R Companion - Clewer & Scarisbrick (2001)

R Companion material for *Clewer, Alan G., and David H. Scarisbrick. Practical statistics and experimental design for plant and crop science. John Wiley & Sons, 2001.*

Rodrigo R Amadeu
rramadeu@ufl.edu

Contents

Chapter 2 - Basic Statistical Calculations **2**

Chapter 3 - Basic Data Summary **4**

Chapter 4: The Normal Distribution, the t-Distribution and Confidence Intervals **12**

Chapter 5: Introduction to Hypothesis Testing **24**

Chapter 6: Introduction to Hypothesis Testing **26**

Chapter 7: Linear Regression and Correlation **28**

Chapter 8: Curve Fitting **35**

Chapter 9: The Completely Randomized Design **39**

Chapter 10: The Randomized Block Design **57**

Necessary R codes to perform the analysis of the Clewer & Scarisbrick (2001). For interpretation and theoretical explanation you should go to the book.

Chapter 2 - Basic Statistical Calculations

Data input in R

Data in R are stored in vectors, to build a vector 'x' with the elements from the Example 2.1 in the book (i.e., 14.8, 15.2, 17.4, 11.6, and 12.5), you can use the function `c()`, where `c` stands for combine values into a vector or list. An example:

```
x = c(14.8, 15.2, 17.4, 11.6, 12.5)
```

To visualize the built data, in the RStudio you can look at the `Enviroment` tab (upper right) which shows you all the loaded data into your `R` environment. Another option is to explicitly call it or print it:

```
x
```

```
## [1] 14.8 15.2 17.4 11.6 12.5
```

```
print(x)
```

```
## [1] 14.8 15.2 17.4 11.6 12.5
```

Another way to input the data in R is to import from a file. In the RStudio, you can go to `Import Dataset` option under the `Environment` tab and select you data type. Each data type requires a specific command with specific arguments.

Basic Statistical Calculations

Mean

There are several ways to compute the same value in 'R', it follows some of the ways to compute the sample mean:

You can use R as a calculator and explicitly compute the mean, the signals follow the standard of other statistical softwares:

```
(14.8 + 15.2 + 17.4 + 11.6 + 12.5)/5
```

```
## [1] 14.3
```

Or you can use built-in function on your vector `x`, as:

```
sum(x)/length(x) #sum all the elements of 'x' and divide such value for its length (n)
```

```
## [1] 14.3
```

```
mean(x) #compute the mean of x
```

```
## [1] 14.3
```

Hereafter, we show just compact ways to compute different statistics:

```
median(x) #median of x
```

```
## [1] 14.8
```

```
var(x) #sample variance of x
```

```
## [1] 5.3
```

```
var(x)*(length(x)-1) #corrected sum of squares of x
```

```
## [1] 21.2
```

```r
sd(x) #sample standard deviation of x
```

```
## [1] 2.302173
```

```r
sd(x)/mean(x) #coefficient of variation of x
```

```
## [1] 0.1609911
```

There is no built-in function to directly compute the coefficient of variation (CV) in R. Loading an R package is a way to have more functions and dataset to work with. There are thousands of available R packages in different repositories for different objectives. Here we show how to install and load the 'sjstats' packages which has a built-in function to compute the CV.

```r
install.packages('sjstats') #install sjstats packages
library(sjstats) #loading the package
```

You can also install a package using RStudio interface, go to the Tools -> Install Packages and use the search.

Now we have the function `cv()` available in our R enviroment. To look at its details and to use it:

```r
?cv
cv(x)
```

```
## [1] 0.1609911
```

Weighted mean

```r
x = c(3.5, 4.8, 5.2) #sample means
n = c(4, 5, 10) #sample sizes
?weighted.mean
weighted.mean(x,n)
```

```
## [1] 4.736842
```

Chapter 3 - Basic Data Summary

3.2: Frequency distributions (discrete data)

Example 3.1: Data input number of tillers

```
x = c(1, 1, 1, 2, 2, 2, 2,
      3, 3, 3, 3, 3, 3, 3,
      3, 4, 4, 4, 4, 4, 4,
      4, 4, 4, 4, 4, 4, 4,
      4, 4, 4, 5, 5, 5, 5,
      5, 5, 5, 5, 5, 5, 5,
      5, 5, 6, 6, 6, 6, 6,
      6, 6, 6, 6, 7, 7, 7,
      7, 7, 8, 8)
```

Basic data summary

```
sum(x)
```

```
## [1] 269
```

```
length(x)
```

```
## [1] 60
```

```
mean(x)
```

```
## [1] 4.483333
```

```
var(x)
```

```
## [1] 2.762429
```

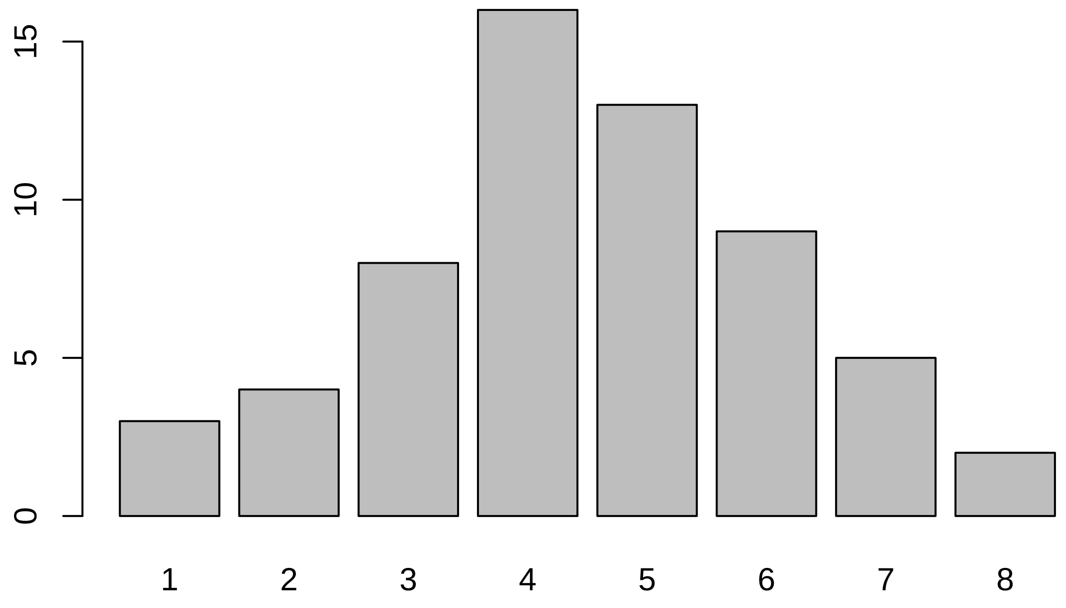
Frequency table

```
table(x)
```

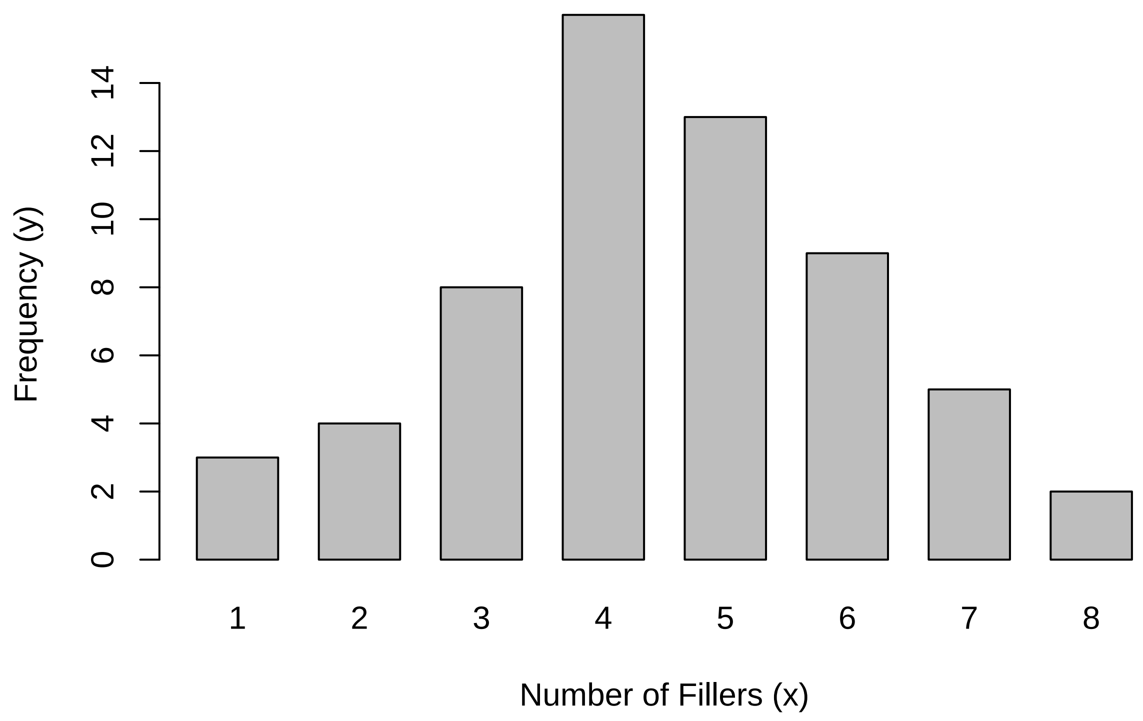
```
## x
##  1  2  3  4  5  6  7  8
##  3  4  8 16 13  9  5  2
```

Bar plot

```
counts <- table(x)
barplot(counts)
```



```r
# Customizing it a little bit
barplot(counts, xlab = "Number of Fillers (x)", ylab = "Frequency (y)", space = 0.5, ylim=c(0,15))
```



The mode

```
sort(counts) #the mode is 4 with 16 counts
```

```
## x
##  8  1  2  7  3  6  5  4
##  2  3  4  5  8  9 13 16
```

```
median(x)
```

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

3.3 Frequency distributions (continuos data)

Example 3.2: Data input barley yield

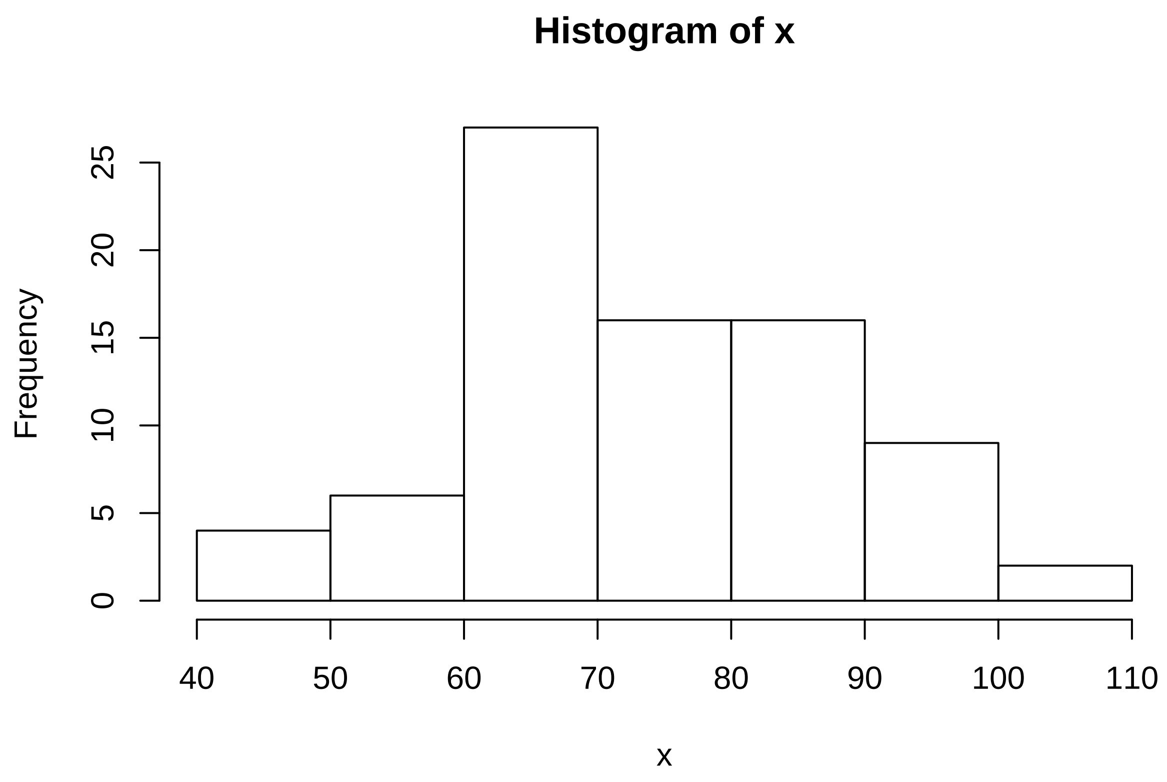
```
x = c(95, 70, 68, 88, 79, 92, 64, 83, 67, 63,
      56, 70, 53, 78, 71, 62, 42, 80, 50, 68,
      78, 104, 62, 66, 90, 86, 66, 82, 83, 56,
      82, 90, 71, 77, 93, 68, 91, 98, 79, 75,
      92, 93, 73, 79, 95, 78, 77, 108, 86, 87,
      68, 68, 49, 75, 82, 61, 68, 65, 56, 96,
      52, 61, 87, 79, 64, 64, 84, 63, 64, 44,
      87, 63, 65, 64, 81, 72, 62, 58, 84, 67)
```

```r
length(x) #checking dimension
```

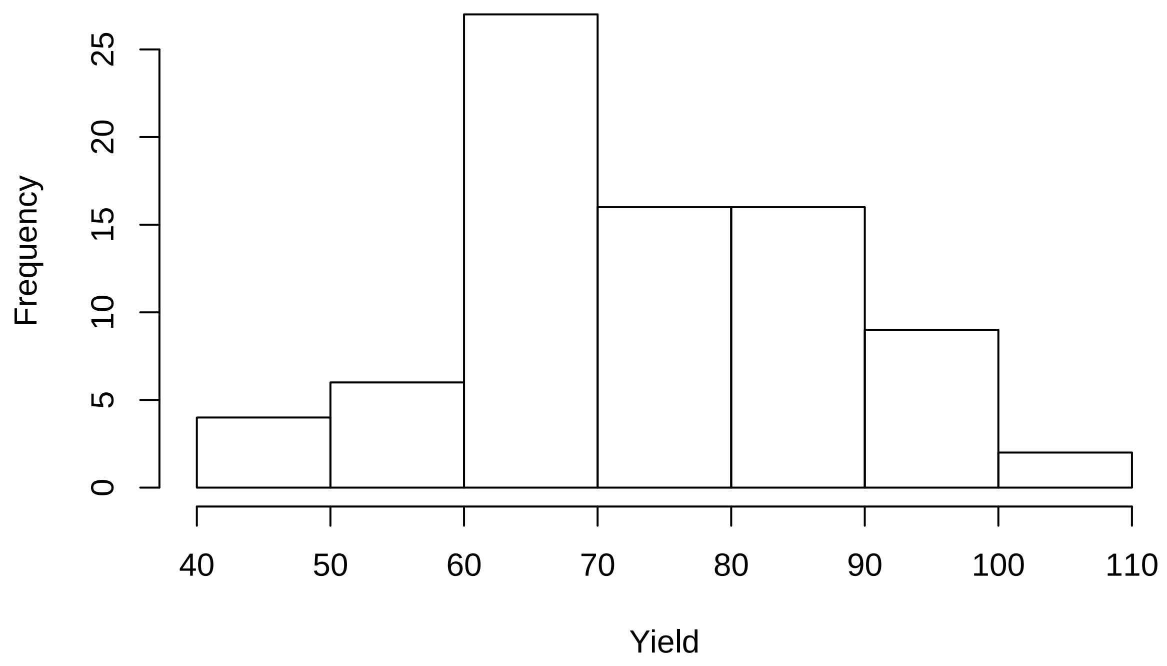
```
## [1] 80
```

The histogram

```r
hist(x)
```



```r
# Customizing it a little bit
hist(x, xlab="Yield", main=NULL)
```



Quartiles and Ranges

```r
quantile(x,0.25) #Q1
```

```
## 25%
##  64
```

```r
quantile(x,0.75) #Q3
```

```
## 75%
##  84
```

```r
IQR(x)  #IQR (i.e., Q3-Q1)
```

```
## [1] 20
```

```r
max(x)
```

```
## [1] 108
```

```r
min(x)
```

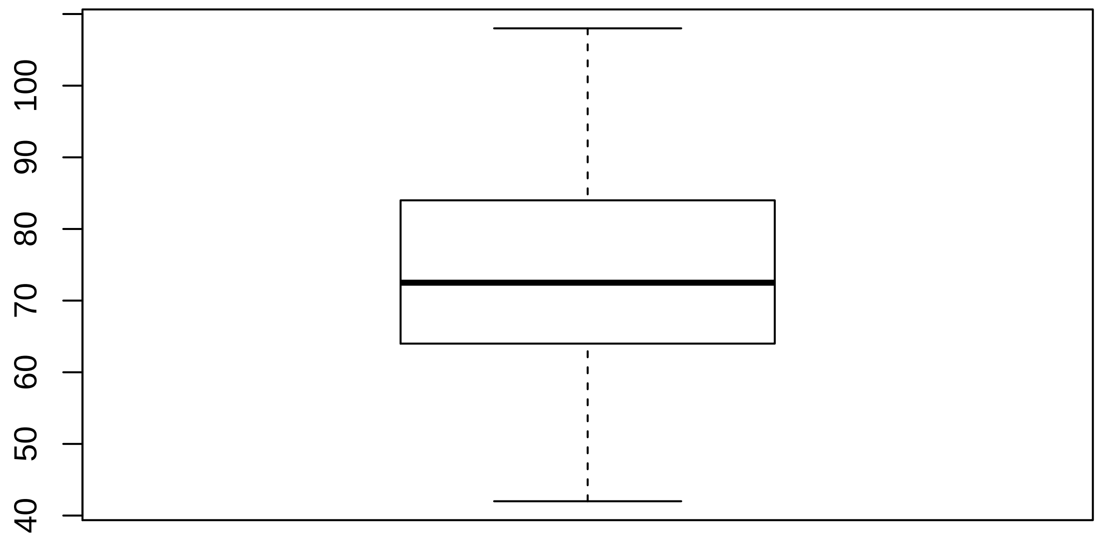
```
## [1] 42
```

```r
range(x)
```

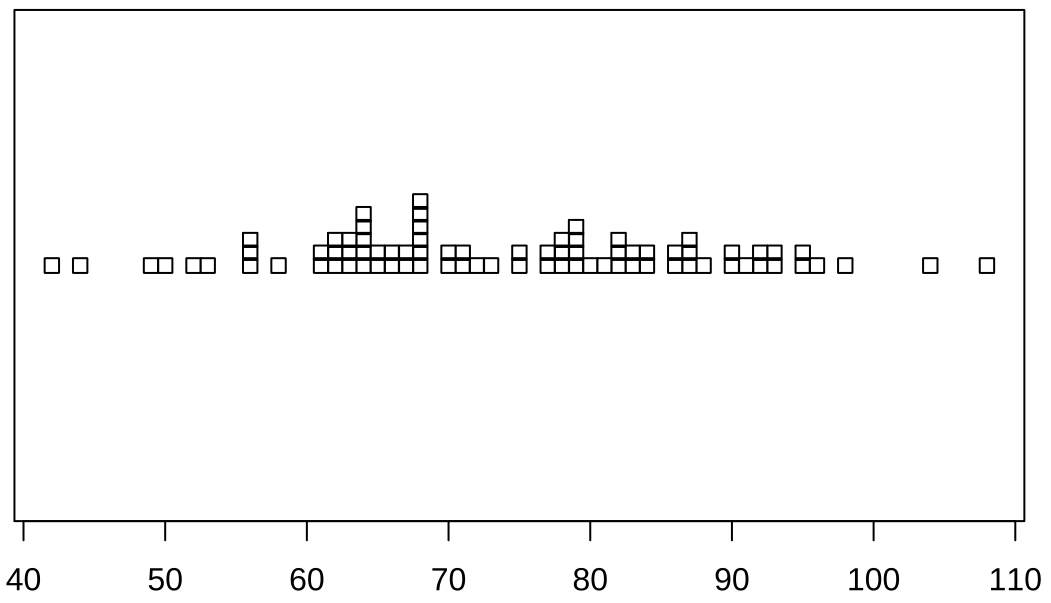
```
## [1]  42 108
```

Other graphical methods

```r
boxplot(x) #boxplot
```



```r
stripchart(x,method = "stack") #dotplot
```



```
stem(x) #stem and leaf plot
```

```
##
##   The decimal point is 1 digit(s) to the right of the |
##
##    4 | 249
##    5 | 0236668
##    6 | 11222333444445566677888888
##    7 | 00112355778889999
##    8 | 012223344667778
##    9 | 00122335568
##   10 | 48
```

Descriptive statistics

The R default summary:

```
## R Default:
length(x) #N
```

```
## [1] 80
```

```
summary(x)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   42.00   64.00   72.50   73.96   84.00  108.00
```

```r
sd(x) #standard deviation
```

```
## [1] 14.02389
```

```r
var(x) #variance
```

```
## [1] 196.6695
```

```r
mean(x) #mean
```

```
## [1] 73.9625
```

```r
mean(x, trim=0.05) #trimmed mean by 0.05
```

```
## [1] 73.97222
```

There are several packages that provide different types of summary statistics, to see some of them click here.

