CH3: Data Description

We use crime rate data (Example 3.14) from Ott & Longnecker to illustrate some basic summary statistics and graphs.

Using Department of Justice Data from 2000, the crime rate in 90 US cities were obtained. (Note that we have no indication of how these cities were selected.) Rates represent the number of violent crimes per 100,000 inhabitants, rounded to the nearest whole number.

```
CrimeData <- read.csv("~/Google Drive/My Drive/teaching/CSU/STAR511 2022/week 2/CH3_Crime.csv")
str(CrimeData)
```

```
## 'data.frame':    90 obs. of  1 variable:
##  $ Crime: int  498 676 344 368 772 497 415 925 555 260 ...
```

1 Summary Statistics

```
summary(CrimeData$Crime)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   140.0   398.5   497.5   541.1   659.2  1094.0
```

```
mean(CrimeData$Crime)
```

```
## [1] 541.1
```

```
sd(CrimeData$Crime)
```

```
## [1] 217.85
```

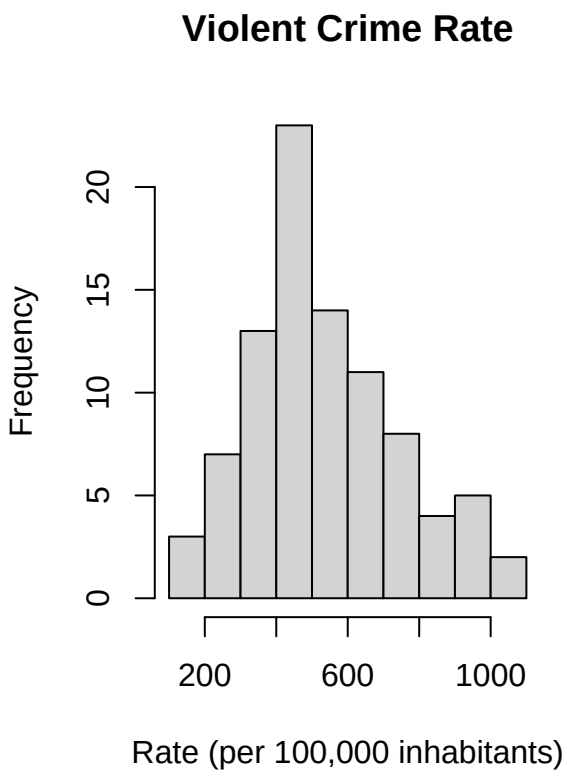
```
median(CrimeData$Crime)
```

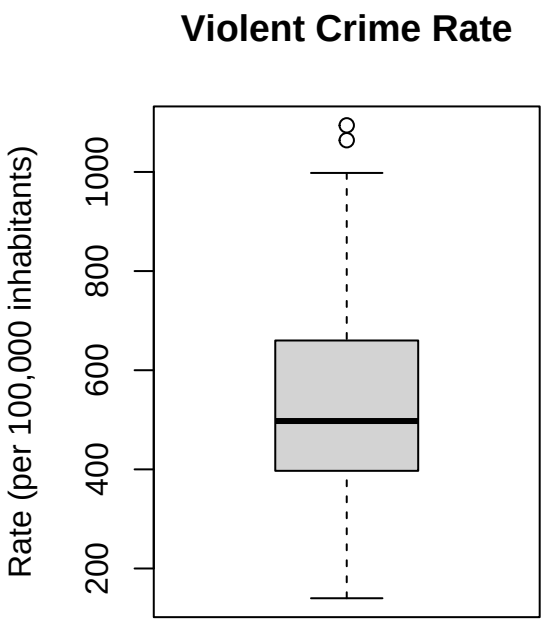
```
## [1] 497.5
```

2 Summary Graphs (Base R)

We see evidence of right skew, but not too strong.

