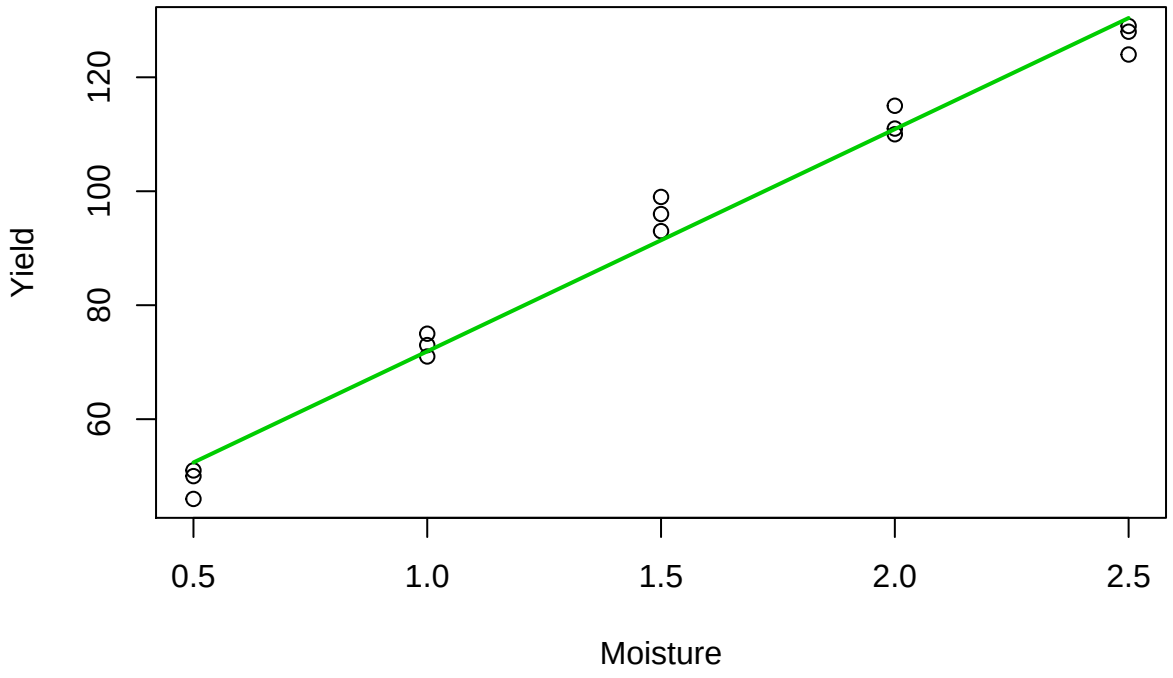
Fit a simple linear regression model

```
model <- lm(Yield ~ Moisture, data=tomato)
summary(model)
```

```
##
## Call:
## lm(formula = Yield ~ Moisture, data = tomato)
##
## Residuals:
##    Min     1Q Median     3Q    Max
##  -6.40  -1.90  -0.90   2.35   7.60
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   32.900      2.423   13.58 4.68e-09 ***
## Moisture      39.000      1.461   26.70 9.68e-13 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.001 on 13 degrees of freedom
## Multiple R-squared:  0.9821, Adjusted R-squared:  0.9807
## F-statistic: 712.6 on 1 and 13 DF,  p-value: 9.683e-13
```

Visualization

```
with(tomato, plot(x=Moisture, y=Yield, xlab="Moisture", ylab="Yield"))
with(tomato, lines(x=sort(Moisture), y=fitted(model)[order(Moisture)], col="green3", lwd=2))
```



Perform lack of fit tests

```
anova(model)
```

```
## Analysis of Variance Table
##
## Response: Yield
##           Df  Sum Sq Mean Sq F value    Pr(>F)
## Moisture   1 11407.5   11408  712.63 9.683e-13 ***
## Residuals 13   208.1      16
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

For simple linear regression, this is testing:

$$H_0 : \beta_1 = 0 \quad \text{vs} \quad H_A : \beta_1 \neq 0$$

Since the p-value is less than 0.05, we reject the null hypothesis in favour of the alternative. We conclude that the proposed model is more useful than the null model.

```
anovaPE(model)
```

```
##                     Df  Sum Sq Mean Sq   F value    Pr(>F)
## Moisture             1 11407.5 11407.5 1677.5735 1.802e-12 ***
## Lack of Fit          3   140.1    46.7    6.8676  0.008599 **
```

```
## Pure Error          10     68.0      6.8
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

From *Module 2.3*, the hypotheses are:

$$H_0 : \mu_i = \beta_0 + \beta_1 x_i \quad \text{vs} \quad H_A : \mu_i \neq \beta_0 + \beta_1 x_i$$