Chapter 4: The Normal Distribution, the t-Distribution and Confidence Intervals

Simulating data in R

To simulate data in R, you need to choose a distribution (probability density function),its parameters, and how many data points (`n`) to sample from this distribution.

Here you can find several probability density functions that `R` can handle. The core idea on simulating data from such distributions is to know what each one of the simulation functions does (functions `r...`, `p...`, `q...`,and `d...`). Knowing this, you can simulate or extract data from different probability density function. Here we present the four functions of the normal (Gaussian) distribution:

Distribution Family functions

`rnorm`

`r`andomly generated `n` numbers based on a Normal Distribution with mean `mean` and standard deviation `sd`:

```
rnorm(n = 10, mean = 0, sd = 1) #"r": random, randomly generated numbers f
```

```
##  [1]  0.007148746 -0.175584725  0.773127908 -0.756723486 -0.360001071
##  [6]  0.968107050 -2.716321732  0.567764901  0.741498870 -0.592522684
```

`pnorm`

What is the `p`robability to have values lower than `q` based on a Normal Distribution with mean `mean` and standard deviation `sd`:

```
pnorm(q = 0, mean = 0, sd = 1) #"p": probability, cumulative density function
```

```
## [1] 0.5
```

`qnorm` (inverse of `pnorm`)

What is the value (`q`) correspondent to the probability `p` to have values lower based on a Normal Distribution with mean `mean` and standard deviation `sd`:

```
qnorm(p = 0.5, mean = 0, sd = 1) #"q": quantiles, cumulative density function (quantiles)
```

```
## [1] 0
```

`dnorm`

What is the density of correspondent to the point `x` based on a Normal Distribution with mean `mean` and standard deviation `sd`.

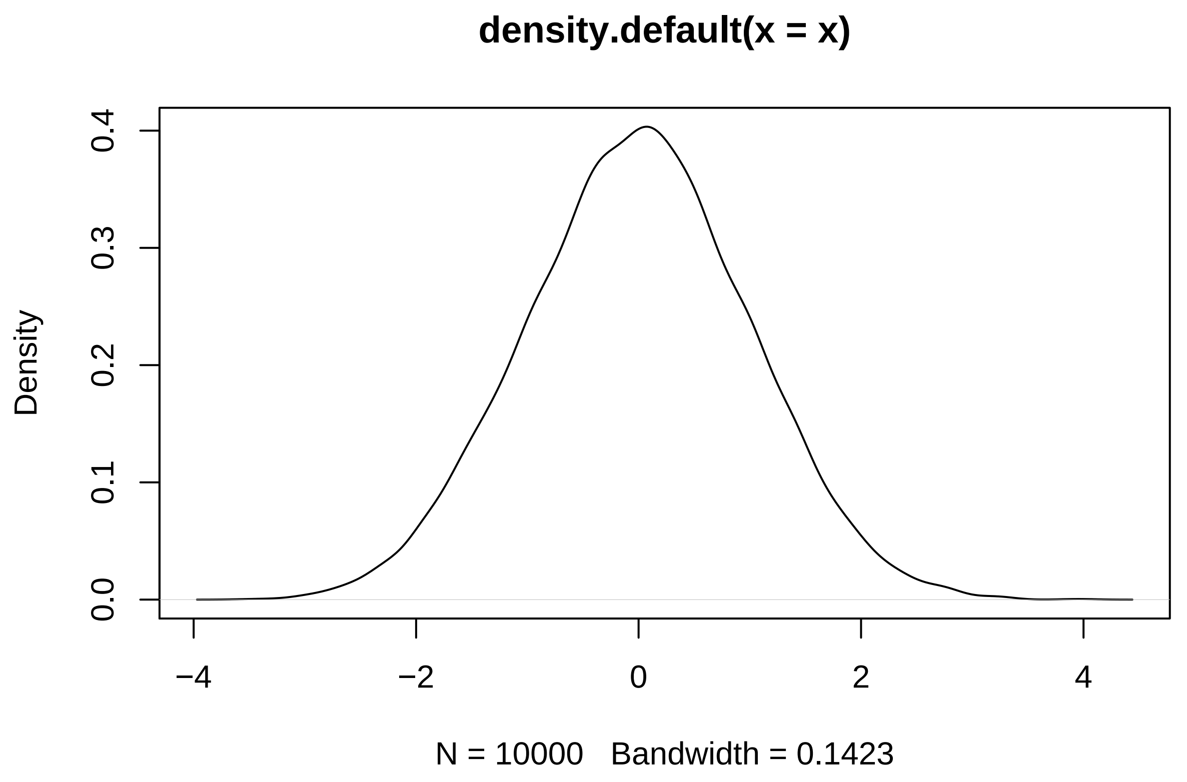
```
dnorm(x = 1, mean = 0, sd = 1)
```

```
## [1] 0.2419707
```

Visualizing the distribution

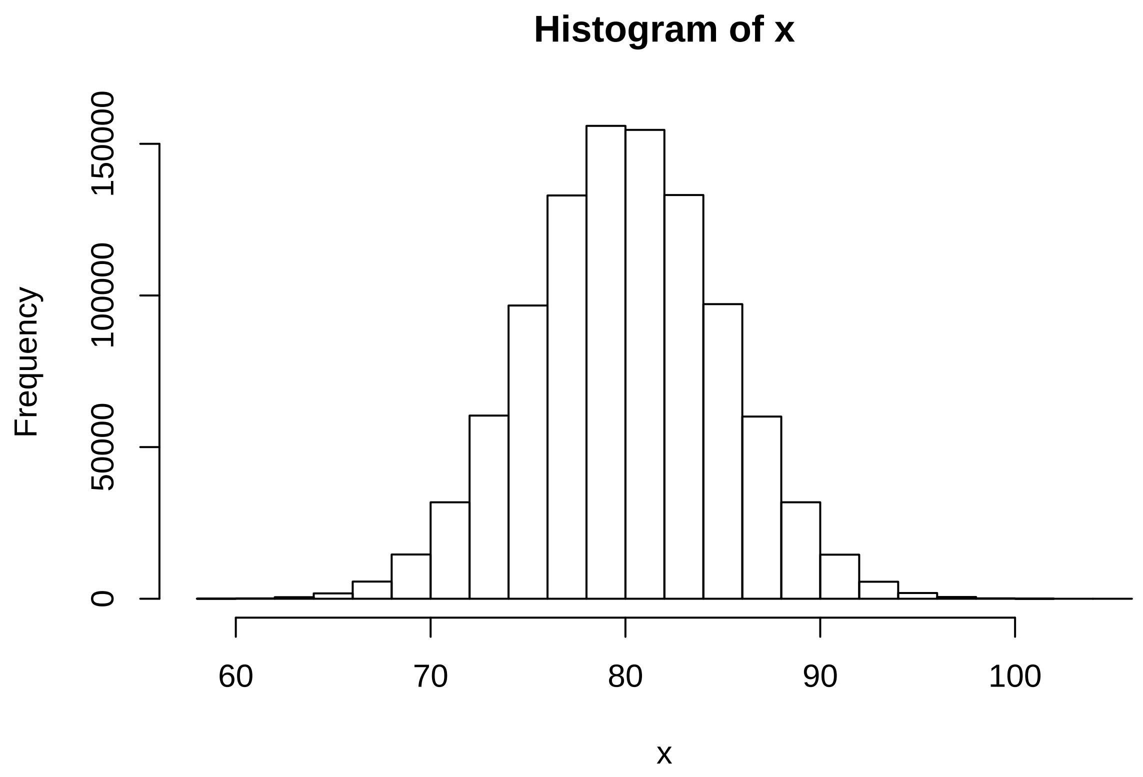
An way to visualize a distribution is to randomly sample a large number of data points from it, then, to estimate its density estimates, and to plot it in a density plot.

```r
x = rnorm(n = 10000, mean = 0, sd = 1)
dens_x <- density(x)
plot(dens_x)
```

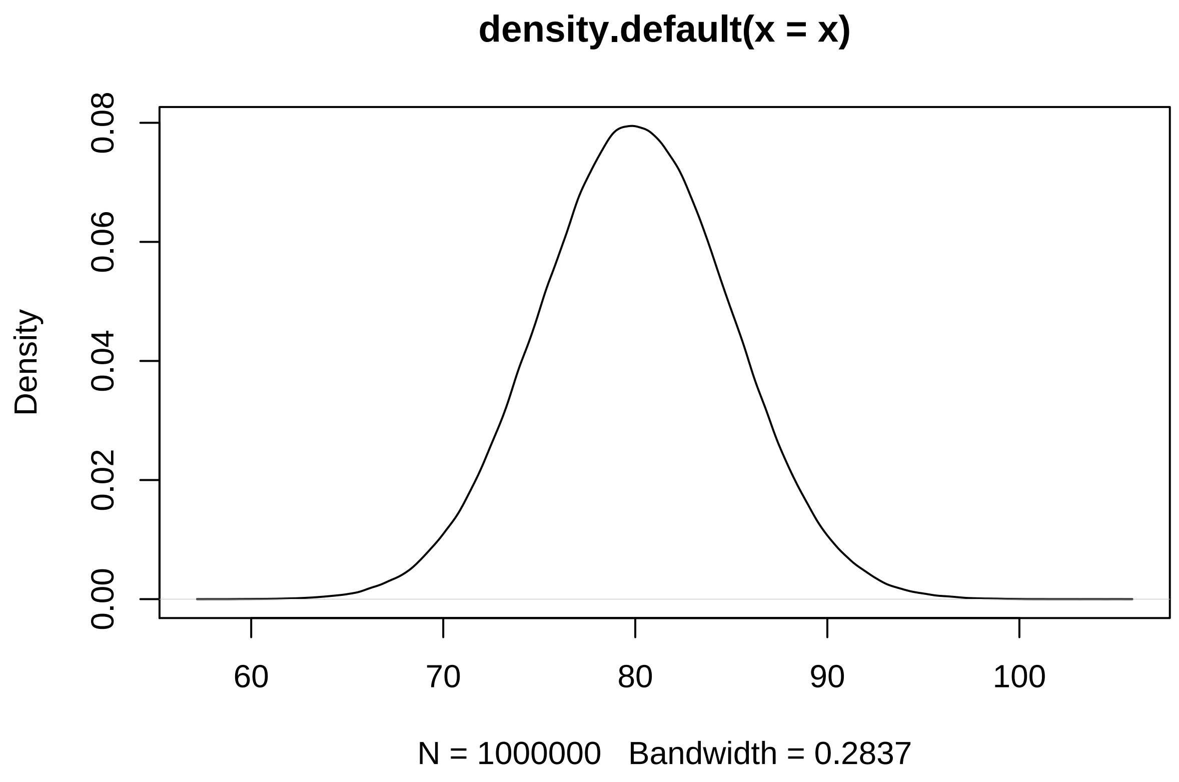


Example 4.1

```r
x <- rnorm(n = 1000000, mean = 80, sd = 5)
hist(x)
```



```r
dens_x <- density(x)
plot(dens_x)
```



We can estimate some of the properties of the Normal Distribution with simulations:

(2) About 67% of the population is within one standard deviation of the mean:

```
## How many elements are within 1 sd from the mean (i.e., 75 < x < 85)
interval <- which(x > 75 & x < 85)
length(interval)/length(x)
```

```
## [1] 0.682942
```

(3) 90% of the population is within 1.645 standard deviations of the mean

```
interval <- which(x > (80 - 1.645*5) & x < 80 + (1.645*5))
length(interval)/length(x)
```

```
## [1] 0.900433
```

(4) 95% of the population is within 1.96 standard deviations of the mean, and hence only 2.5% of the population have values which are greater than 1.96 standard deviations above the mean.

```
interval <- which(x > (80 - 1.96*5) & x < 80 + (1.96*5))
length(interval)/length(x)
```

```
## [1] 0.950032
```

```
interval <- which(x > 80 + (1.96*5))
length(interval)/length(x)
```

```
## [1] 0.024977
```

(5) 99% of the population is within 2.576 standard deviations of the mean

```r
interval <- which(x > (80 - 2.576*5) & x < 80 + (2.576*5))
length(interval)/length(x)
```

```
## [1] 0.990104
```

(6) 99.9% of the population is within 3.29 standard deviations of the mean

```r
interval <- which(x > (80 - 3.29*5) & x < 80 + (3.29*5))
length(interval)/length(x)
```

```
## [1] 0.999024
```

Example 4.2

To find the $\theta(z)$ in the table of the normal distribution function in R, you can use:

```r
pnorm(q = -1.60, mean = 0, sd = 1) #find the theta(-1.60) value
```

```
## [1] 0.05479929
```

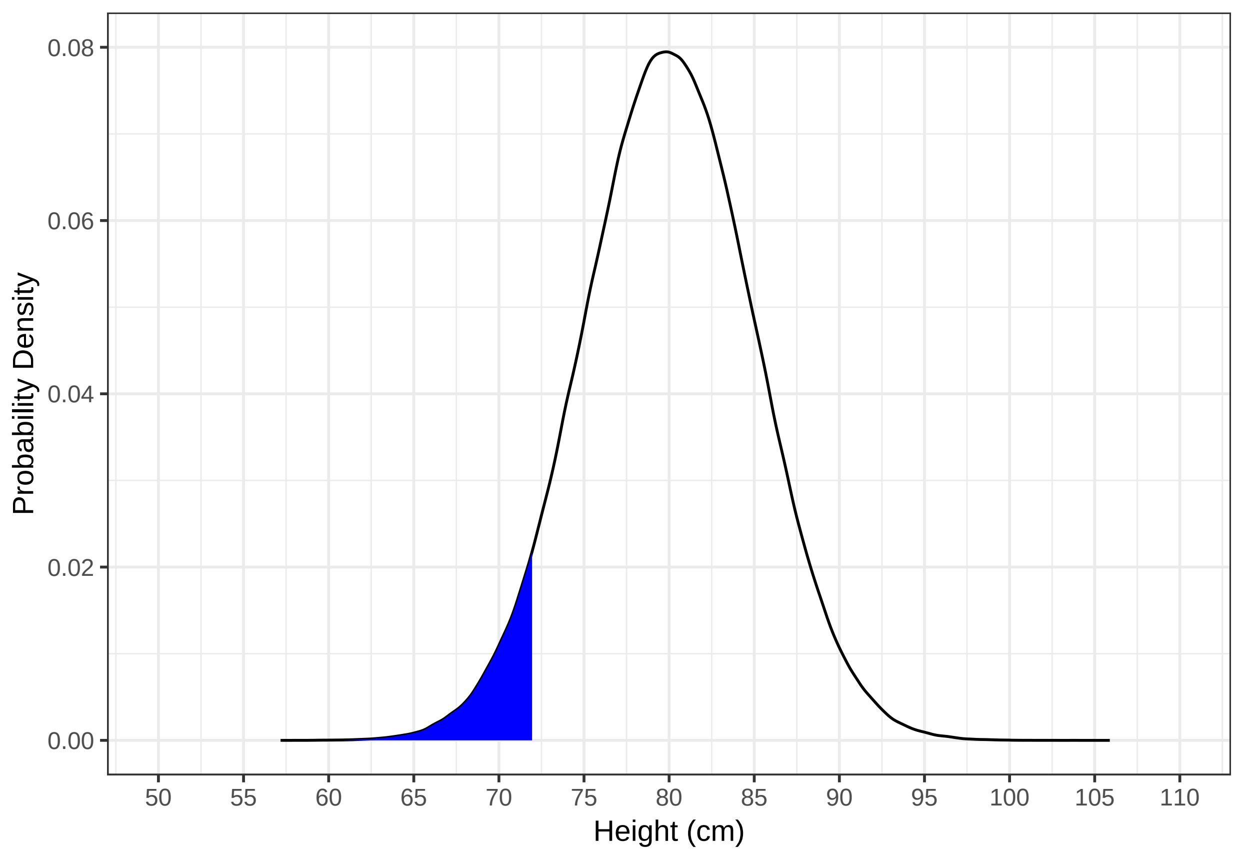
Otherwise, you can just plug the parameters distribution and avoid the use of the table. Find the proportion of plants having heights using `pnorm` function: (1) less than 72 cm

```r
pnorm(q = 72, mean = 80, sd = 5)
```

```
## [1] 0.05479929
```

This value corresponds to the area of the following colored area in probability density plot:

```
## Warning: Removed 356 rows containing missing values (position_stack).
```



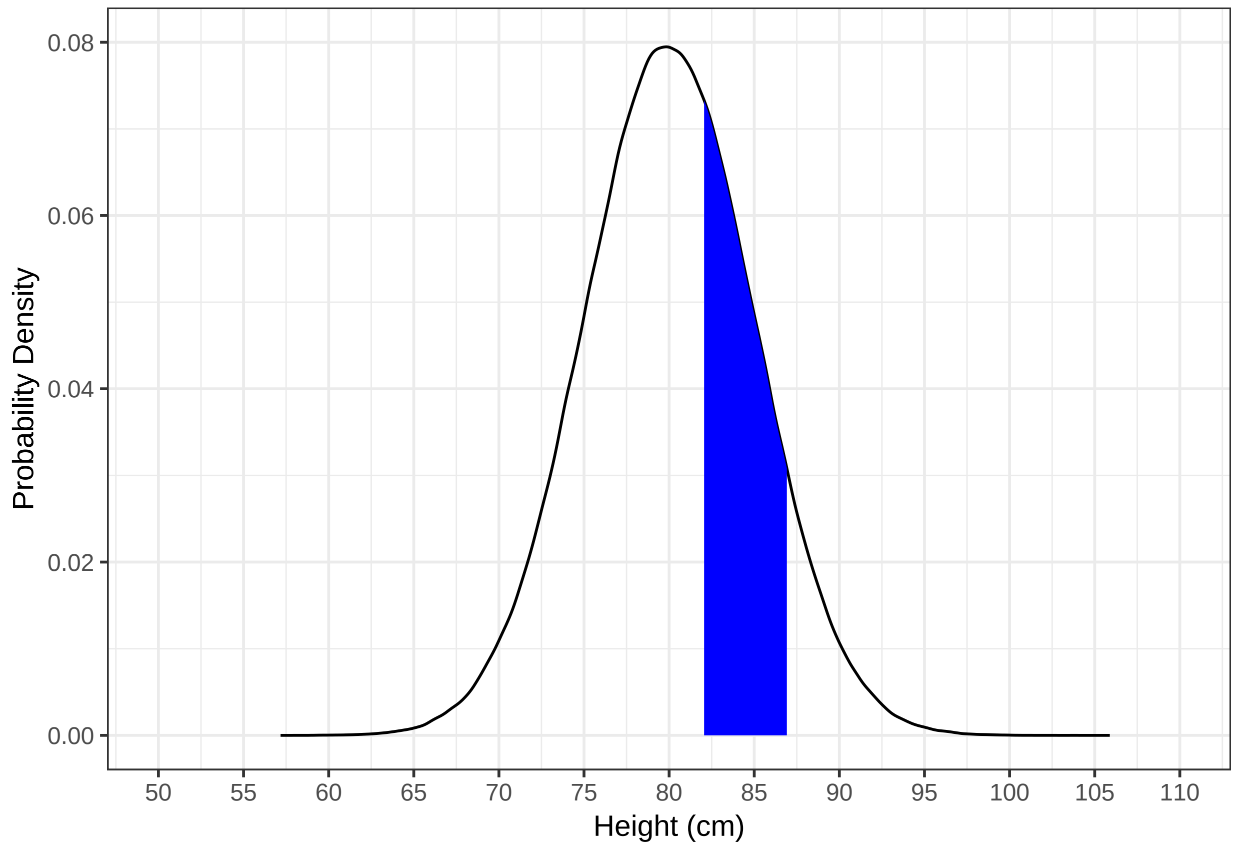
(2) between 82 and 87 cm

```
pnorm(q = 87, mean = 80, sd = 5) - pnorm(q = 82, mean = 80, sd = 5) #attention for the minus signal
```

```
## [1] 0.2638216
```