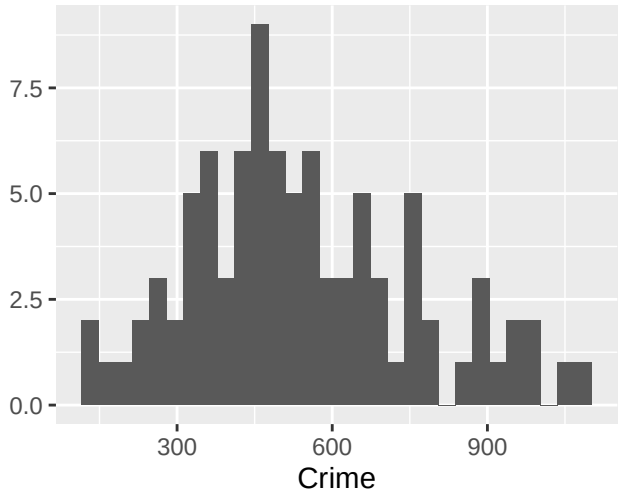
```
par(mfrow=c(1,2))
hist(CrimeData$Crime, main = "Violent Crime Rate", xlab = "Rate (per 100,000 inhabitants)")
boxplot(CrimeData$Crime, main = "Violent Crime Rate", ylab = "Rate (per 100,000 inhabitants)")
```



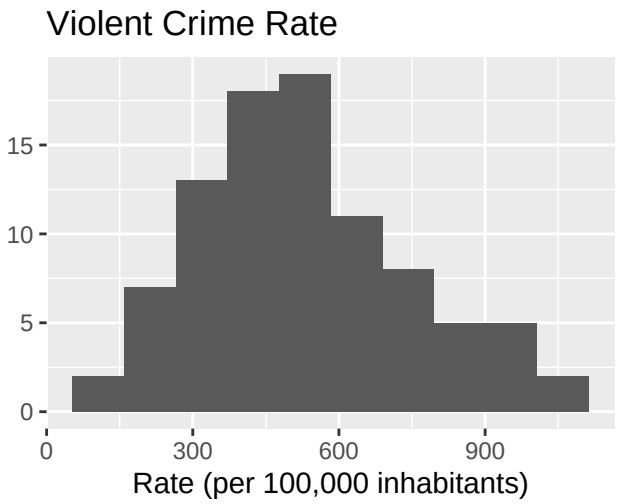


3 Summary Graphs (tidyverse)

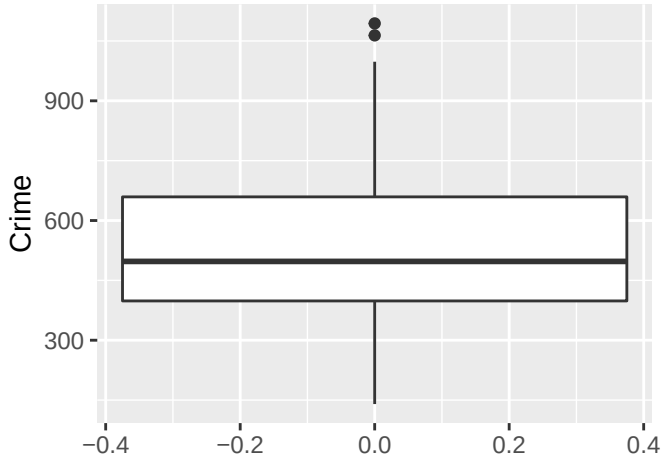
```
library(tidyverse)
qplot(x = Crime, data = CrimeData)
```




```
qplot(x = Crime, data = CrimeData) +
  stat_bin(bins = 10) +
  xlab("Rate (per 100,000 inhabitants)") +
  ggtitle("Violent Crime Rate")
```




```
qplot(y = Crime, geom = "boxplot", data = CrimeData)
```



Normal Probability Examples

Ex 1: P(Z ≤ 1.31)

```
pnorm(1.31)
```

```
## [1] 0.9049021
```

Ex 2: P(Z > 1.72) = P(Z ≥ 1.72)

```
1-pnorm(1.72)
```

```
## [1] 0.04271622
```

```
pnorm(1.72, lower.tail = FALSE) #Same Answer
```

```
## [1] 0.04271622
```

Not looking for Probability, looking for a value z
Ex 3: Find z, such that P(Z > z) = 0.95

```
qnorm(1-.95)
```

```
## [1] -1.644854
```

```
qnorm(.05)
```

```
## [1] -1.644854
```

```
qnorm(.95, lower.tail = F)
```

```
## [1] -1.644854
```