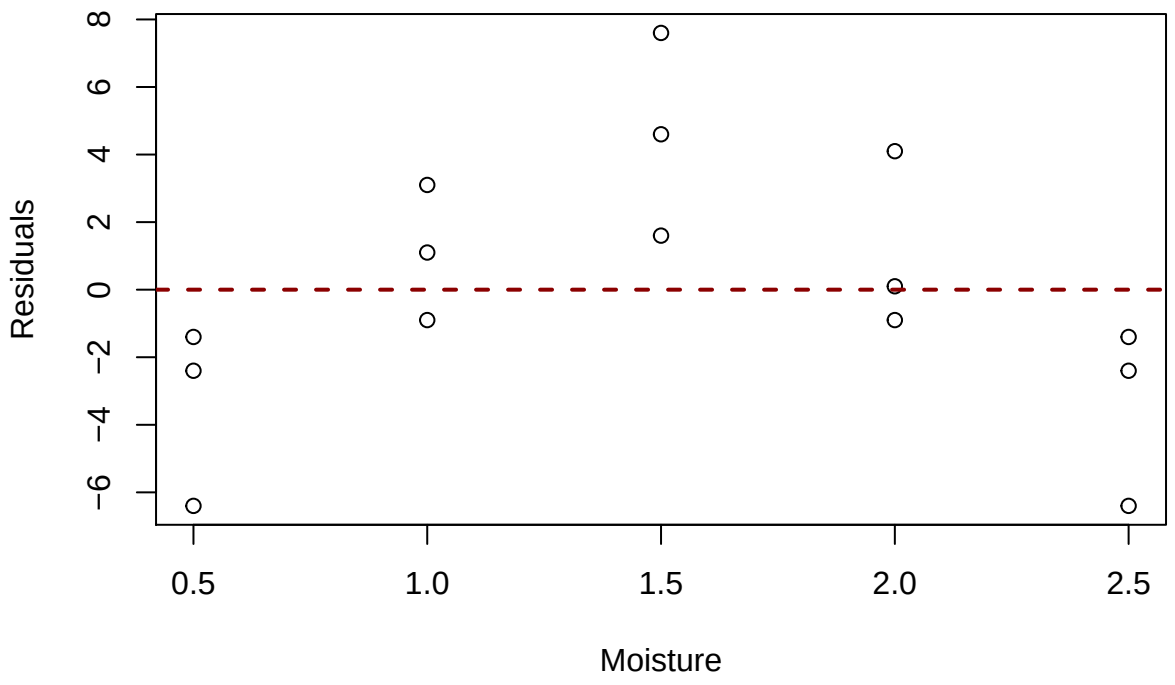
Since the p-value (0.008599) is less than 0.05, we reject the null hypothesis in favour of the alternative. In other words, we are concluding that $\mu_i \neq \beta_0 + \beta_1 x_i$ for at least one of $i = 1, 2, \ldots, 5$.

Diagnostic plots

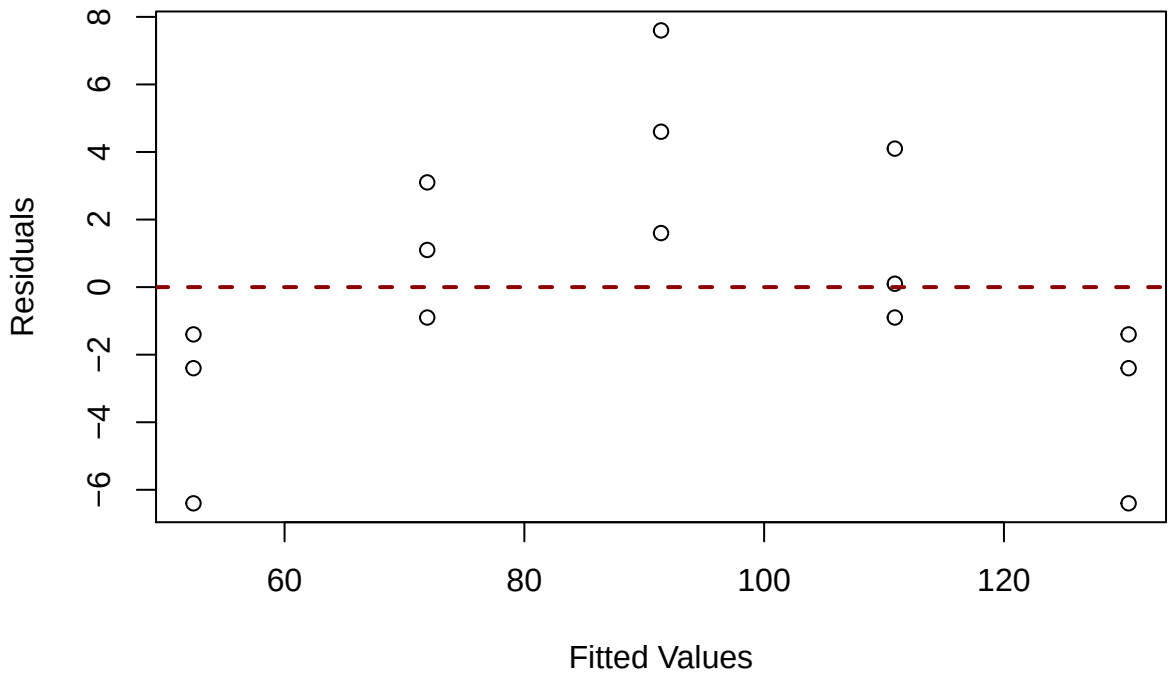
```
fits <- fitted(model)
res <- resid(model)
```

Don't need to do any sorting since we are just plotting points.

```
with(tomato, plot(x=Moisture, y=res, xlab="Moisture", ylab="Residuals"))
abline(h=0, col="darkred", lwd=2, lty=2)
```




```
plot(x=fits, y=res, xlab="Fitted Values", ylab="Residuals")
abline(h=0, col="darkred", lwd=2, lty=2)
```



These plots suggest a non-linear functional form.