This value corresponds to the area of the following colored area in probability density plot:

```
## Warning: Removed 460 rows containing missing values (position_stack).
```



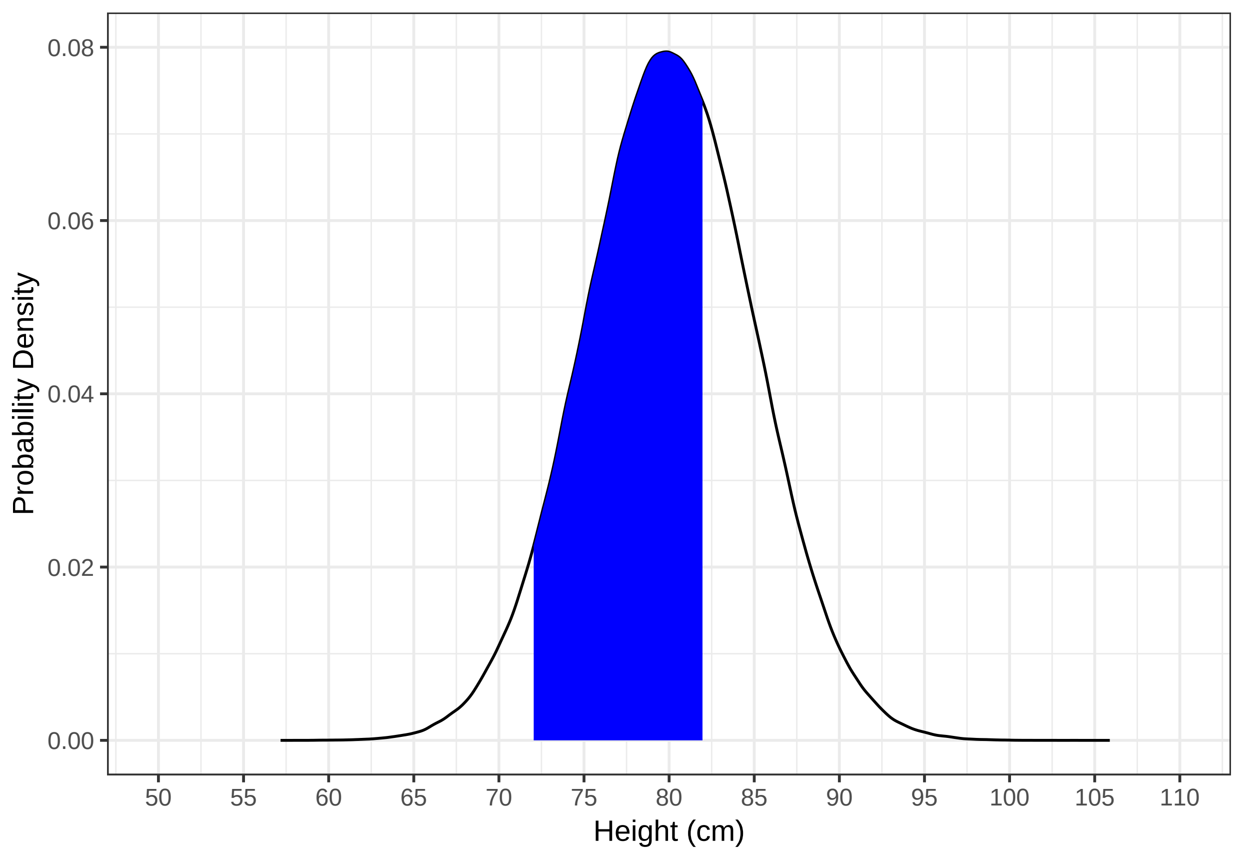
(3) between 72 and 82 cm

```
pnorm(q = 82, mean = 80, sd = 5) - pnorm(q = 72, mean = 80, sd = 5) #attention for the minus signal
```

```
## [1] 0.6006224
```

This value corresponds to area of the following colored area in probability density plot:

```
## Warning: Removed 407 rows containing missing values (position_stack).
```



Example 4.3

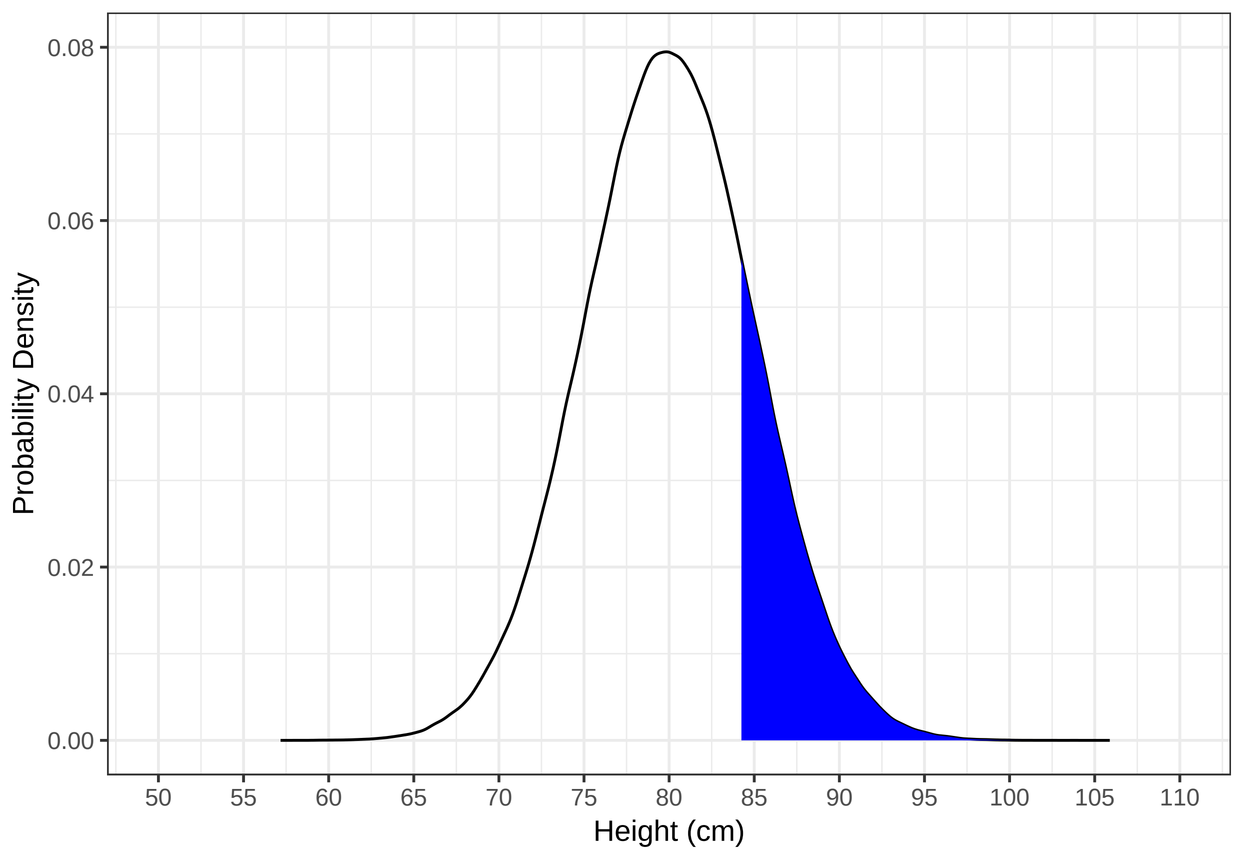
(a) Above what height are the top 20% of plants?

```
qnorm(p = 0.8, mean = 80, sd = 5) #80% of the points are below, then, 20% are above.
```

```
## [1] 84.20811
```

Top 20% of the data is the following colored area in probability density plot:

```
## Warning: Removed 284 rows containing missing values (position_stack).
```



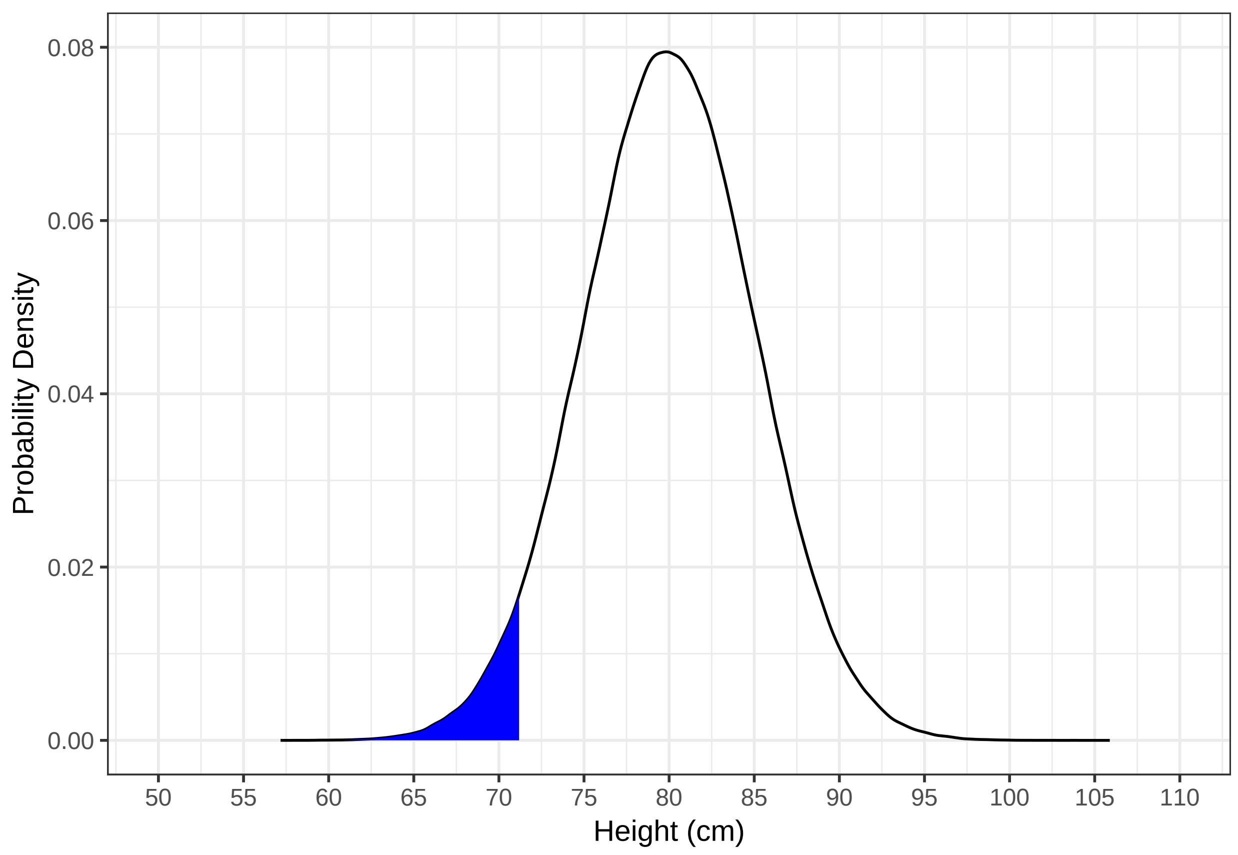
(b) Below what height are the bottom 4% of plants?

```
qnorm(p = 0.04, mean = 80, sd = 5) #80% of the points are below, then, 20% are above.
```

```
## [1] 71.24657
```

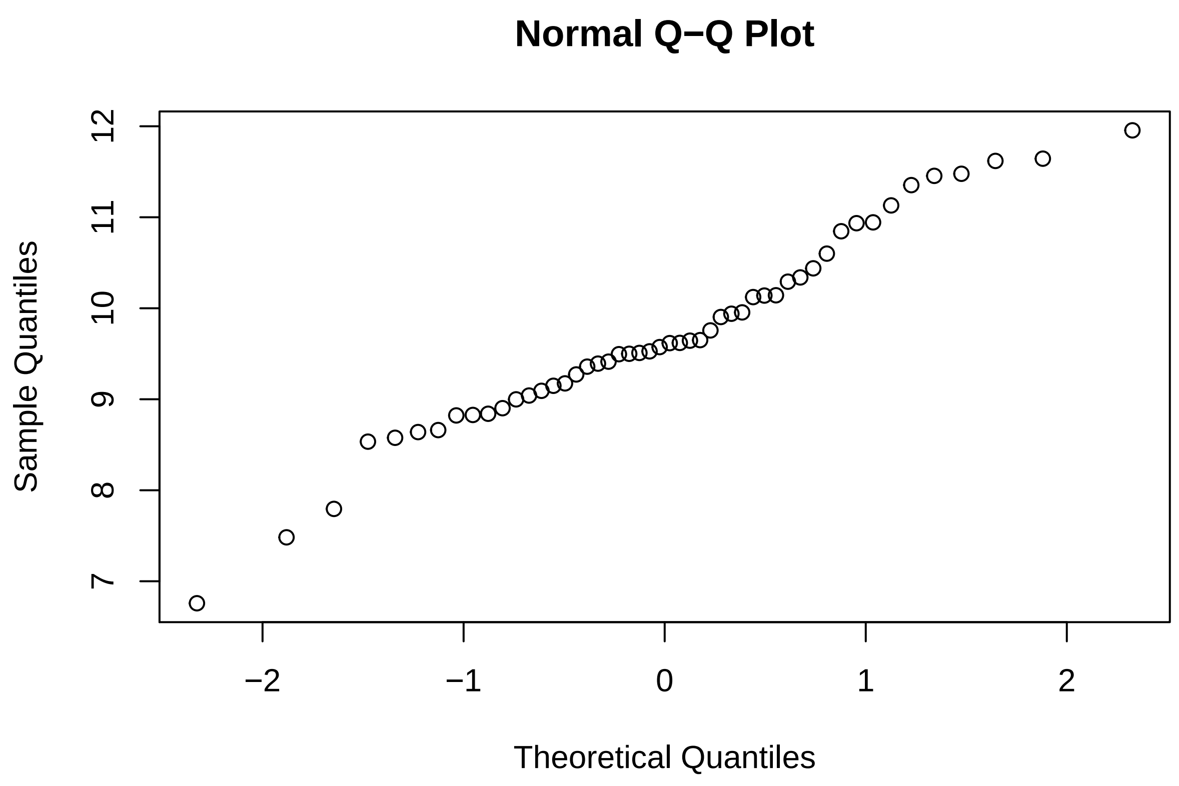
Bottom 4% of the data is the following colored area in probability density plot:

```
## Warning: Removed 364 rows containing missing values (position_stack).
```



Normal probability plot

```r
## Generating some data
x <- rnorm(n = 50, mean = 10, sd = 1)
qqnorm(x)
```



4.5 Example

```
x = c(72.3, 78.9, 82.6, 71.8, 86.1, 80.5, 72.0, 91.8, 77.3, 88.2)
qt(p = 0.975, df = 9) #take the t-table value
```

```
## [1] 2.262157
```

```
## To find the interval you can carry out a t-test
t.test(x, conf.level = 0.95) #CI 95%
```

```
##
##  One Sample t-test
##
## data:  x
## t = 35.864, df = 9, p-value = 5.042e-11
## alternative hypothesis: true mean is not equal to 0
## 95 percent confidence interval:
##  75.09445 85.20555
## sample estimates:
## mean of x
##     80.15
```

```
t.test(x, conf.level = 0.99) #CI 95%
```

```
##
##  One Sample t-test
```

```
##
## data:  x
## t = 35.864, df = 9, p-value = 5.042e-11
## alternative hypothesis: true mean is not equal to 0
## 99 percent confidence interval:
##  72.88714 87.41286
## sample estimates:
## mean of x
##     80.15
```

4.6 Example

```r
x = c(171.8, 267.7, 274.7, 203.2, 208.6, 267.2, 184.1, 234.5)
qt(p = 0.975, df = 7) #take the t-table value
```

```
## [1] 2.364624
```

```r
## To find the interval you can carry out a t-test
t.test(x, conf.level = 0.95) #CI 95%
```

```
##
##  One Sample t-test
##
## data:  x
## t = 15.877, df = 7, p-value = 9.536e-07
## alternative hypothesis: true mean is not equal to 0
## 95 percent confidence interval:
##  192.7453 260.2047
## sample estimates:
## mean of x
##   226.475
```

Chapter 5: Introduction to Hypothesis Testing

5.1 Example

```
x = c(2.6, 2.1, 2.5, 2.4, 1.9, 2.3)
t.test(x, mu = 2.0, conf.level = 0.95) #CI 95%
```

```
##
##  One Sample t-test
##
## data:  x
## t = 2.818, df = 5, p-value = 0.0372
## alternative hypothesis: true mean is not equal to 2
## 95 percent confidence interval:
##  2.026341 2.573659
## sample estimates:
## mean of x
##       2.3
```

```
## p-value = 0.0373 which is lower than 5%, therefore, we reject H0 at the 5% level.
```

5.1 Example

```
x = c(8.1, 8.7, 9.2, 7.8, 8.4, 9.4)
t.test(x, mu = 8, conf.level = 0.95)
```

```
##
##  One Sample t-test
##
## data:  x
## t = 2.3595, df = 5, p-value = 0.0648
## alternative hypothesis: true mean is not equal to 8
## 95 percent confidence interval:
##  7.94631 9.25369
## sample estimates:
## mean of x
##       8.6
```

```
## computed p-value = 0.0648 which is higher than 5%, therefore, we don't reject H0 at the 5% level.
```

5.2 Example

```
A <- c(17.8, 18.5, 12.2, 19.7, 10.8, 11.9, 15.6, 12.5)
B <- c(14.7, 15.2, 12.9, 18.3, 10.1, 12.2, 13.5, 9.9)
diff <- A-B #compute the difference
t.test(diff)
```

```
##
##  One Sample t-test
##
## data:  diff
## t = 2.8446, df = 7, p-value = 0.02488
## alternative hypothesis: true mean is not equal to 0
## 95 percent confidence interval:
##  0.2573084 2.7926916
## sample estimates:
## mean of x
##     1.525
```

```
## computed p-value = 0.02488 which is lower than 5%, therefore, we reject H0 at the 5% level.
```

Chapter 6: Introduction to Hypothesis Testing

6.1 Example

```r
Control = c(2.65, 3.28, 2.62, 2.84, 2.61, 2.80, 2.23, 2.45, 2.95, 3.12)
Growth = c(3.02, 2.21, 2.29, 3.21, 3.30, 3.13, 2.86, 3.35, 2.72, 3.16)
mean(Control)
```

```
## [1] 2.755
```

```r
mean(Growth)
```

```
## [1] 2.925
```

Changin the two values

```r
Control = c(2.65, 3.28, 2.62, 2.84, 2.61, 2.80, 3.23, 2.45, 2.95, 3.12)
Growth = c(3.02, 2.21, 2.29, 3.21, 3.30, 3.13, 2.86, 2.35, 2.72, 3.16)
mean(Control)
```

```
## [1] 2.855
```

```r
mean(Growth)
```

```
## [1] 2.825
```

6.2 Example

```r
New = c(2.6, 2.1, 2.5, 2.4, 1.9, 2.3)
Standard = c(1.7, 2.1, 2.0, 1.8, 2.3, 1.6, 2.0, 2.1, 2.2, 1.9)
t.test(New, Standard, var.equal = TRUE)
```

```
##
##  Two Sample t-test
##
## data:  New and Standard
## t = 2.7056, df = 14, p-value = 0.01707
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  0.06840468 0.59159532
## sample estimates:
## mean of x mean of y
##      2.30      1.97
```

```r
## computed p-value = 0.01707 which is lower than 5%, therefore, we reject H0 at the 5% level.
```

6.3 Example

```r
var.test(New,Standard)
```

```
##
##  F test to compare two variances
##
## data:  New and Standard
## F = 1.3878, num df = 5, denom df = 9, p-value = 0.6296
## alternative hypothesis: true ratio of variances is not equal to 1
## 95 percent confidence interval:
##  0.309462 9.271667
## sample estimates:
## ratio of variances
##           1.387755
```

```
## computed p-value = 0.6296 which is higer than 5%, therefore, we don't reject H0 at the 5% level.
```

Chapter 7: Linear Regression and Correlation

7.1 Example

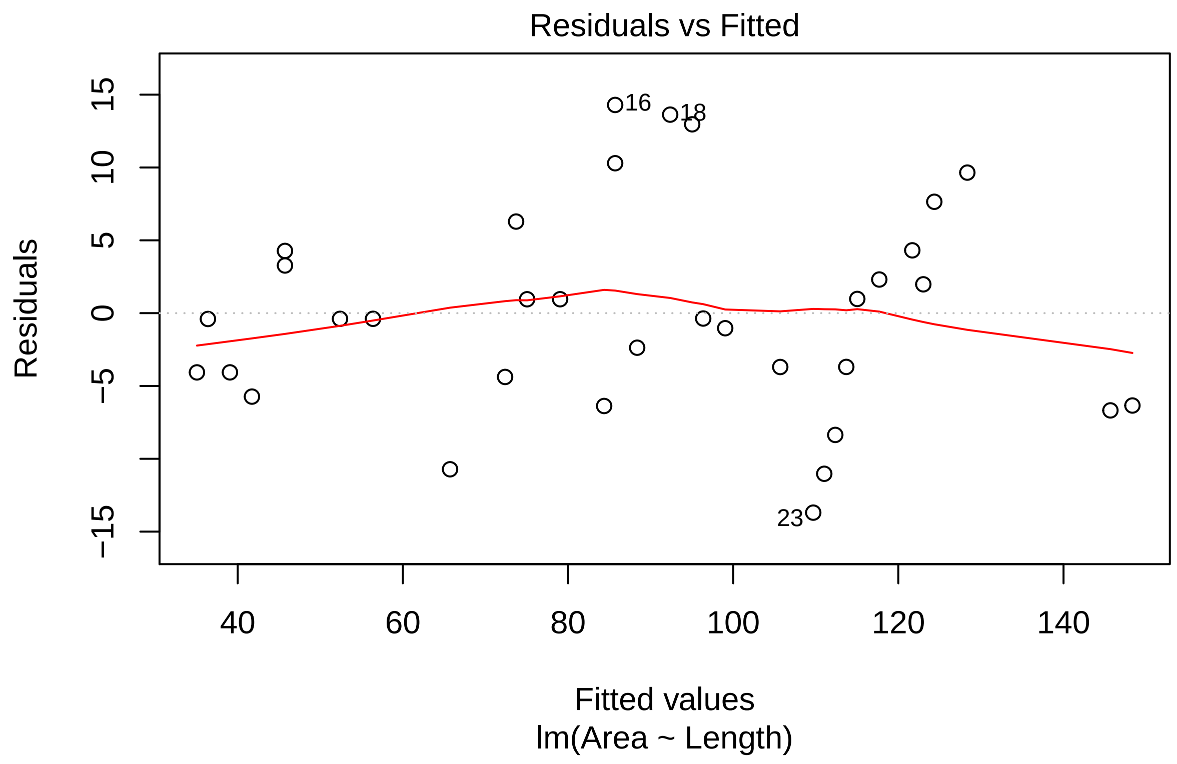
```
Length = c(22, 23, 25, 27, 30, 30, 35, 38, 45,
           50, 51, 52, 55, 59, 60, 60, 62, 65,
           67, 68, 70, 75, 78, 79, 80, 81,
           82, 84, 87, 88, 89, 92, 105, 107)

Area = c(31, 36, 35, 36, 50, 49, 52, 56, 55,
         68, 80, 76, 80, 78, 96, 100, 86, 106,
         108, 96, 98, 102, 96, 100, 104, 110,
         116, 120, 126, 125, 132, 138, 139, 142)

## The Linear Model (regression)
model = lm(Area ~ Length)
summary(model)
```

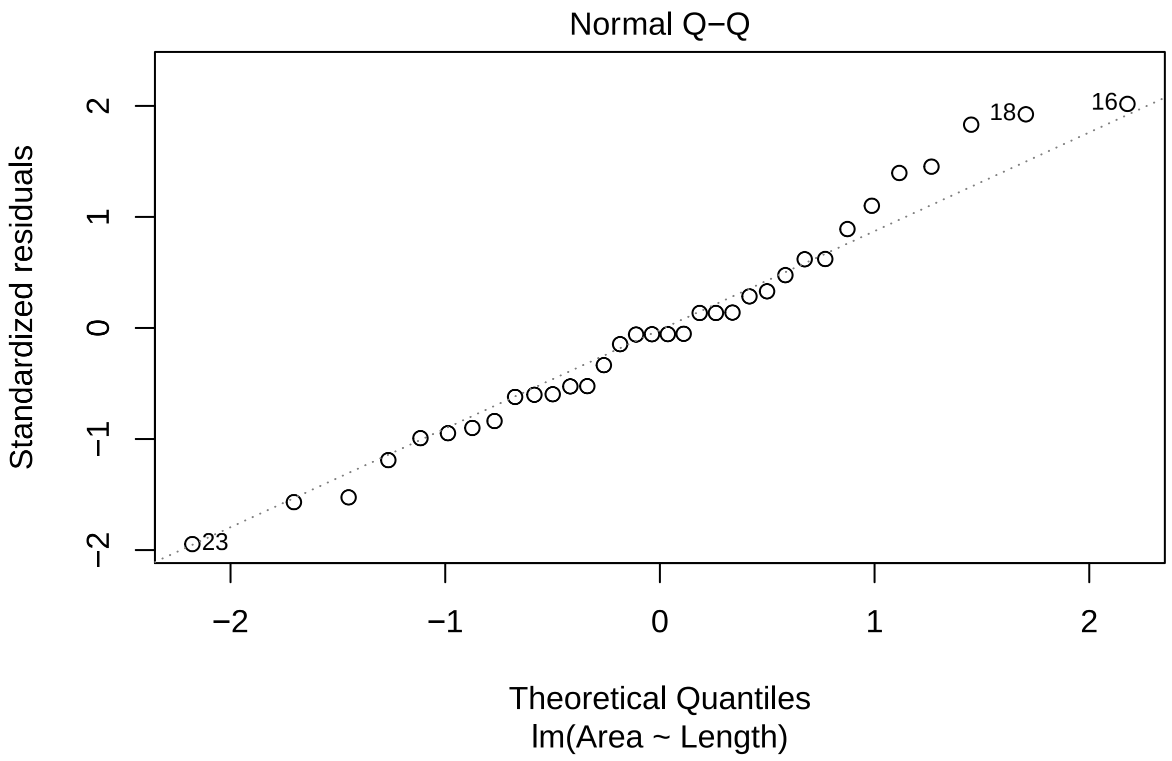
```
##
## Call:
## lm(formula = Area ~ Length)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -13.6951  -4.3027  -0.3905   4.0217  14.2925
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  5.74914    3.47886   1.653    0.108
## Length       1.33264    0.05215  25.555   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 7.189 on 32 degrees of freedom
## Multiple R-squared:  0.9533, Adjusted R-squared:  0.9518
## F-statistic: 653.1 on 1 and 32 DF,  p-value: < 2.2e-16
```

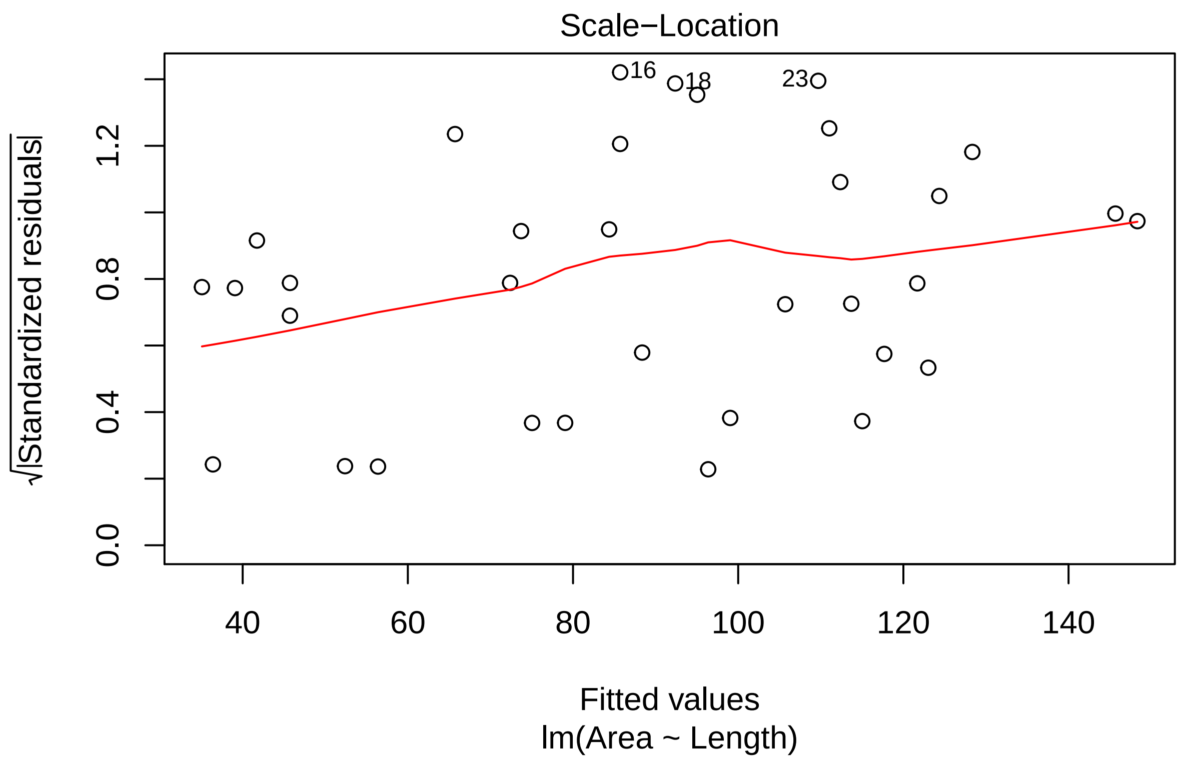
```
## Analysis of Variance
anova(model)
```

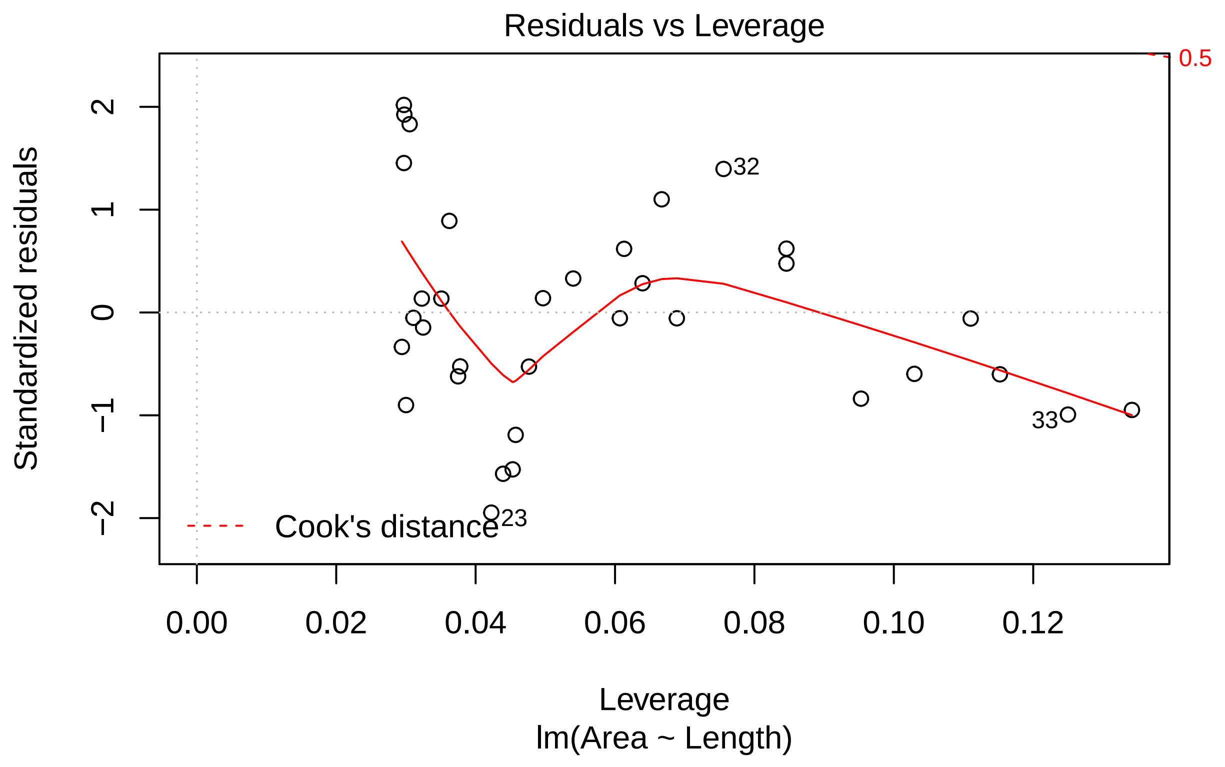
```
## Analysis of Variance Table
##
## Response: Area
##           Df Sum Sq Mean Sq F value    Pr(>F)
## Length     1  33750   33750  653.07 < 2.2e-16 ***
## Residuals 32   1654      52
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
## Residual Diagnostics
plot(model)
```

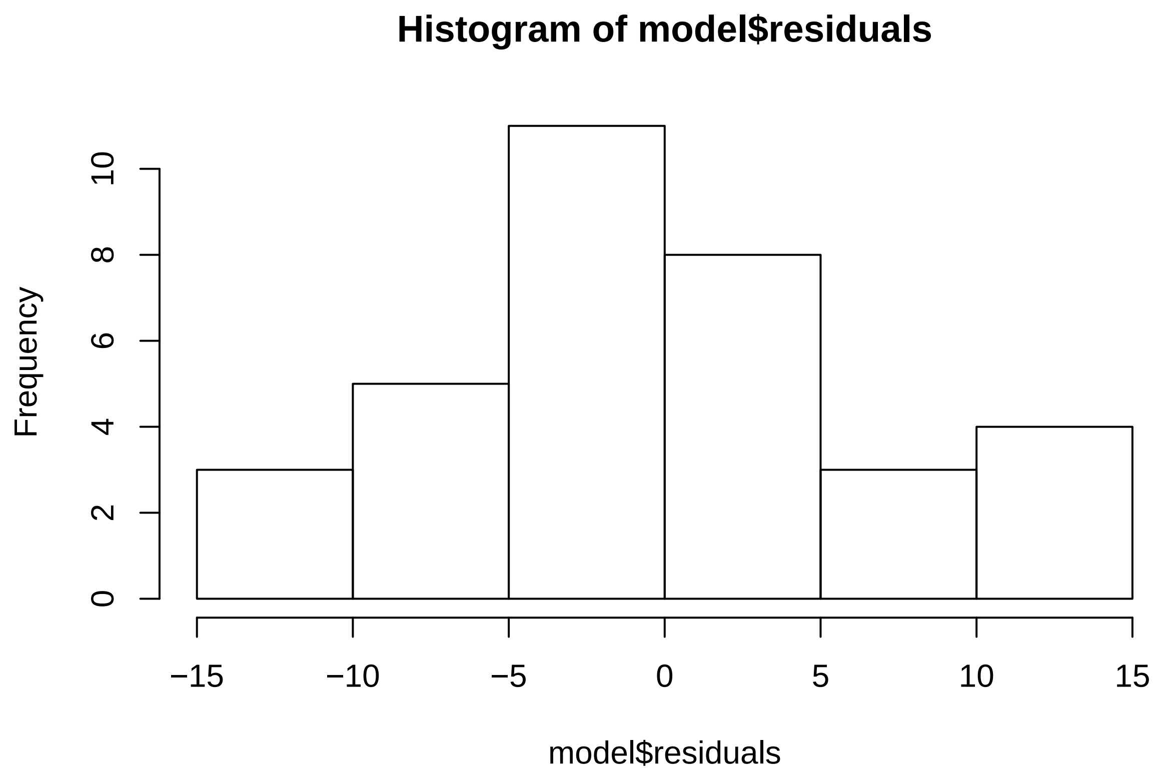

Residuals vs Fitted
Residuals
15
10
5
0
−5
−15
40
60
80
100
120
140
16
18
23
Fitted values
lm(Area ~ Length)


Normal Q−Q
Standardized residuals
Theoretical Quantiles
lm(Area ~ Length)
23
18
16
−2
−1
0
1
2


Scale−Location
√|Standardized residuals|
0.0
0.4
0.8
1.2
40
60
80
100
120
140
16
18
23
Fitted values
lm(Area ~ Length)



```r
hist(model$residuals)
```



7.2 Example

```r
x = c(0, 25, 50, 75, 100, 125)
y = c(3.70, 4.45, 4.75, 5.20, 5.15, 4.95)
model = lm(y ~ x)
summary(model)
```

```
##
## Call:
## lm(formula = y ~ x)
##
## Residuals:
##        1        2        3        4        5        6
## -0.37143  0.12714  0.17571  0.37429  0.07286 -0.37857
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 4.071429   0.249066  16.347  8.2e-05 ***
## x           0.010057   0.003291   3.056   0.0378 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.3441 on 4 degrees of freedom
## Multiple R-squared:  0.7002, Adjusted R-squared:  0.6252
## F-statistic: 9.341 on 1 and 4 DF,  p-value: 0.03779
```

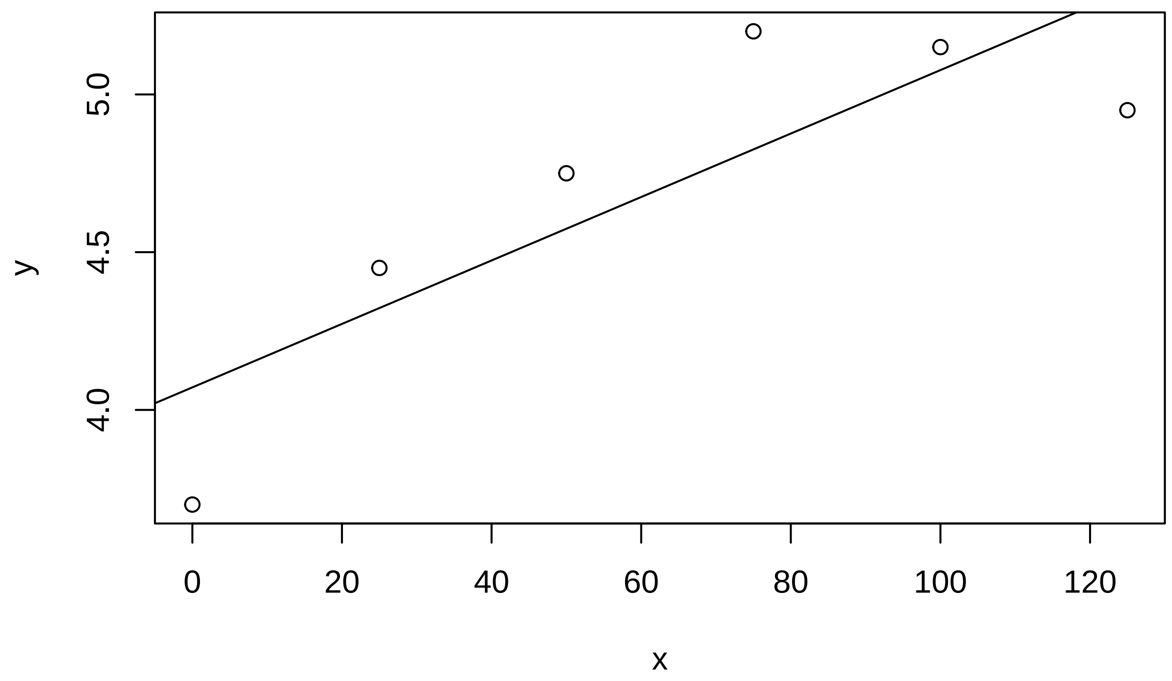
7.3 Example

```r
anova(model)
```

```
## Analysis of Variance Table
##
## Response: y
##           Df  Sum Sq Mean Sq F value  Pr(>F)
## x          1 1.10629 1.10629  9.3414 0.03779 *
## Residuals  4 0.47371 0.11843
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Graphics