Tests for non-constant variance

Modified Levene test

```
(fitsize <- factor(fits <= 100))
```

```
##     1     2     3     4     5     6     7     8     9    10    11    12    13
##  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE FALSE FALSE FALSE FALSE
##    14    15
## FALSE FALSE
## Levels: FALSE TRUE
```

```
leveneTest(res, group=fitsize)
```

```
## Levene's Test for Homogeneity of Variance (center = median)
##       Df F value Pr(>F)
## group  1  0.4427 0.5175
##       13
```

From *Module 2.5*, our hypotheses are:

$$H_0 : \text{The error term variance is constant}$$

$$H_A : \text{The error term variance is non-constant}$$

In addition, it is recommended that we test this using $\alpha = 0.10$ as this allows the test to be more powerful for detecting departures from a constant error term variance.

Since the p-value is greater than 0.10, we fail to reject the null hypothesis. In other words, there is insufficient evidence to conclude that the error term variance is non-constant.

Breusch-Pagan test

```
ncvTest(model)
```

```
## Non-constant Variance Score Test
## Variance formula: ~ fitted.values
## Chisquare = 0.003117379, Df = 1, p = 0.95547
```

From *Module 2.5*, our hypotheses are:

$$H_0 : \text{The error term variance is constant}$$

$$H_A : \text{The error term variance is non-constant}$$

It is once again recommended to use $\alpha = 0.10$. Since the p-value is greater than 0.10, we fail to reject the null hypothesis. In other words, there is insufficient evidence to conclude that the error term variance is non-constant.

The nitty-gritty of the Breusch-Pagan test

The Breusch-Pagan test works by fitting a new model of the squared residuals against the fitted values.

```
res_sq <- res^2
bp_model <- lm(res_sq ~ fits)
anova(bp_model)
```

```
## Analysis of Variance Table
##
## Response: res_sq
##           Df Sum Sq Mean Sq F value Pr(>F)
## fits       1    1.2    1.20  0.0033  0.955
## Residuals 13 4709.5  362.27
```

The test statistic is:

$$\chi_0^2 = \frac{SSR^*}{2(SSE/n)^2}$$

where:

- SSR^* is the sum of squares regression from the new model
- SSE is the sum of squares error from the original model
- n is the number of observations

We compute the test statistic as:

```
test_stat <- anova(bp_model)["fits", "Sum Sq"] /
  (2*(anova(model)["Residuals", "Sum Sq"]/nrow(tomato))^2)

all.equal(ncvTest(model)$ChiSquare, test_stat)
```

```
## [1] TRUE
```

Residual analysis

Numerical summaries

```
summary(res)
```

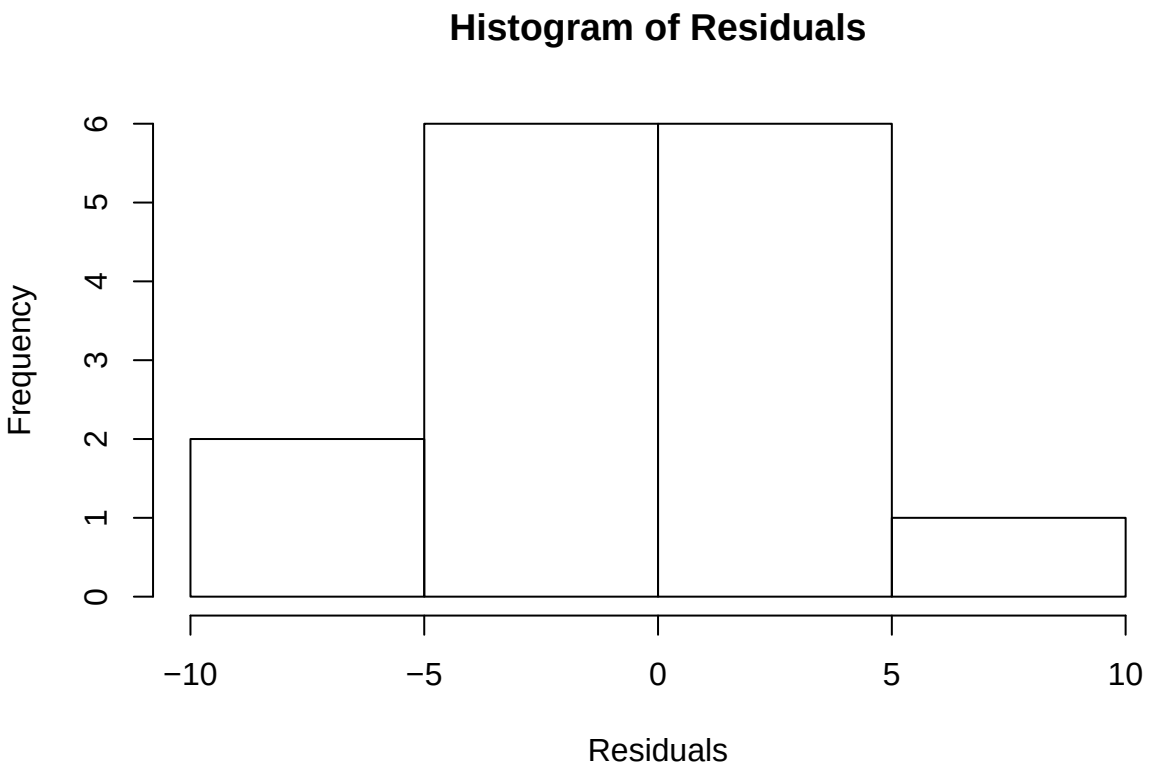
```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   -6.40   -1.90   -0.90    0.00    2.35    7.60
```