```r
plot(x,y)
abline(model)
```



Chapter 8: Curve Fitting

8.1 Example

```r
y = c(3.70, 4.45, 4.75, 5.20, 5.15, 4.95)
x = c(0, 25, 50, 75, 100, 125)
x2 = x^2
model = lm(y ~ x + x2)
summary(model)
```

```
##
## Call:
## lm(formula = y ~ x + x2)
##
## Residuals:
##          1          2          3          4          5          6
## -0.0053571  0.0539286 -0.1171429  0.0814286 -0.0003571 -0.0125000
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  3.705e+00  8.012e-02   46.24 2.23e-05 ***
## x            3.202e-02  3.015e-03   10.62  0.00178 **
## x2          -1.757e-04  2.315e-05   -7.59  0.00474 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.08841 on 3 degrees of freedom
## Multiple R-squared:  0.9852, Adjusted R-squared:  0.9753
## F-statistic: 99.58 on 2 and 3 DF,  p-value: 0.001808
```

```r
anova(model)
```

```
## Analysis of Variance Table
##
## Response: y
##           Df  Sum Sq Mean Sq F value   Pr(>F)
## x          1 1.10629 1.10629 141.551 0.001277 **
## x2         1 0.45027 0.45027  57.612 0.004745 **
## Residuals  3 0.02345 0.00782
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

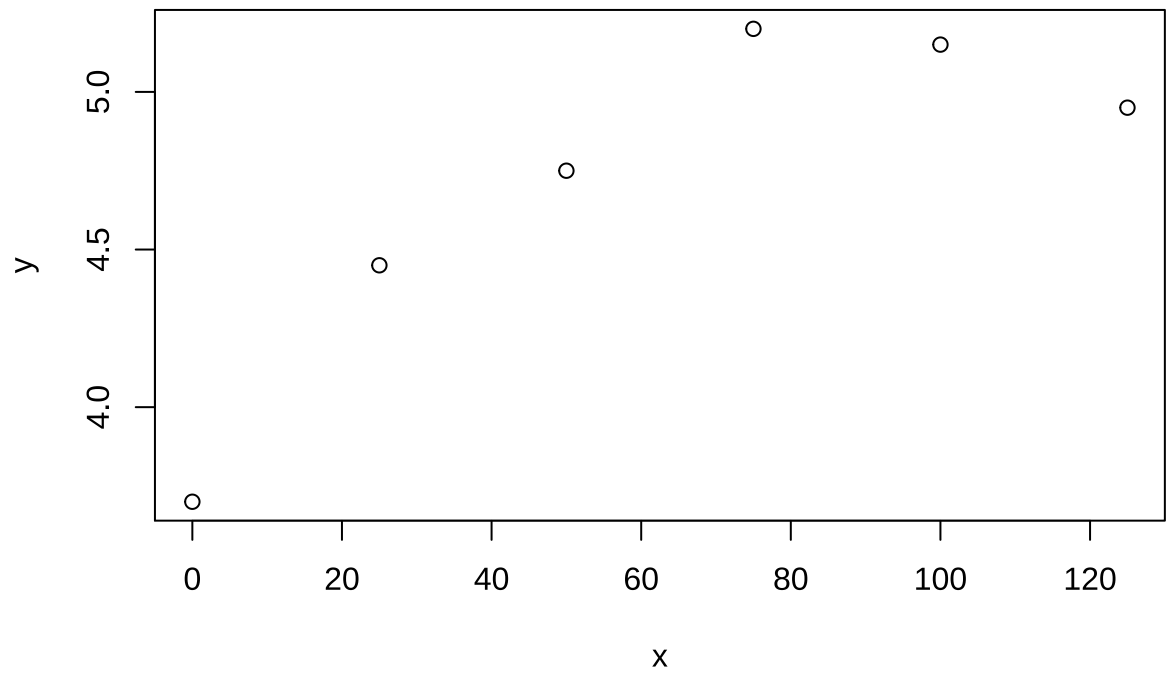
```r
model$fitted.values
```

```
##        1        2        3        4        5        6
## 3.705357 4.396071 4.867143 5.118571 5.150357 4.962500
```

```r
model$residuals
```

```
##             1             2             3             4             5
## -0.0053571429  0.0539285714 -0.1171428571  0.0814285714 -0.0003571429
##             6
## -0.0125000000
```

```r
plot(x,y)
```



8.2 Example

```
x = c(2.5, 3, 7.2, 7.8, 8.3, 9.8,
      10.8, 15.5, 24., 31.5, 40.2, 64.4)

y = c(5.5, 7.9, 9.8, 11, 13.6, 10.9,
      12.3, 17.5, 20.5, 25.6, 20.4, 26.8)
lnx = log(x)
lny = log(y)


## Model 1: y = a + bx
model1 = lm(y~x)
summary(model1)
```

```
##
## Call:
## lm(formula = y ~ x)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4.2375 -1.9219 -0.8859  2.3061  6.2033
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   8.9048     1.3901   6.406 7.78e-05 ***
## x             0.3331     0.0539   6.179 0.000104 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.306 on 10 degrees of freedom
## Multiple R-squared:  0.7925, Adjusted R-squared:  0.7717
## F-statistic: 38.18 on 1 and 10 DF,  p-value: 0.0001042
```

```
model1$fitted.values
```

```
##         1         2         3         4         5         6         7
##  9.737508  9.904046 11.302967 11.502813 11.669351 12.168966 12.502042
##         8         9        10        11        12
## 14.067502 16.898651 19.396724 22.294489 30.354939
```

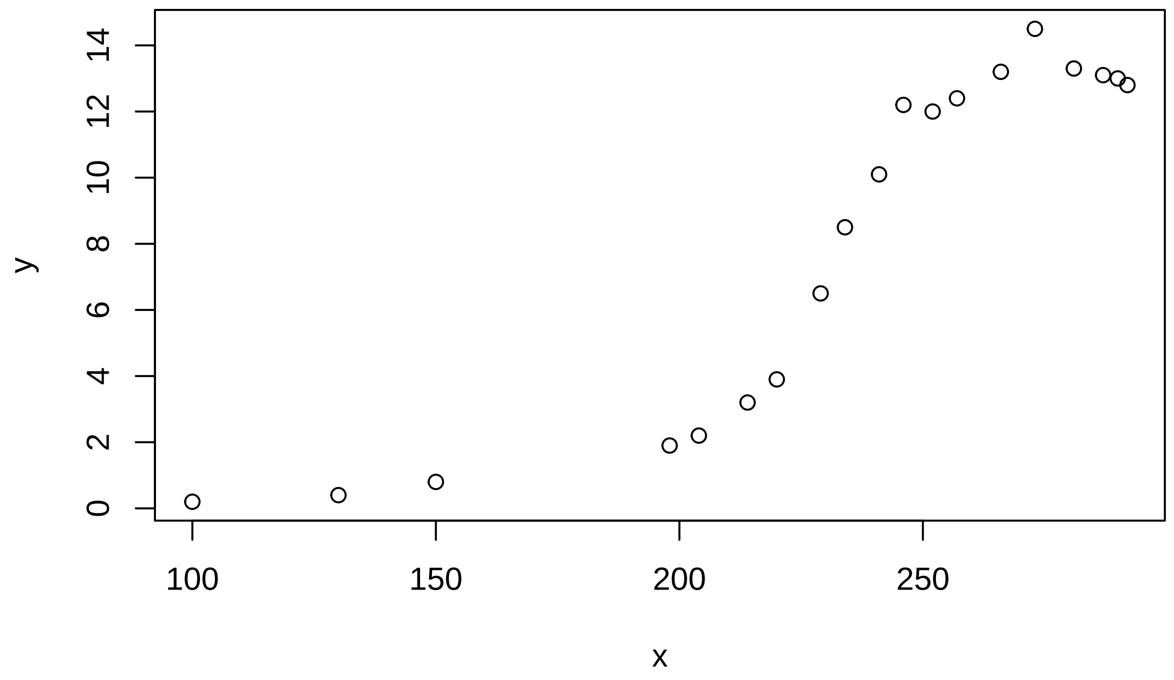
```
model1$residuals
```

```
##          1          2          3          4          5          6
## -4.2375081 -2.0040464 -1.5029673 -0.5028132  1.9306486 -1.2689660
##          7          8          9         10         11         12
## -0.2020424  3.4324984  3.6013488  6.2032756 -1.8944893 -3.5549387
```

```
## Model 2: ln(y) = a + ln(x)
model2 = lm(lny~lnx)
```

8.3 Example

```
x = c(100, 130, 150, 198, 204, 214, 220, 229, 234,
      241, 246, 252, 257, 266, 273, 281, 287, 290, 292)

y = c(0.2, 0.4, 0.8, 1.9, 2.2, 3.2, 3.9, 6.5, 8.5,
      10.1, 12.2, 12, 12.4, 13.2, 14.5, 13.3, 13.1, 13, 12.8)

plot(x,y)
```



```
## Logistic growth curve
model = nls(y~alpha/(1+exp(h+c*x)), start=list(alpha=13.54, h=19.70, c=-0.0864))
```

Chapter 9: The Completely Randomized Design

9.1 Example

From Chapter 9 and beyond we should have our data settled in a data frame (spreadsheet) which is a combination of vectors of same size. You can create it internally in R or in a externally with the aid of a spreadsheet software (MS Excel, LibreOffice Calc etc) and import in R with `Import Dataset` from RStudio.

Below, we build the data frame of Example 9.1. You can read the data in different ways. Here, we read the data row after row based on Table 9.3.

```
data = data.frame(variety = c("A", "D", "B", "D", "C",
                              "C", "D", "D", "A", "D",
                              "A", "B", "C", "C", "B",
                              "A", "B", "A", "C", "B"),
                  yield = c(22.2, 23.9, 24.1, 21.7, 25.9,
                            18.4, 24.8, 28.2, 17.3, 26.4,
                            21.2, 30.3, 23.2, 21.9, 27.4,
                            25.2, 26.4, 16.1, 22.6, 34.8))

## To access the vectors use $ sign
print(data)
```

```
##    variety yield
## 1        A  22.2
## 2        D  23.9
## 3        B  24.1
## 4        D  21.7
## 5        C  25.9
## 6        C  18.4
## 7        D  24.8
## 8        D  28.2
## 9        A  17.3
## 10       D  26.4
## 11       A  21.2
## 12       B  30.3
## 13       C  23.2
## 14       C  21.9
## 15       B  27.4
## 16       A  25.2
## 17       B  26.4
## 18       A  16.1
## 19       C  22.6
## 20       B  34.8
```

```
data$variety
```

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

```
data$yield
```

```
##  [1] 22.2 23.9 24.1 21.7 25.9 18.4 24.8 28.2 17.3 26.4 21.2 30.3 23.2 21.9
## [15] 27.4 25.2 26.4 16.1 22.6 34.8
```

```r
## Always check the structure of your data:
str(data)
```

```
## 'data.frame':    20 obs. of  2 variables:
##  $ variety: Factor w/ 4 levels "A","B","C","D": 1 4 2 4 3 3 4 4 1 4 ...
##  $ yield  : num  22.2 23.9 24.1 21.7 25.9 18.4 24.8 28.2 17.3 26.4 ...
```

```r
#num = quantitative variable
#factor = qualitative variable
```

Be carefull, always check your data structure, if the variable types are correct, if the number of factor levels are correct etc.

```r
tapply(data$yield, data$variety, length) #N
```

```
## A B C D
## 5 5 5 5
```

```r
tapply(data$yield, data$variety, mean) #Mean
```

```
##    A    B    C    D
## 20.4 28.6 22.4 25.0
```

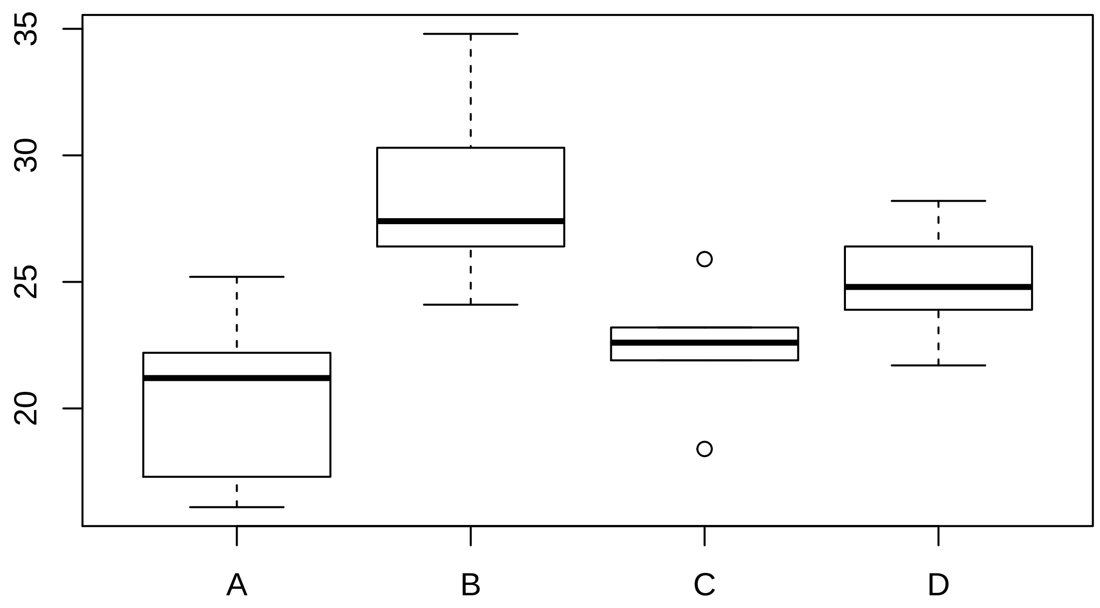
```r
tapply(data$yield, data$variety, sd) #StDev
```

```
##        A        B        C        D
## 3.708773 4.118859 2.700926 2.466779
```

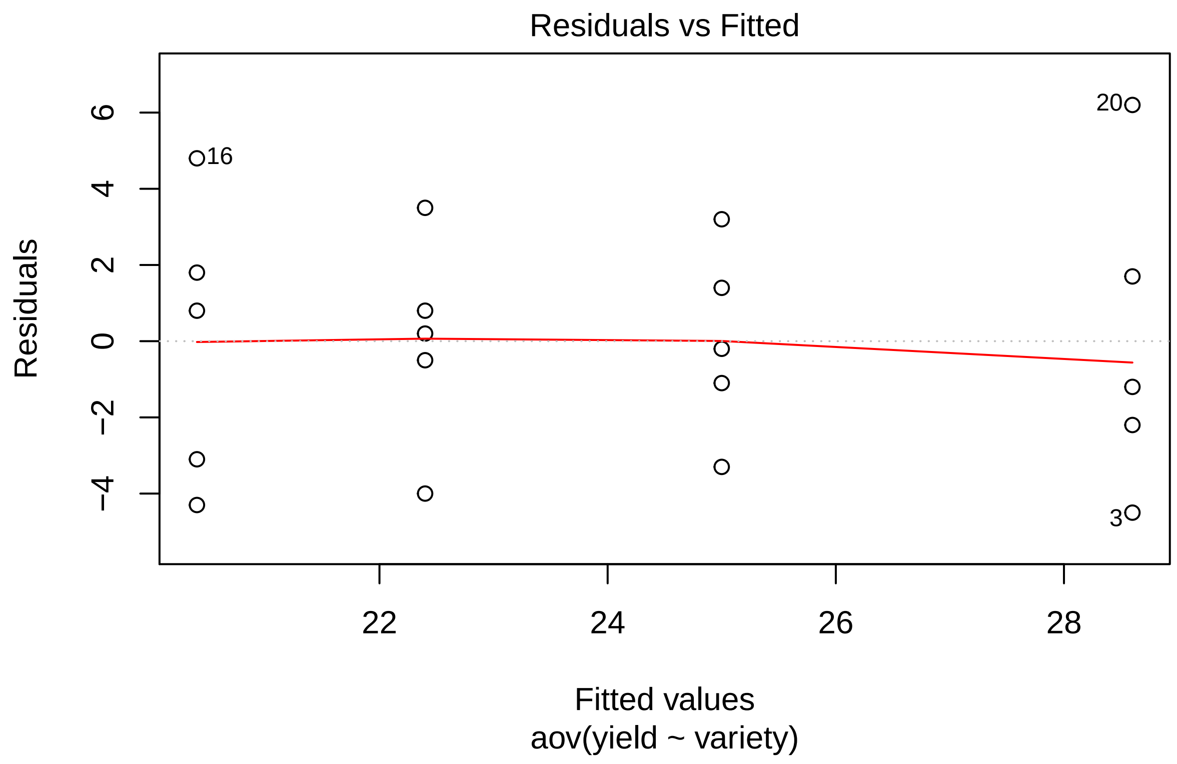
```r
model = aov(yield ~ variety, data = data)
anova(model)
```