Graphical summaries

```
stem(res)
```

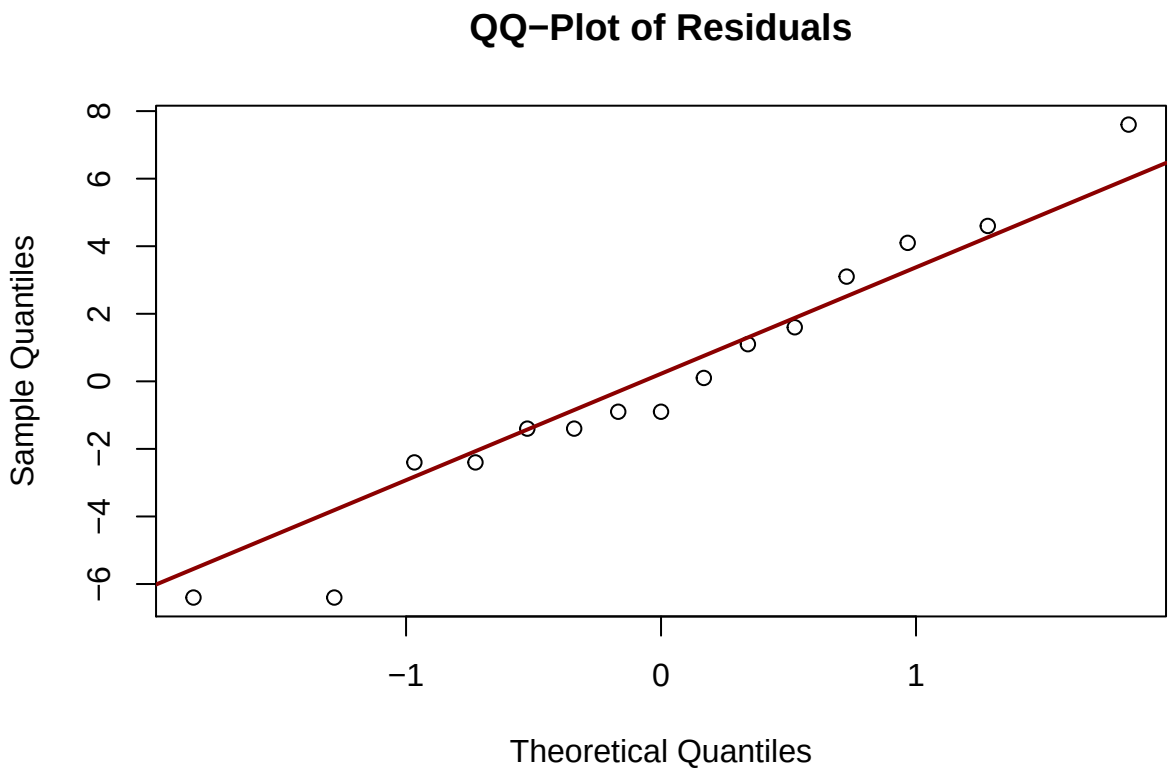
```
##
##   The decimal point is 1 digit(s) to the right of the |
##
##   -0 | 66
##   -0 | 221111
##    0 | 01234
##    0 | 58
```

```
hist(res, main="Histogram of Residuals", xlab="Residuals")
```



Based on the stem and leaf plot and the histogram, the data seems to have a mound-shape.

```r
qqnorm(res, main="QQ-Plot of Residuals")
qqline(res, col="darkred", lwd=2)
```



The fit on the QQ-line is satisfactory.

Hypothesis tests

For the following tests, the hypotheses are:

$$H_0 : \text{The data are normally distributed}$$

$$H_A : \text{The data are not normally distributed}$$

```r
ad.test(res)
```

```
##
##  Anderson-Darling normality test
##
## data:  res
## A = 0.25546, p-value = 0.6751
```

```r
shapiro.test(res)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  res
## W = 0.96508, p-value = 0.7797
```

```r
lillie.test(res)
```

```
##
##  Lilliefors (Kolmogorov-Smirnov) normality test
##
## data:  res
## D = 0.13347, p-value = 0.6701
```

Each test results in a large p-value. For each test, we fail to reject the null hypothesis. We conclude that there is insufficient evidence of non-normality in the errors.

Tomato data - Round 2

We will re-run all the above code, but this time applying a square root transformation to the `Moisture` variable. There are three ways of going about this. We can either:

1. Take the square root of `Moisture` and store it as an external variable
2. Create a new column in the `tomato` data set
3. Apply the transformation in the model specification (risky)

Method 1: Create an external variable

```
sqrt_moisture <- sqrt(tomato$Moisture)
sqrt_model <- lm(Yield ~ sqrt_moisture, data=tomato)
```

Method 2: Create new column in `tomato` data

I will be using this method as I would like all variables related to the data to be contained within the data set.

```
tomato <- transform(tomato, sqrt_moisture=sqrt(Moisture))
sqrt_model <- lm(Yield ~ sqrt_moisture, data=tomato)
```

Method 3: Apply the transformation in the model specification (risky)

```
sqrt_model <- lm(Yield ~ sqrt(Moisture), data=tomato)
```

This method is risky because of how formulas are specified in R. In a formula, the signs `+`, `-`, `*`, `^`, are formula operators rather than mathematical operators. For example, if we are interested in using the square of a predictor variable, we cannot use `y ~ x^2`. Instead, we must use `y ~ I(x^2)` which tells R to interpret the `^` as a mathematical operator rather than a formula operator. For more information, see the Details section of `?formula`. Transformations that do not use the above operators, such as `sqrt()` and `log()`, are safe to use without `I()`.

Summary of model

```
summary(sqrt_model)
```

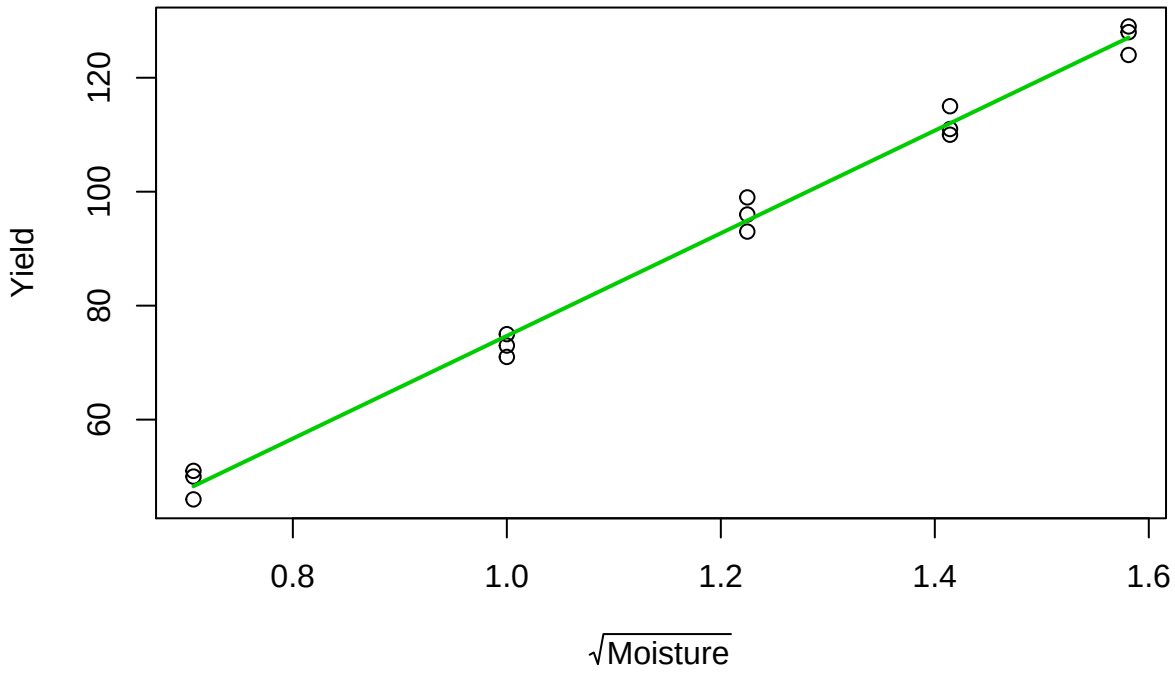
```
##
## Call:
## lm(formula = Yield ~ sqrt_moisture, data = tomato)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.6926 -1.9763  0.3074  1.8226  4.0589
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)    -15.403      2.571   -5.99 4.52e-05 ***
## sqrt_moisture   90.096      2.099   42.91 2.16e-15 ***
```

```
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.503 on 13 degrees of freedom
## Multiple R-squared:  0.993,  Adjusted R-squared:  0.9925
## F-statistic:  1842 on 1 and 13 DF,  p-value: 2.162e-15
```

Visualization

Since the original data was already sorted by `Moisture` and the square root preserves order, it is not actually necessary to sort before plotting.

```
with(tomato, plot(x=sqrt_moisture, y=Yield, xlab=expression(sqrt("Moisture")), ylab="Yield"))
with(tomato, lines(x=sort(sqrt_moisture), y=fitted(sqrt_model)[order(sqrt_moisture)],
                   col="green3", lwd=2))
```



Perform lack of fit tests

```
anova(sqrt_model)
```

```
## Analysis of Variance Table
##
## Response: Yield
##               Df  Sum Sq Mean Sq F value    Pr(>F)
## sqrt_moisture  1 11534.2 11534.2  1841.6 2.162e-15 ***
## Residuals     13    81.4     6.3
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

For simple linear regression, this is testing:

$$H_0 : \beta_1 = 0 \quad \text{vs} \quad H_A : \beta_1 \neq 0$$

Since the p-value is less than 0.05, we reject the null hypothesis in favour of the alternative. Once again, we conclude that our proposed model is more useful than the null model.