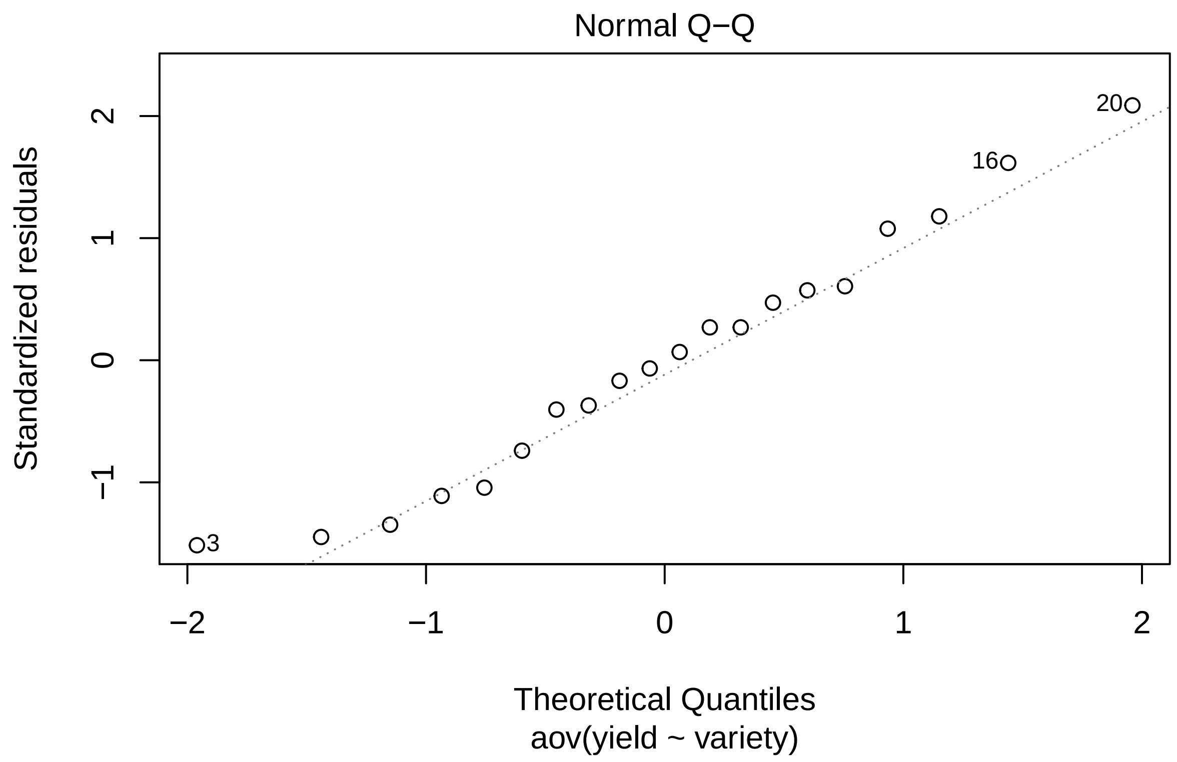
```
## Analysis of Variance Table
##
## Response: yield
##           Df Sum Sq Mean Sq F value  Pr(>F)
## variety    3  188.2  62.733  5.6901 0.00756 **
## Residuals 16  176.4  11.025
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

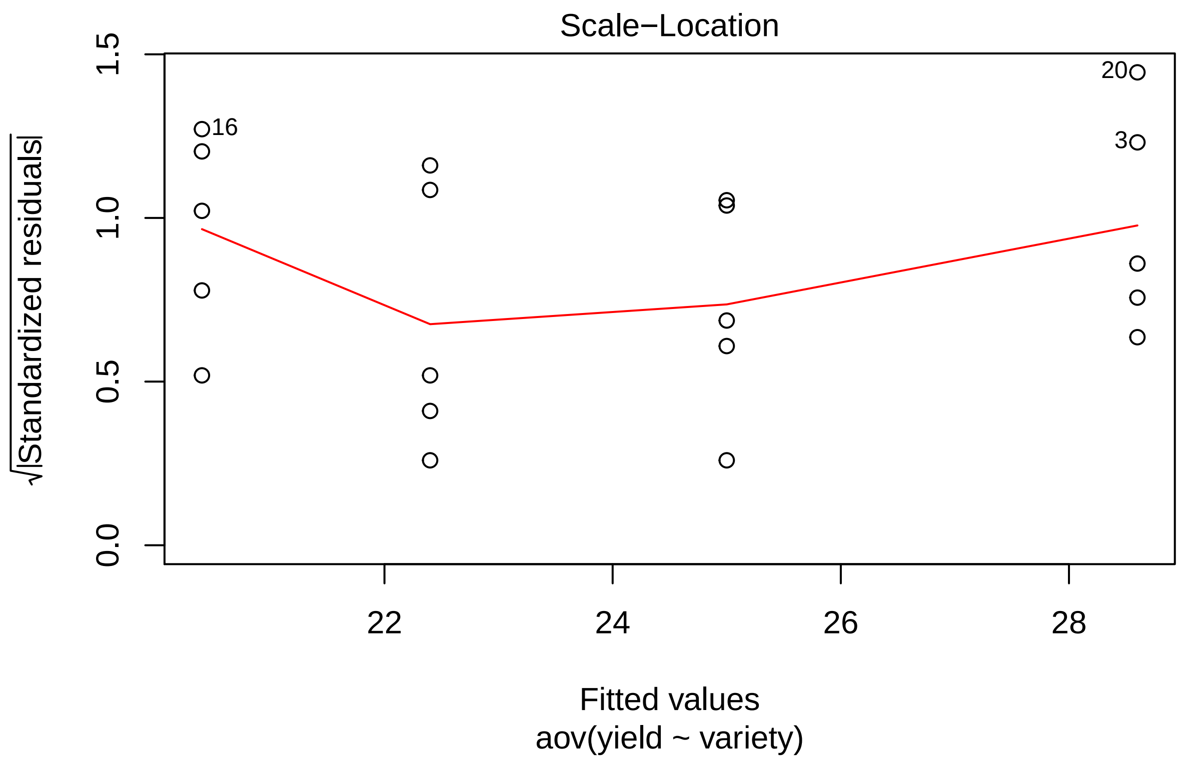
```r
boxplot(yield ~ variety, data = data)
```

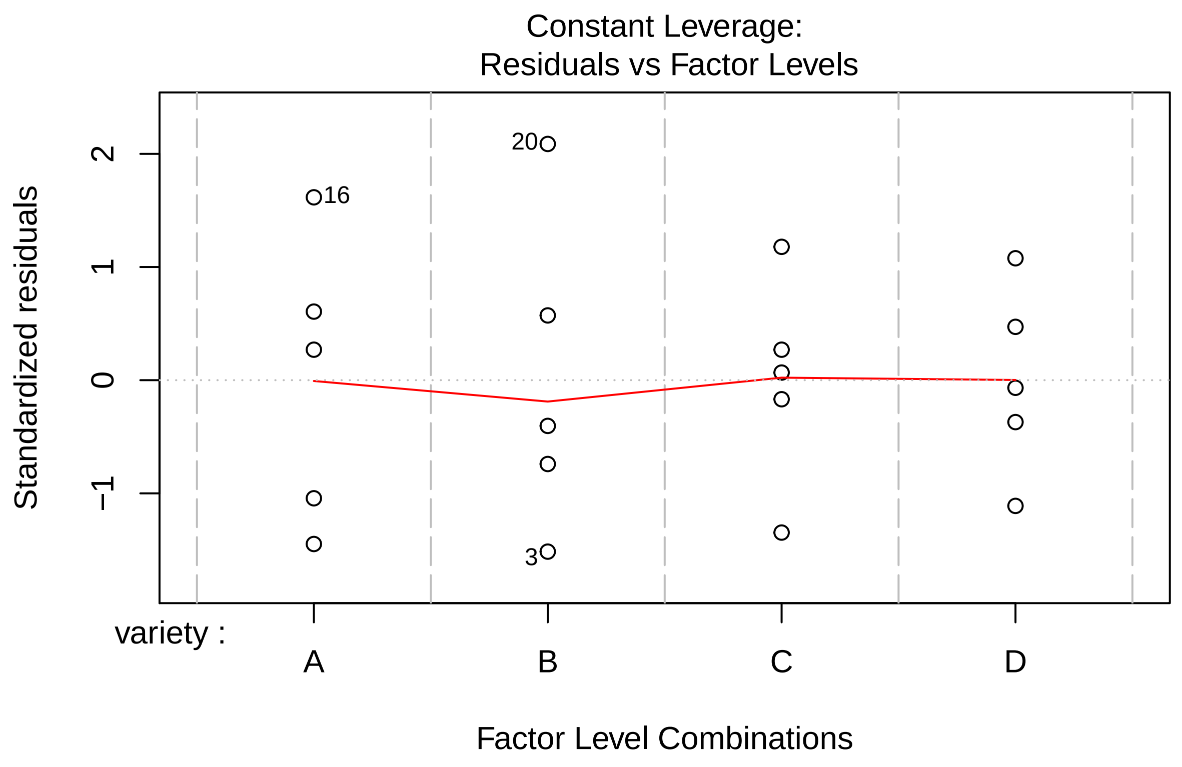


```r
plot(model)
```


Residuals vs Fitted
Residuals
6
4
2
0
−2
−4
16
20
3
22
24
26
28
Fitted values
aov(yield ~ variety)


Normal Q−Q
Standardized residuals
Theoretical Quantiles
aov(yield ~ variety)
3
16
20
−2
−1
0
1
2


Scale−Location
√|Standardized residuals|
0.0
0.5
1.0
1.5
22
24
26
28
Fitted values
aov(yield ~ variety)
16
20
3



9.2 Example

```
data = data.frame(variety = c("A","A","A",
                              "B","B","B","B",
                              "C","C","C","C",
                              "D","D","D","D","D"),
                  yield = c(17.3, 21.2, 16.1,
                            24.1, 30.3, 26.4, 34.8,
                            25.9, 18.4, 21.9, 22.6,
                            23.9, 21.7, 24.8, 28.2, 26.4))

tapply(data$yield, data$variety, length) #N
```

```
## A B C D
## 3 4 4 5
```

```
tapply(data$yield, data$variety, mean) #Mean
```

```
##    A    B    C    D
## 18.2 28.9 22.2 25.0
```

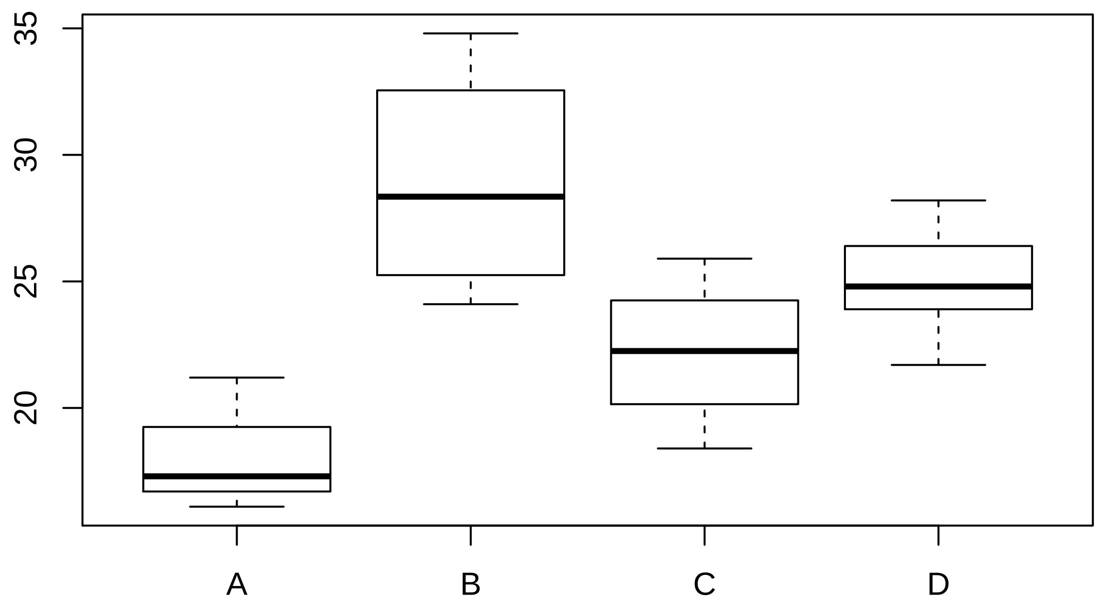
```
tapply(data$yield, data$variety, sd) #StDev
```

```
##        A        B        C        D
## 2.666458 4.692547 3.075711 2.466779
```

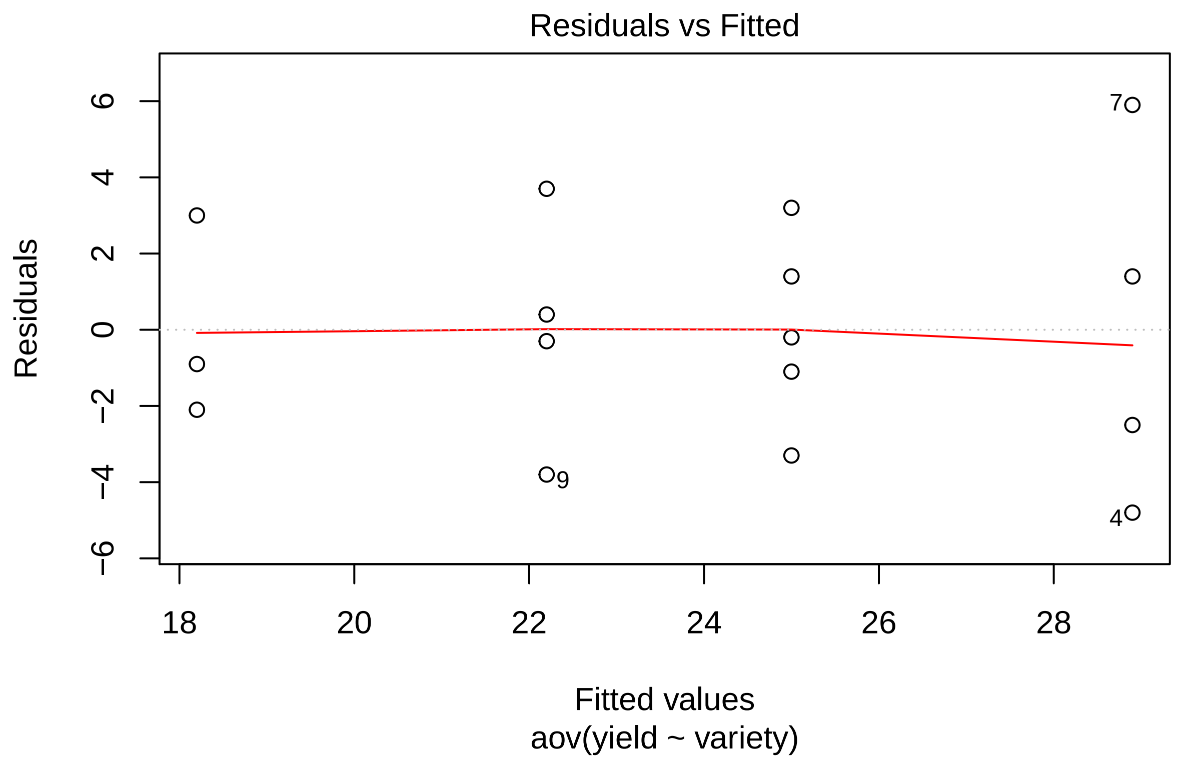
```
model = aov(yield ~ variety, data = data)
anova(model)
```

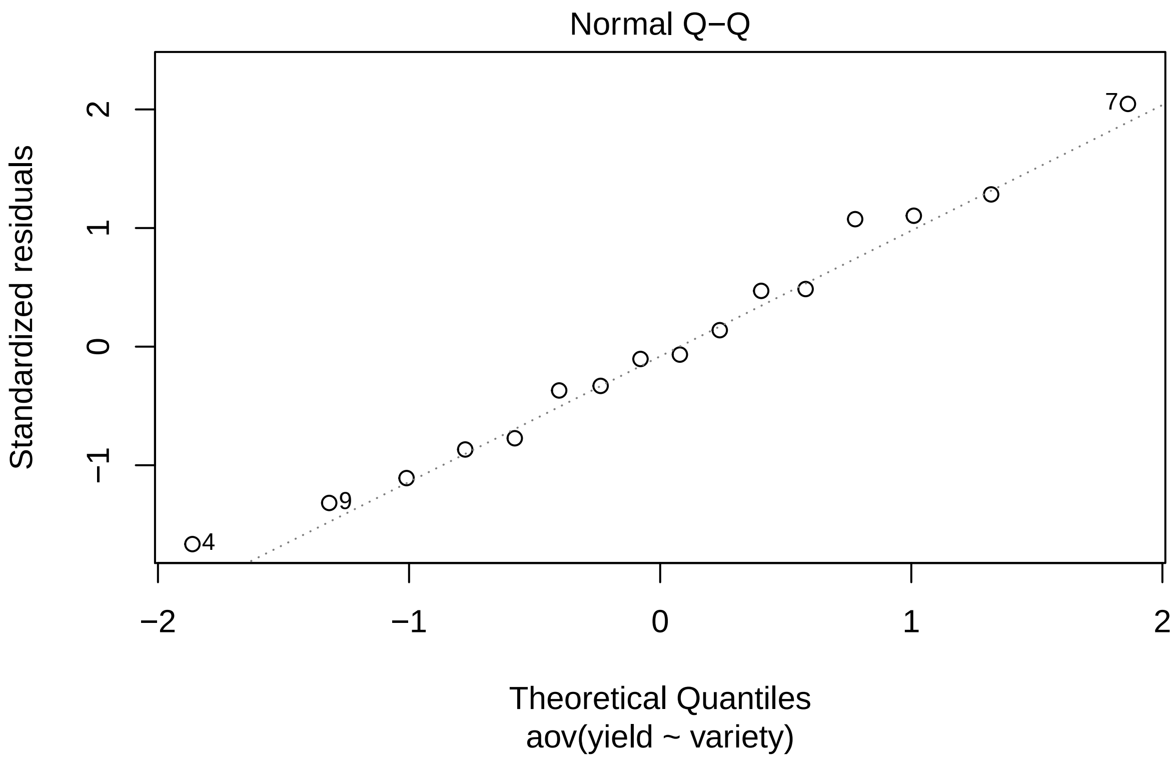
```
## Analysis of Variance Table
##
## Response: yield
##           Df Sum Sq Mean Sq F value   Pr(>F)
## variety    3 214.92  71.640  6.4638 0.007499 **
## Residuals 12 133.00  11.083
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

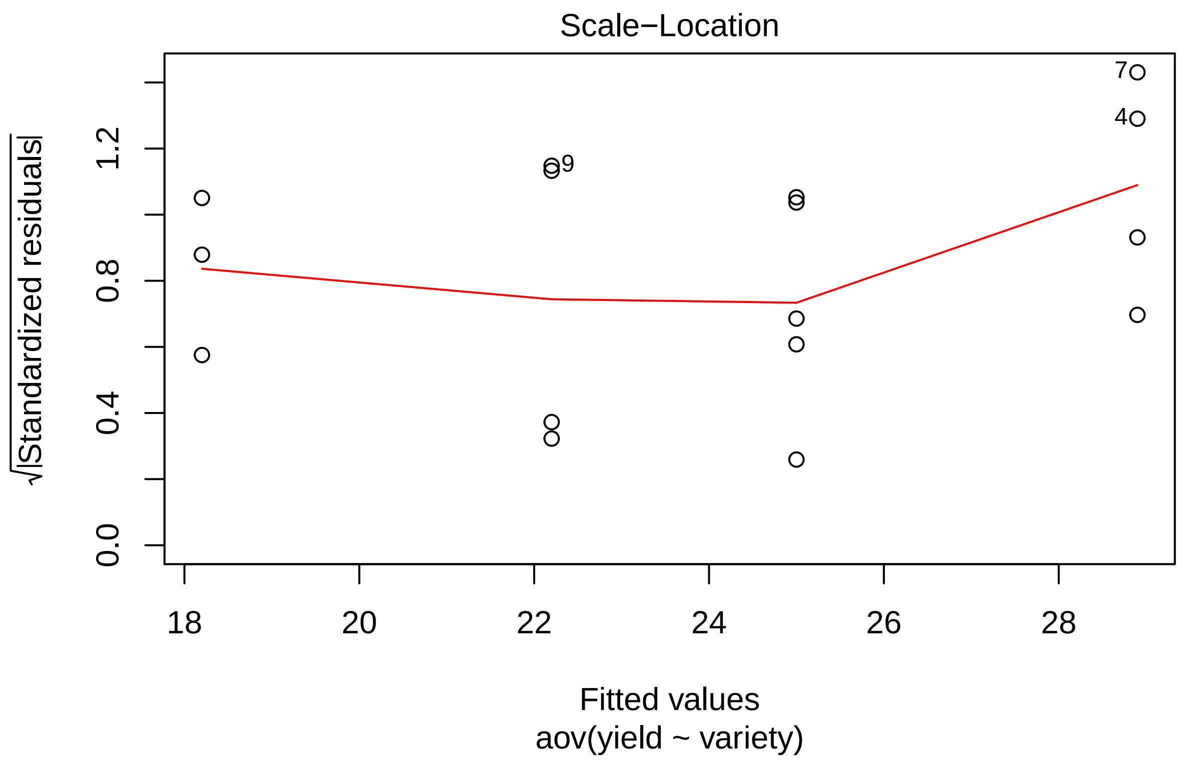
```
boxplot(yield ~ variety, data = data)
```