```
anovaPE(sqrt_model)
```

```
##                          Df  Sum Sq Mean Sq   F value    Pr(>F)
## sqrt_moisture             1 11534.2 11534.2 1696.2031 1.706e-12 ***
## Lack of Fit               3    13.4     4.5    0.6578    0.5964
## Pure Error               10    68.0     6.8
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

From *Module 2.3*, the hypotheses are:

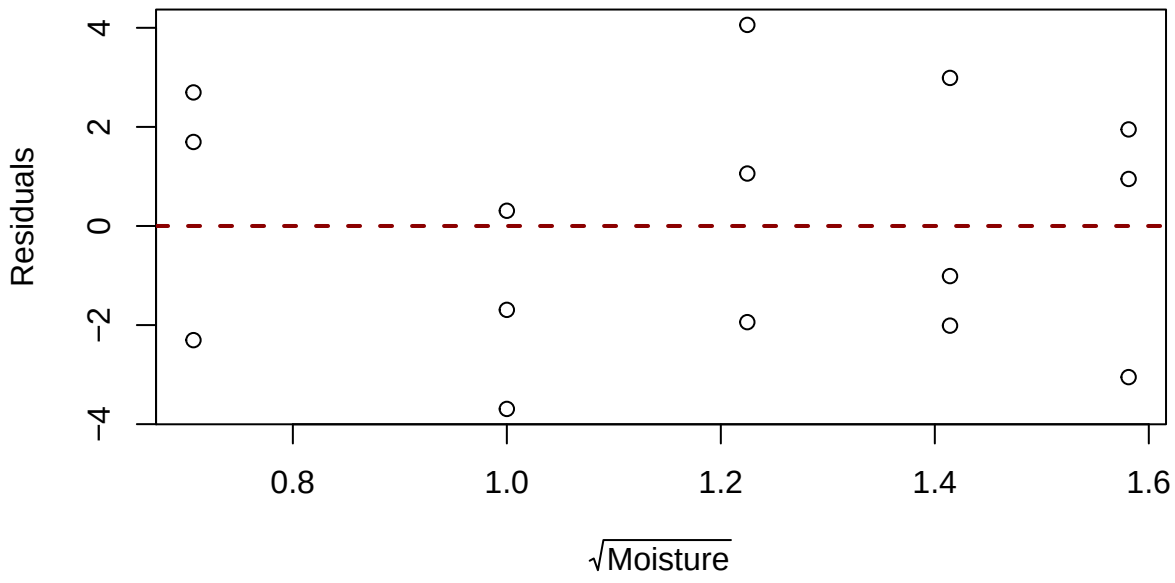
$$H_0 : \mu_i = \beta_0 + \beta_1 x_i \quad \text{vs} \quad H_A : \mu_i \neq \beta_0 + \beta_1 x_i$$

For this model, the p-value (0.5964) is greater than 0.05 so we fail to reject the null hypothesis. In other words, there is insufficient evidence that our model is inadequate.

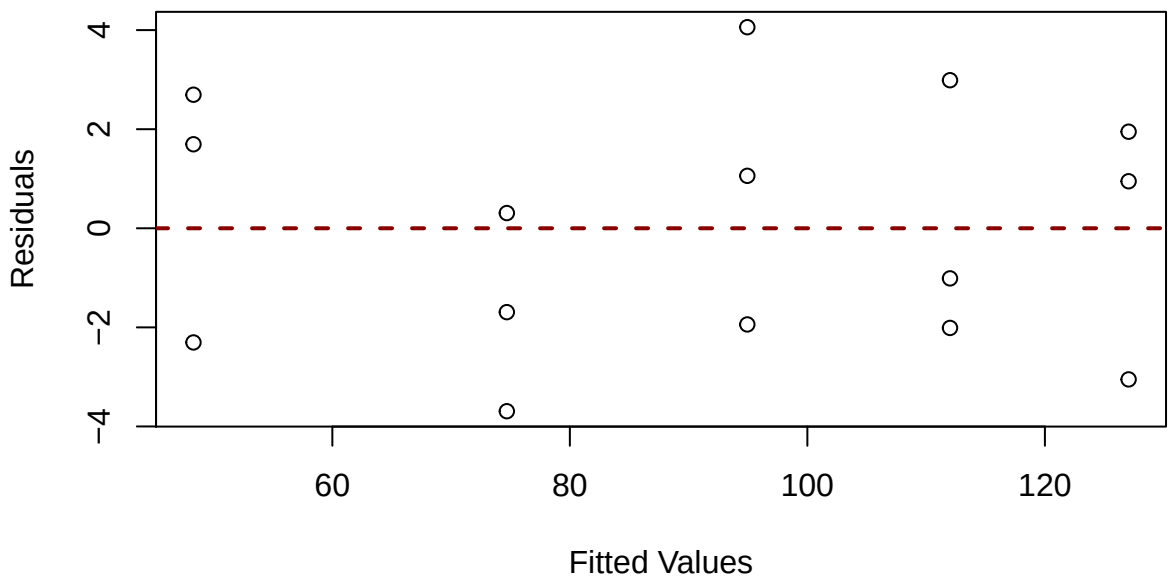
Diagnostic plots

```
# I'm going to overwrite the `fits` and `res` variables from the previous model
fits <- fitted(sqrt_model)
res <- resid(sqrt_model)
```

```
with(tomato, plot(x=sqrt_moisture, y=res, xlab=expression(sqrt("Moisture")), ylab="Residuals"))
abline(h=0, col="darkred", lwd=2, lty=2)
```




```
plot(x=fits, y=res, xlab="Fitted Values", ylab="Residuals")
abline(h=0, col="darkred", lwd=2, lty=2)
```



This is a large improvement compared to the last model. The residuals appear to be contained within a horizontal band centred at zero and there do not appear to be any noticeable trends.

Tests for non-constant variance

Modified Levene test

```
fitsize <- factor(fits <= 100)
leveneTest(res, group=fitsize)
```

```
## Levene's Test for Homogeneity of Variance (center = median)
##       Df F value Pr(>F)
## group  1  0.0472 0.8314
##       13
```

Our hypotheses are once again:

$$H_0 : \text{The error term variance is constant}$$

$$H_A : \text{The error term variance is non-constant}$$

Using $\alpha = 0.10$ again as recommended, since the p-value of this test is greater than 0.10, we fail to reject the null hypothesis. In other words, there is insufficient evidence that the error term variance is non-constant.

Breusch-Pagan test

```
ncvTest(sqrt_model)
```

```
## Non-constant Variance Score Test
## Variance formula: ~ fitted.values
## Chisquare = 0.009319361, Df = 1, p = 0.92309
```

Our hypotheses are once again:

$$H_0 : \text{The error term variance is constant}$$

$$H_A : \text{The error term variance is non-constant}$$

Using $\alpha = 0.10$ again as recommended, since the p-value of this test is greater than 0.10, we fail to reject the null hypothesis. In other words, there is insufficient evidence that the error term variance is non-constant.

We will skip the nitty-gritty details of the test for this model as it was already demonstrated with the previous model.

Residual analysis

Numerical summaries

```
summary(res)
```

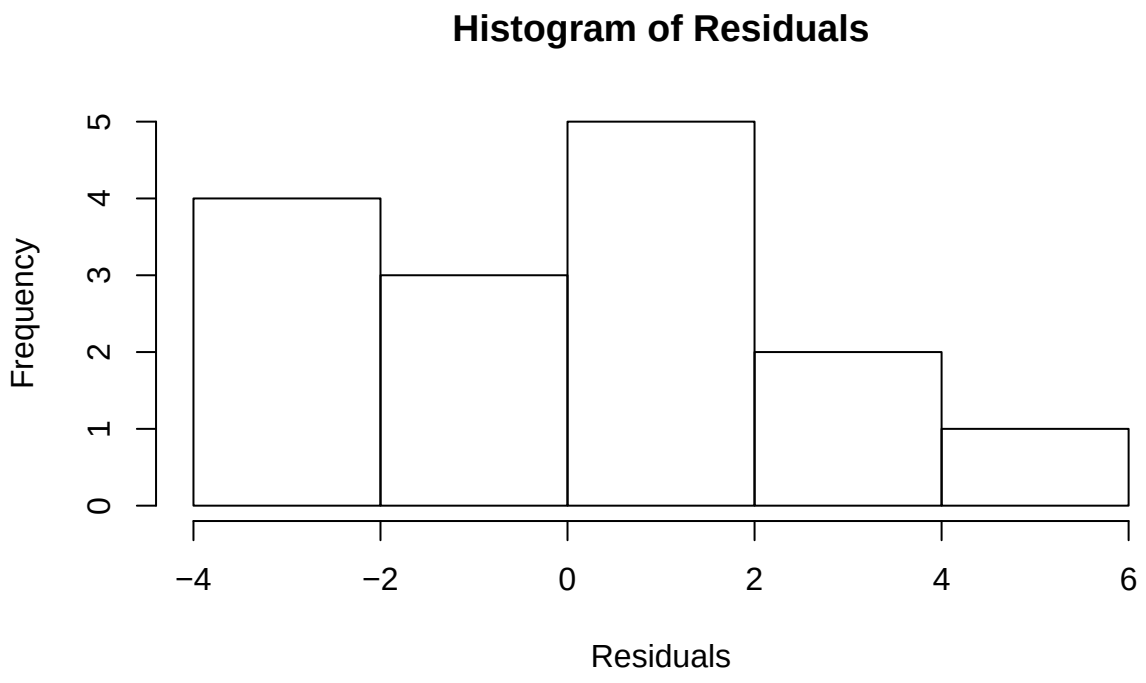
```
##     Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## -3.6926 -1.9763  0.3074  0.0000  1.8226  4.0589
```

Graphical summaries

```
stem(res)
```

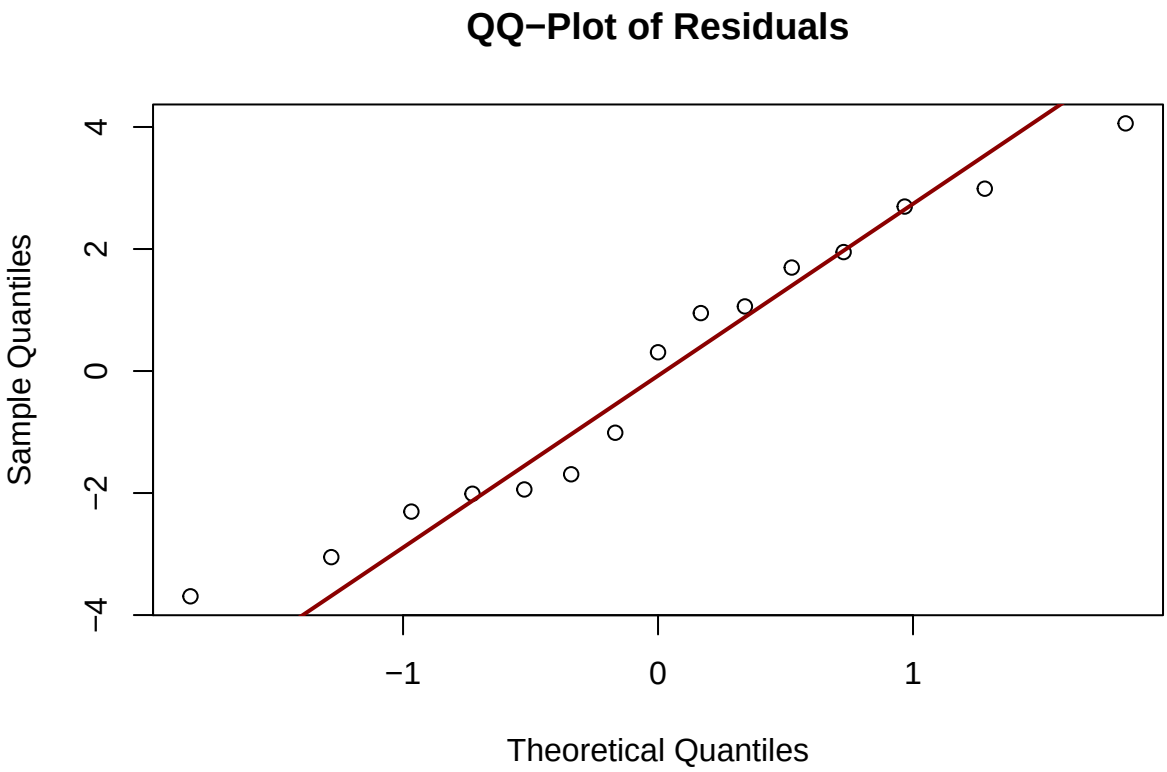
```
##
##   The decimal point is at the |
##
##   -2 | 7130
##   -0 | 970
##    0 | 39179
##    2 | 70
##    4 | 1
```

```
hist(res, main="Histogram of Residuals", xlab="Residuals")
```