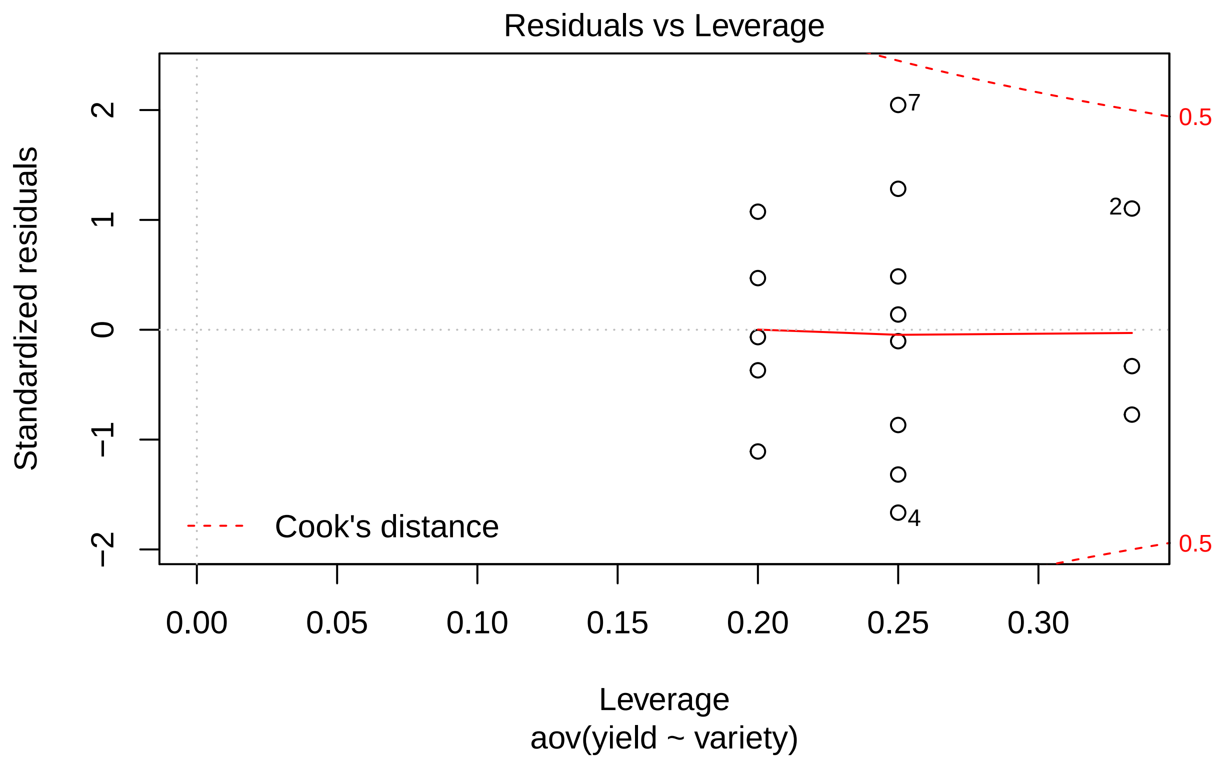


```
plot(model)
```


Residuals vs Fitted
Residuals
6
4
2
0
−2
−4
−6
7
9
4
18
20
22
24
26
28
Fitted values
aov(yield ~ variety)


Normal Q−Q
Standardized residuals
Theoretical Quantiles
aov(yield ~ variety)
−2
−1
0
1
2
4
9
7


Scale−Location
√|Standardized residuals|
0.0
0.4
0.8
1.2
18
20
22
24
26
28
Fitted values
aov(yield ~ variety)
7
4
9



agricolae package

There are several R package for Experimental Design Analysis for Agricultural and Plant Breeding experiments (more *here*). `agricolae` *is by far the most-used package from this task view (status: October 2017).*

Here, we show how to do the RCD analysis using `agricolae`

To install and load `agricolae`

```
#install.packages("agricolae")
library(agricolae)
```

Sampling the treatments

```
variety = c("A","B","C","D")

field_design = design.crd(variety, r=5)
field_design$book
```

```
##    plots r variety
## 1    101 1       D
## 2    102 1       C
## 3    103 1       A
## 4    104 2       A
## 5    105 2       D
## 6    106 2       C
## 7    107 3       D
## 8    108 3       A
## 9    109 1       B
## 10   110 3       C
## 11   111 4       C
## 12   112 4       D
## 13   113 2       B
## 14   114 3       B
## 15   115 5       C
## 16   116 4       A
## 17   117 5       A
## 18   118 4       B
## 19   119 5       D
## 20   120 5       B
```

Data analysis

```
data = data.frame(variety = c("A", "D", "B", "D", "C",
                              "C", "D", "D", "A", "D",
                              "A", "B", "C", "C", "B",
                              "A", "B", "A", "C", "B"),
                  yield = c(22.2, 23.9, 24.1, 21.7, 25.9,
                            18.4, 24.8, 28.2, 17.3, 26.4,
                            21.2, 30.3, 23.2, 21.9, 27.4,
                            25.2, 26.4, 16.1, 22.6, 34.8))

model = aov(yield ~ variety, data = data)
anova(model)
```

```
## Analysis of Variance Table
##
## Response: yield
##           Df Sum Sq Mean Sq F value  Pr(>F)
## variety    3  188.2  62.733  5.6901 0.00756 **
## Residuals 16  176.4  11.025
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
## Coefficient of Variation
cv.model(model)
```

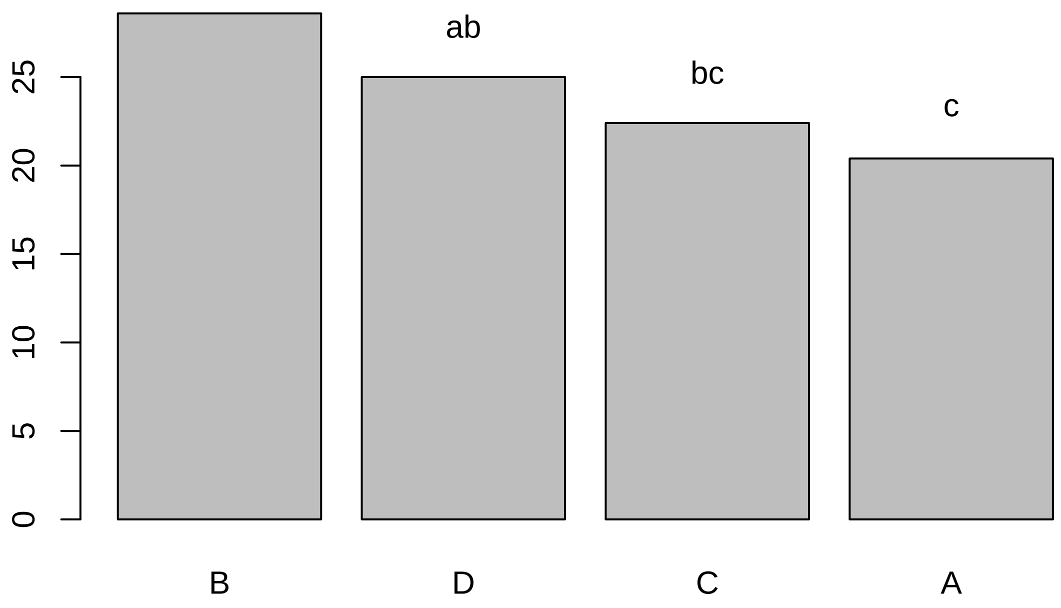
```
## [1] 13.77756
```

```
## Least Significant Difference Analysis
LSD = LSD.test(model, "variety", console = TRUE)
```

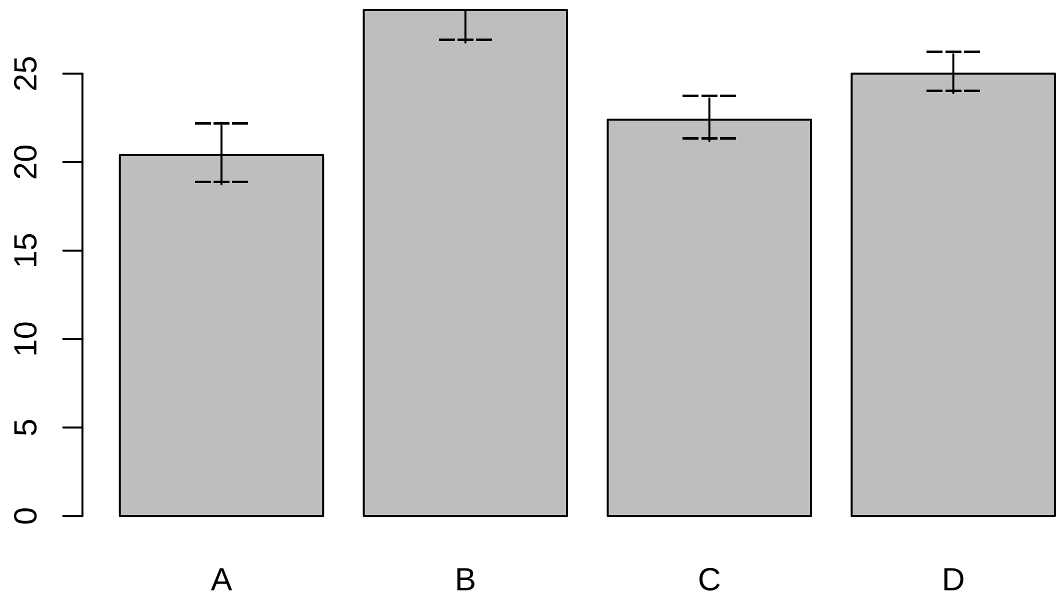
```
##
## Study: model ~ "variety"
##
## LSD t Test for yield
##
## Mean Square Error:  11.025
##
## variety,  means and individual ( 95 %) CI
```

```
##
##   yield      std r      LCL      UCL  Min  Max
## A  20.4 3.708773 5 17.2521 23.5479 16.1 25.2
## B  28.6 4.118859 5 25.4521 31.7479 24.1 34.8
## C  22.4 2.700926 5 19.2521 25.5479 18.4 25.9
## D  25.0 2.466779 5 21.8521 28.1479 21.7 28.2
##
## Alpha: 0.05 ; DF Error: 16
## Critical Value of t: 2.119905
##
## least Significant Difference: 4.451801
##
## Treatments with the same letter are not significantly different.
##
##   yield groups
## B  28.6      a
## D  25.0     ab
## C  22.4     bc
## A  20.4      c
```

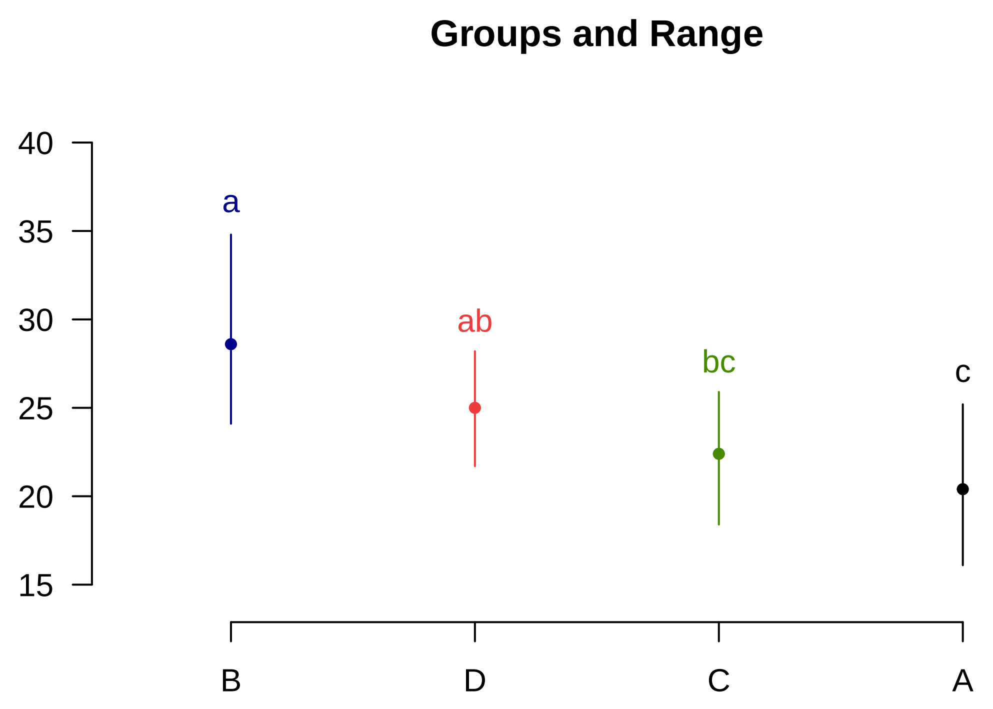
```r
bar.group(LSD$group)
```



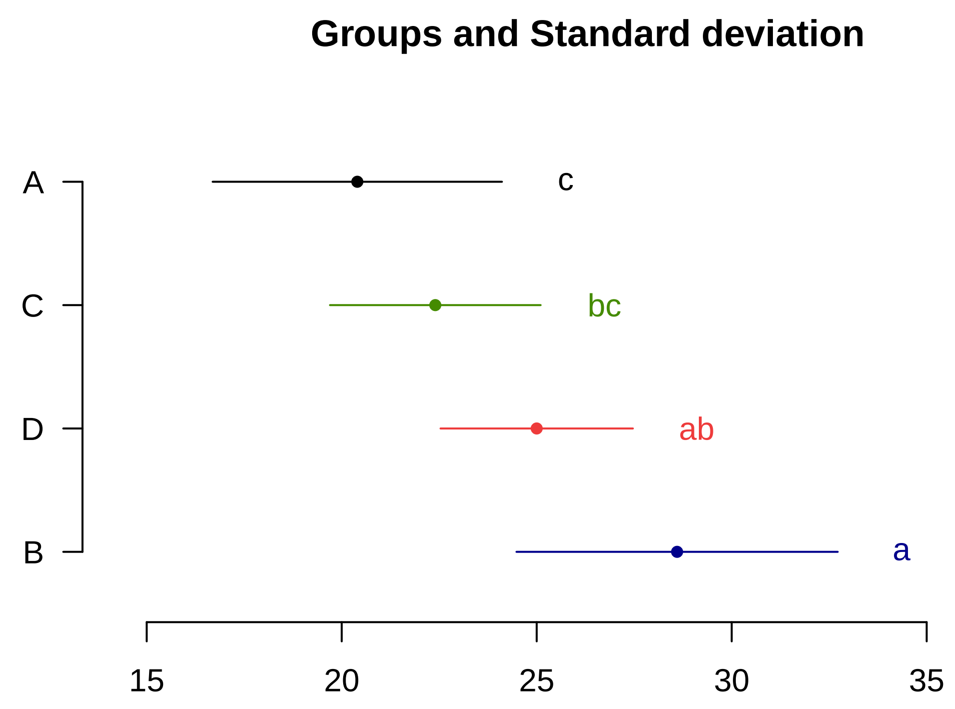
```r
bar.err(LSD$means)
```



```r
plot(LSD, variation="range",las=1)
```



```r
plot(LSD, horiz=TRUE, variation="SD",las=1)
```



```r
## Tukey's (HSD) Test
HSD.test(model, "variety", console=TRUE)
```

```
##
## Study: model ~ "variety"
##
## HSD Test for yield
##
## Mean Square Error:  11.025
##
## variety,  means
##
##   yield      std r  Min  Max
## A  20.4 3.708773 5 16.1 25.2
## B  28.6 4.118859 5 24.1 34.8
## C  22.4 2.700926 5 18.4 25.9
## D  25.0 2.466779 5 21.7 28.2
##
## Alpha: 0.05 ; DF Error: 16
## Critical Value of Studentized Range: 4.046093
##
## Minimun Significant Difference: 6.008142
##
## Treatments with the same letter are not significantly different.
##
##   yield groups
## B  28.6       a
## D  25.0      ab
## C  22.4       b
## A  20.4       b
```

Chapter 10: The Randomized Block Design

10.1 Example

Below, we build the data frame of Example 10.1. Here we are reading the data by columns.

```
example10.1 = data.frame(variety = c("V1", "V1", "V1",
                                     "V2", "V2", "V2",
                                     "V3", "V3", "V3",
                                     "V4", "V4", "V4"),
                         block = c("B1", "B2", "B3",
                                   "B1", "B2", "B3",
                                   "B1", "B2", "B3",
                                   "B1", "B2", "B3"),
                         yield = c(7.4, 6.5, 5.6,
                                   9.8, 6.8, 6.2,
                                   7.3, 6.1, 6.4,
                                   9.5, 8.0, 7.4))
## Visualing the data frame
print(example10.1)
```

```
##    variety block yield
## 1       V1    B1   7.4
## 2       V1    B2   6.5
## 3       V1    B3   5.6
## 4       V2    B1   9.8
## 5       V2    B2   6.8
## 6       V2    B3   6.2
## 7       V3    B1   7.3
## 8       V3    B2   6.1
## 9       V3    B3   6.4
## 10      V4    B1   9.5
## 11      V4    B2   8.0
## 12      V4    B3   7.4
```

Checking the data frame structure, sums, block and variety totals

```
str(example10.1)
```

```
## 'data.frame':    12 obs. of  3 variables:
##  $ variety: Factor w/ 4 levels "V1","V2","V3",..: 1 1 1 2 2 2 3 3 3 4 ...
##  $ block  : Factor w/ 3 levels "B1","B2","B3": 1 2 3 1 2 3 1 2 3 1 ...
##  $ yield  : num  7.4 6.5 5.6 9.8 6.8 6.2 7.3 6.1 6.4 9.5 ...
```

```
#apply the function sum in yield by block
tapply(example10.1$yield, example10.1$block, sum)
```

```
##   B1   B2   B3
## 34.0 27.4 25.6
```

```
#apply the function mean in yield by block
tapply(example10.1$yield, example10.1$block, mean)
```

```
##   B1   B2   B3
## 8.50 6.85 6.40
```

```
#apply the function sum in yield by variety
tapply(example10.1$yield, example10.1$variety, sum)
```

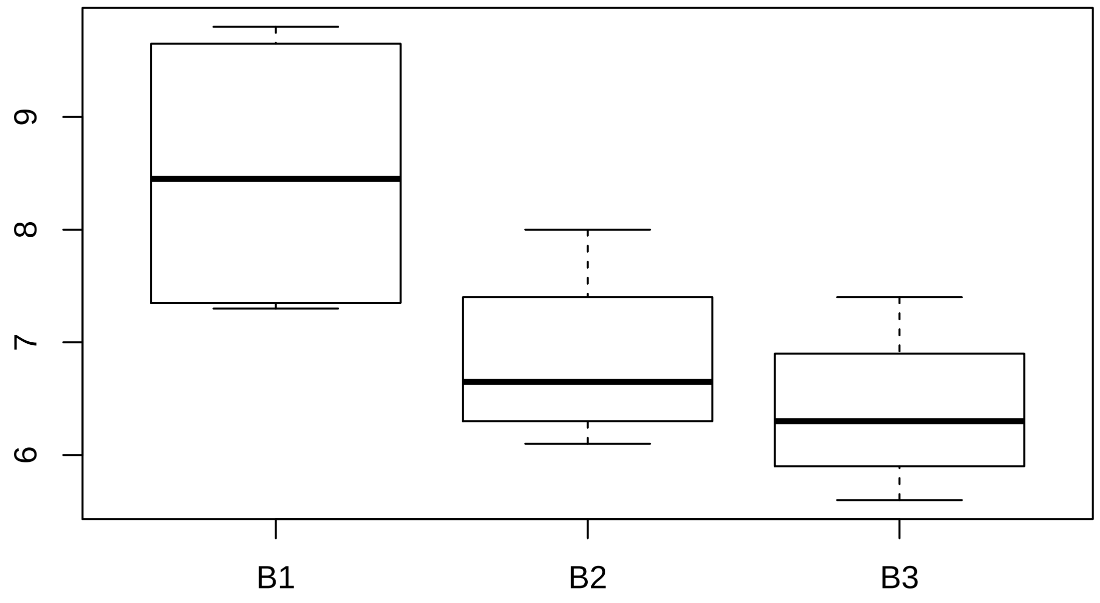
```
##   V1   V2   V3   V4
## 19.5 22.8 19.8 24.9
```

```
#apply the function mean in yield by variety
tapply(example10.1$yield, example10.1$variety, mean)
```