The residuals don't appear to have a mound shape anymore.

```
qqnorm(res, main="QQ-Plot of Residuals")
qqline(res, col="darkred", lwd=2)
```



The fit on the QQ-plot seems okay…

Hypothesis tests

For the following tests, the hypotheses are:

$$H_0 : \text{The data are normally distributed}$$

$$H_A : \text{The data are not normally distributed}$$

```
ad.test(res)
```

```
##
##  Anderson-Darling normality test
##
## data:  res
## A = 0.32522, p-value = 0.4861
```

```
shapiro.test(res)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  res
## W = 0.95109, p-value = 0.5418
```

```
lillie.test(res)
```

```
##
##  Lilliefors (Kolmogorov-Smirnov) normality test
##
```

```
## data:  res
## D = 0.15862, p-value = 0.3932
```

Each test still results in somewhat large p-values. For each test, we fail to reject the null hypothesis. We conclude that there is insufficient evidence of non-normality in the errors.

Intervals

```
confint(sqrt_model)
```

```
##                   2.5 %    97.5 %
## (Intercept)   -20.95793 -9.848172
## sqrt_moisture  85.56010 94.631181
```

```
predict(sqrt_model, interval="confidence", se.fit=TRUE, newdata=data.frame(sqrt_moisture=sqrt(2)))
```

```
## $fit
##        fit      lwr      upr
## 1 112.0114 110.2721 113.7508
##
## $se.fit
## [1] 0.8051166
##
## $df
## [1] 13
##
## $residual.scale
## [1] 2.502596
##
## $n.coefs
## [1] 2
```

```
predict(sqrt_model, interval="prediction", se.fit=TRUE, newdata=data.frame(sqrt_moisture=sqrt(2)))
```

```
## $fit
##        fit     lwr      upr
## 1 112.0114 106.332 117.6909
##
## $se.fit
## [1] 0.8051166
##
## $df
## [1] 13
##
## $residual.scale
## [1] 2.502596
##
## $n.coefs
## [1] 2
```

The equation of our regression line is:

$$\widehat{\text{Yield}} = -15.403 + 90.096\sqrt{\text{Moisture}}$$

This means that for each unit increase in $\sqrt{\text{Moisture}}$, $\widehat{\text{Yield}}$ will increase by 90.096 units. With this in mind, do we need to perform any transformations to the above intervals before interpreting them?

Now suppose instead, we had a regression model of the form:

$$\ln\left(\widehat{\text{Yield}}\right) = \hat{\beta}_0 + \hat{\beta}_1\text{Moisture}$$

This means that for each unit increase in Moisture, $\ln(\widehat{\text{Yield}})$ will increase by $\hat{\beta}_1$ units. If we were to construct intervals with this model using a similar procedure as above, would these intervals require transformations before we interpret them?

Don't forget to try the exercise at the end of the Lab 2 Instructions using the `income` data.

Small note on subsetting

You'll notice that the intervals you get either from `confint()` or `predict()` are of class `matrix`. This means that if you were to subset them, you would need to specify the row and column, rather than a single index. Subsetting will be useful if you need to transform your intervals.

```
(tmp <- predict(sqrt_model, interval="prediction", newdata=data.frame(sqrt_moisture=sqrt(2))))
```

```
##        fit     lwr      upr
## 1 112.0114 106.332 117.6909
```

```
class(tmp)
```

```
## [1] "matrix"
```

```
# Extract row 1, and columns 2 and 3
# Note that `c()` is used to create the vector with elements `2` and `3`
tmp[1, c(2,3)]
```

```
##      lwr      upr
## 106.3320 117.6909
```

```
# Extract row 1, and all columns EXCEPT column 1
tmp[1, -1]
```

```
##      lwr      upr
## 106.3320 117.6909
```

```
# Exponentiate each element in the vector (only if required)
exp(tmp[1, -1])
```

```
##          lwr          upr
## 1.511476e+46 1.295654e+51
```