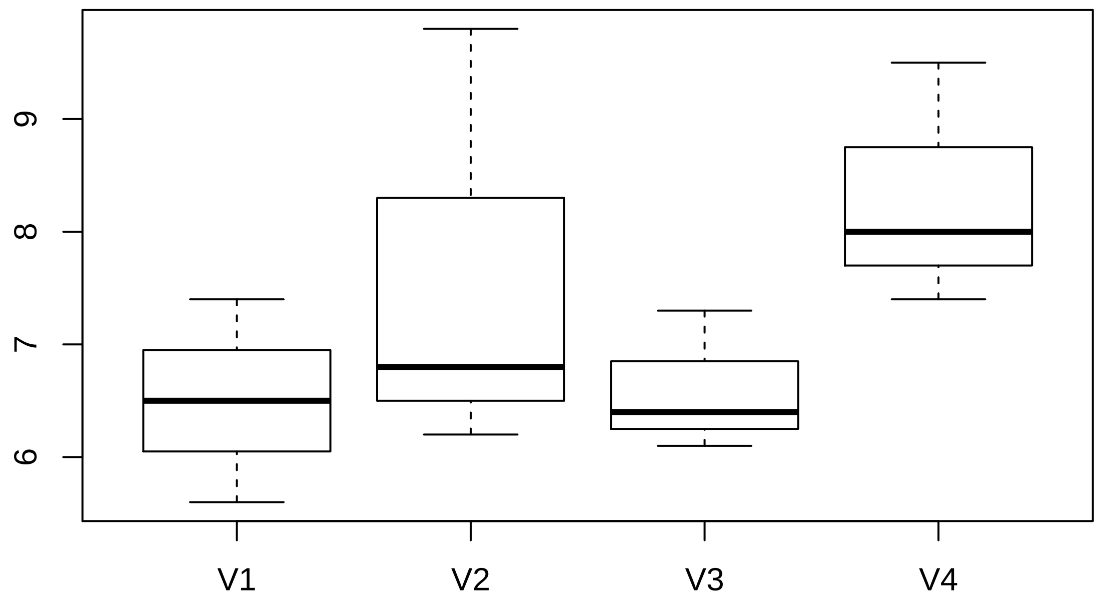
```
##  V1  V2  V3  V4
## 6.5 7.6 6.6 8.3
```

Descriptive plots

```
boxplot( example10.1$yield ~ example10.1$block)
```



```
boxplot( example10.1$yield ~ example10.1$variety)
```



10.2 Analysis ignoring blocks

```
model10.2 = aov(yield ~ variety, data = example10.1)
anova(model10.2)
```

```
## Analysis of Variance Table
##
## Response: yield
##           Df Sum Sq Mean Sq F value Pr(>F)
## variety    3   6.63  2.2100  1.4516 0.2987
## Residuals  8  12.18  1.5225
```

10.3 The analysis including blocks

```
model10.3 = aov(yield ~ block + variety, data = example10.1)
anova(model10.3)
```

```
## Analysis of Variance Table
##
## Response: yield
##           Df Sum Sq Mean Sq F value   Pr(>F)
## block      2   9.78    4.89  12.225 0.007651 **
## variety    3   6.63    2.21   5.525 0.036730 *
## Residuals  6   2.40    0.40
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Residuals and Fitted values

```r
model10.3$residuals
```

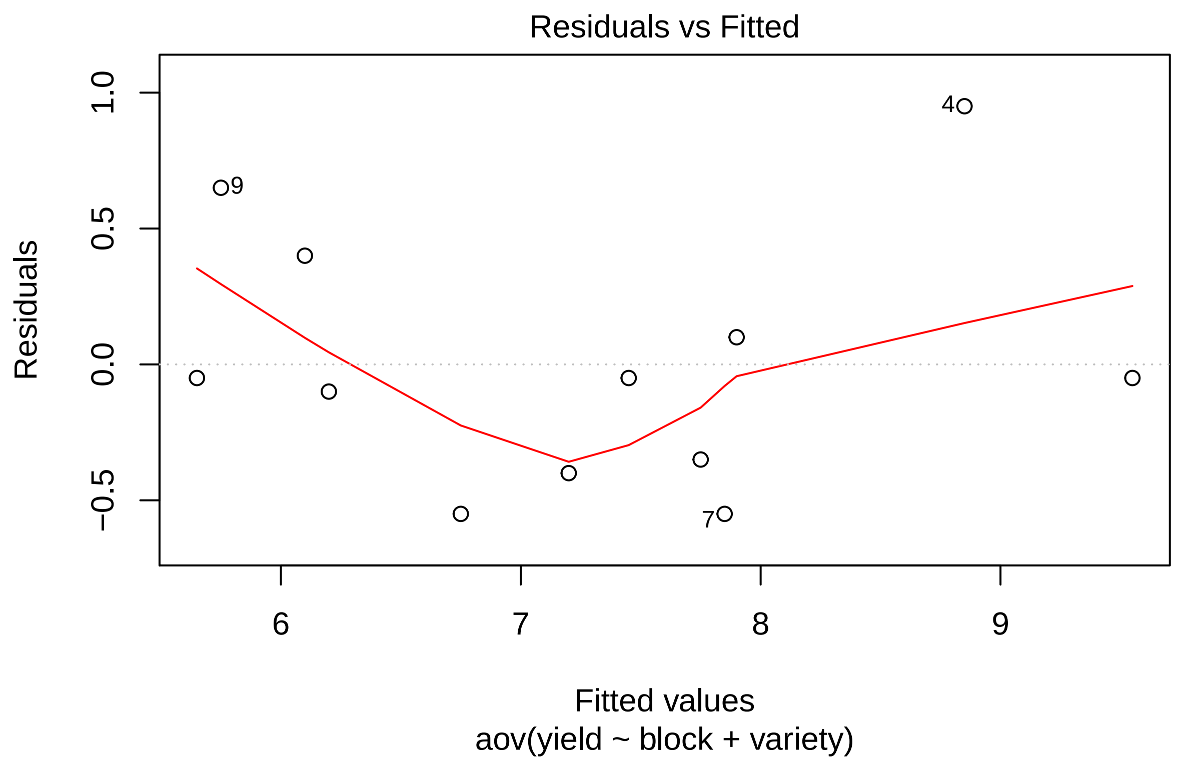
```
##     1     2     3     4     5     6     7     8     9    10    11    12
## -0.35  0.40 -0.05  0.95 -0.40 -0.55 -0.55 -0.10  0.65 -0.05  0.10 -0.05
```

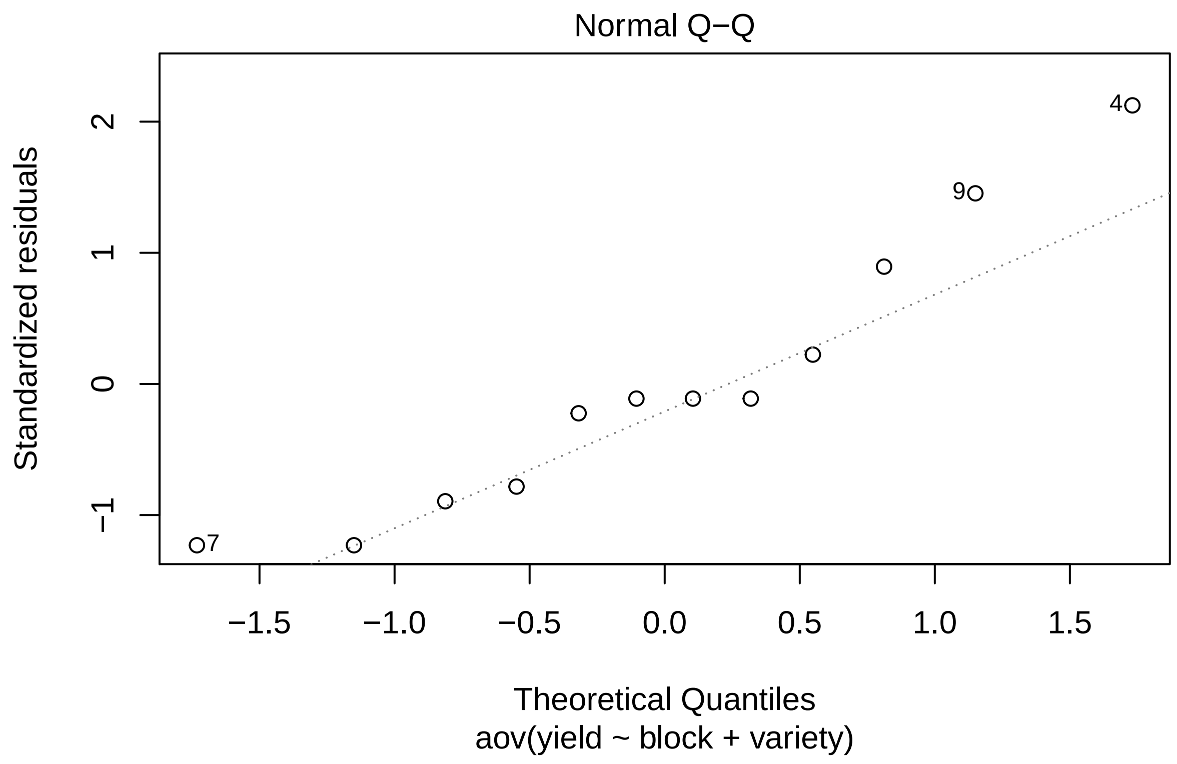
```r
model10.3$fitted.values
```

```
##    1    2    3    4    5    6    7    8    9   10   11   12
## 7.75 6.10 5.65 8.85 7.20 6.75 7.85 6.20 5.75 9.55 7.90 7.45
```

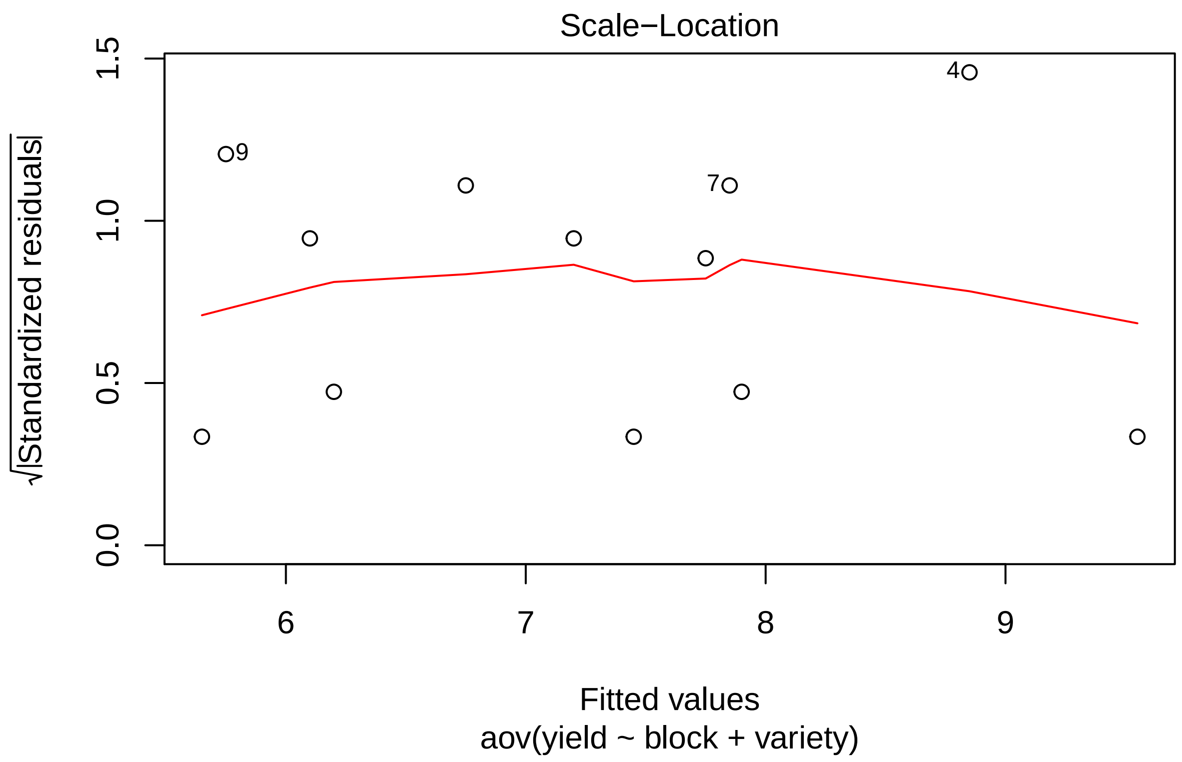
Diagnostics Plots

```r
plot(model10.3)
```

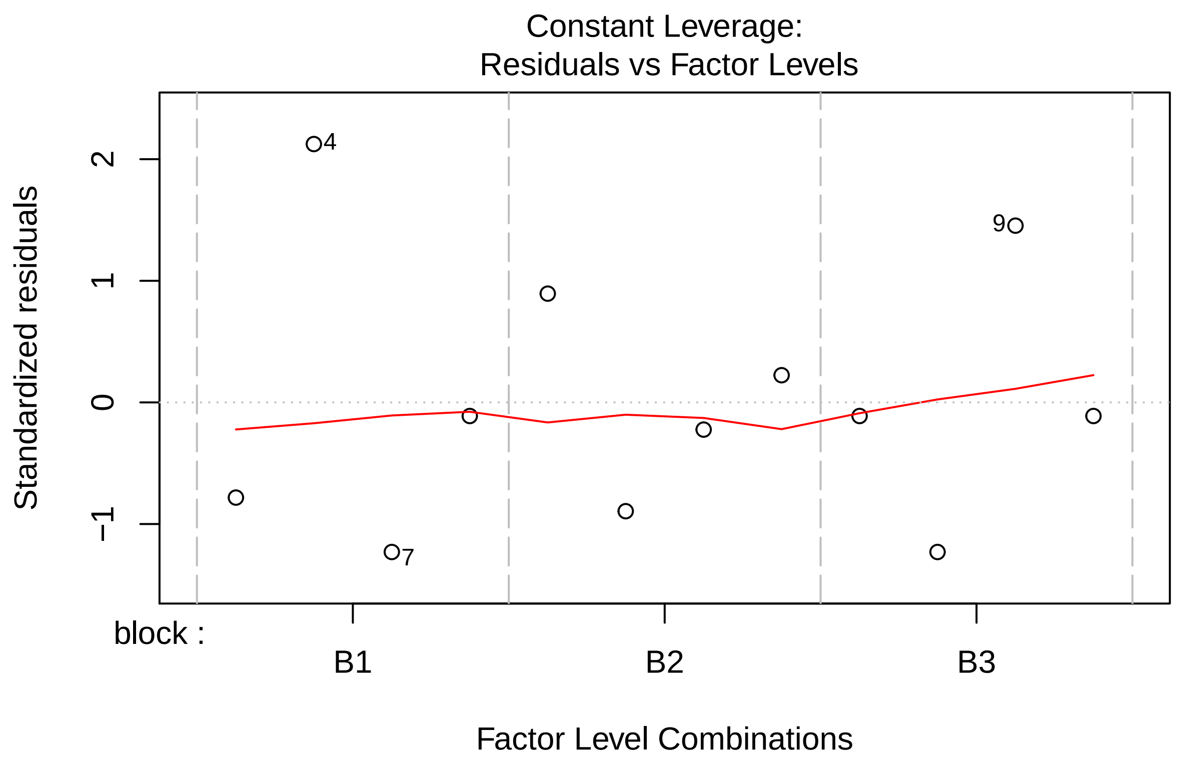




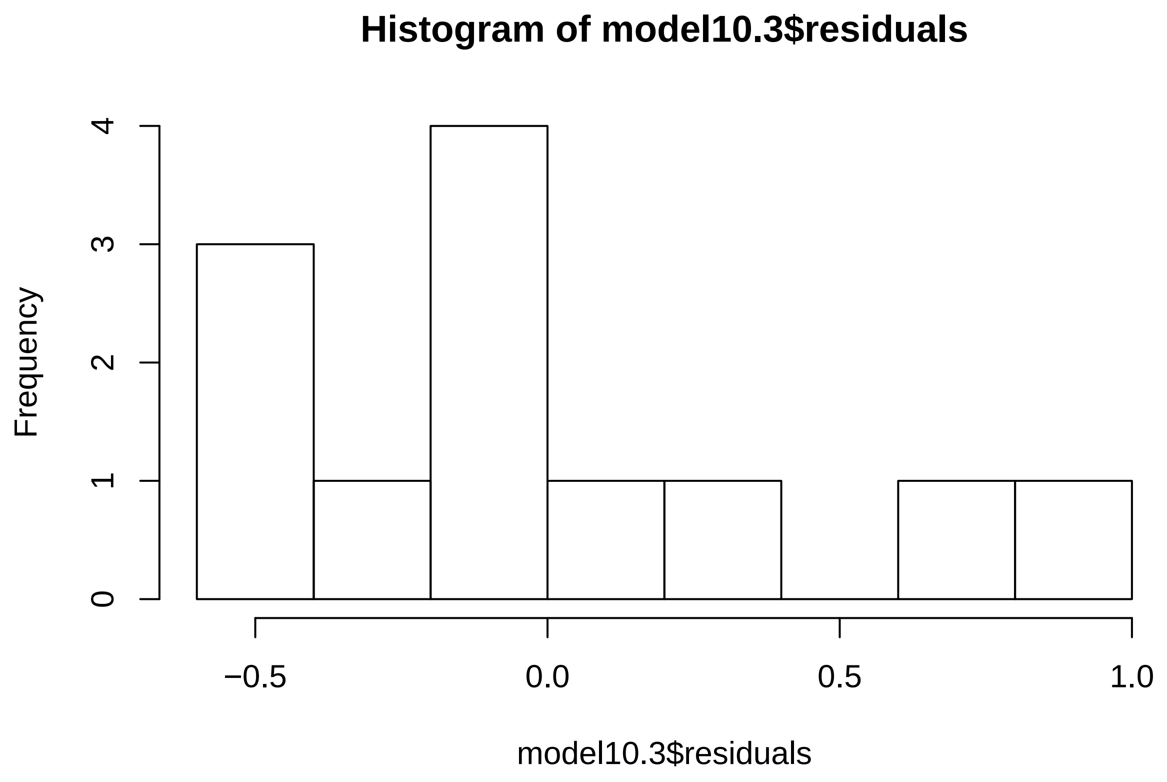
Normal Q−Q
Standardized residuals
2
1
0
−1
4
9
7
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
Theoretical Quantiles
aov(yield ~ block + variety)



Scale−Location
√|Standardized residuals|
0.0
0.5
1.0
1.5
6
7
8
9
Fitted values
aov(yield ~ block + variety)
9
7
4



```r
hist(model10.3$residuals,breaks = 10)
```



10.8 Comparison of treatment means

10.8.1 LSD Analysis and Confidence Intervals

With agricolae

```r
library(agricolae)
LSD.test(model10.3, "variety", console=TRUE)
```

```
##
## Study: model10.3 ~ "variety"
##
## LSD t Test for yield
##
## Mean Square Error:  0.4
##
## variety,  means and individual ( 95 %) CI
##
##    yield       std r      LCL      UCL Min Max
## V1   6.5 0.9000000 3 5.606514 7.393486 5.6 7.4
## V2   7.6 1.9287302 3 6.706514 8.493486 6.2 9.8
## V3   6.6 0.6244998 3 5.706514 7.493486 6.1 7.3
## V4   8.3 1.0816654 3 7.406514 9.193486 7.4 9.5
##
## Alpha: 0.05 ; DF Error: 6
## Critical Value of t: 2.446912
##
## least Significant Difference: 1.26358
##
## Treatments with the same letter are not significantly different.
##
##    yield groups
## V4   8.3      a
## V2   7.6     ab
## V3   6.6      b
## V1   6.5      b
```

Standard Errors and Confidence Intervals (by "hand""")

```
## Extracting RMS
anova(model10.3)
```

```
## Analysis of Variance Table
##
## Response: yield
##           Df Sum Sq Mean Sq F value   Pr(>F)
## block      2   9.78    4.89  12.225 0.007651 **
## variety    3   6.63    2.21   5.525 0.036730 *
## Residuals  6   2.40    0.40
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
## The RMS is the Residual Mean Square error,
## In our ANOVA table, RMS is located in the 3rd row and 3rd column
RMS = anova(model10.3)[3,3]
print(RMS)
```

```
## [1] 0.4
```

```
r = 3 #number of reps (blocks)

SEM = sqrt(RMS/r)
print(SEM)
```

```
## [1] 0.3651484
```

```
lower.t = qt(0.025, 6)
upper.t = qt(0.975, 6)

variety.mean = tapply(example10.1$yield, example10.1$variety, mean)

lower.CI = variety.mean + lower.t*SEM
upper.CI = variety.mean + upper.t*SEM

CI.variety = data.frame(mean = variety.mean,
                        lower = lower.CI,
                        upper = upper.CI)
```