Not Just Standard Normal!
Ex 4: Y ~ N(5,2), Find P(Y ≤ 8) = P(Y < 8)

```
y=8   # observed value for Y
mu = 5  #population mean
sigma = 2  #population std dev
z = (y- mu)/sigma   #the formula
pnorm(z)
```

```
## [1] 0.9331928
```

```
pnorm((8-5)/2)  #same without separate line for z-formula
```

```
## [1] 0.9331928
```

```
pnorm(8, mean = 5, sd =2)  #specify which normal distn in arguments
```

```
## [1] 0.9331928
```

Ex 5: Find y such that P(Y ≤ y) = .975

```
z = qnorm(.975)   #P(Z<z) = .975
mu = 5  #population mean
sigma = 2  #population std dev
```

```r
y = mu + z*sigma   #z-formula solving for y
y
```

```
## [1] 8.919928
```

```r
sigma*qnorm(.975) + mu  #same without separate line for z-formula
```

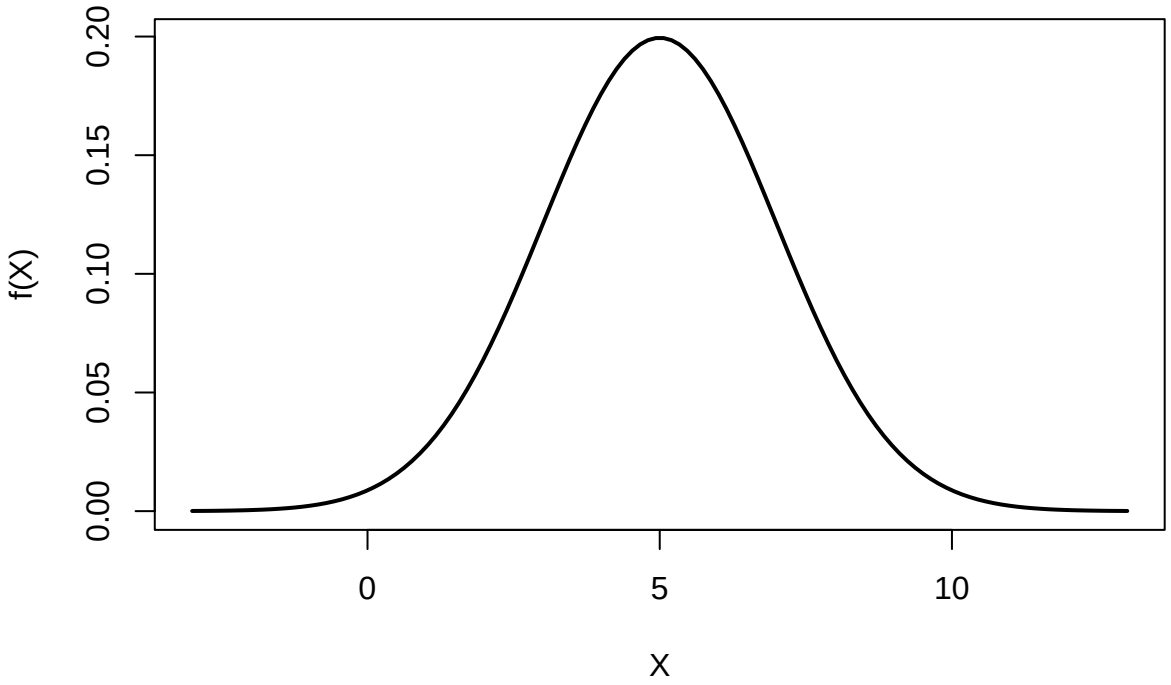
```
## [1] 8.919928
```

```r
qnorm(.975, mean = 5, sd =2)  #specify which normal distn in arguments
```

```
## [1] 8.919928
```

Plot N (5,2)

```r
x <- seq(-3,13, length.out = 120)  #values for x's
plot(x,dnorm(x, mean = 5, sd = 2), xlab = "X", ylab= "f(X)", type = "l", lwd = 2)
```




```r
#abline(h=0, col= "gray", lwd = 2)
```

CH5.1: Confidence Interval for Single Mean

We use the t-based confidence interval to make inference for a single population mean. Requires the assumption of normality or large sample size.

1 Cattle Data

Hormone levels were measured from a random sample of n = 20 cows.

```
CattleData <- read.csv("CH5_Cattle.csv")
str(CattleData)
```

```
## 'data.frame':    20 obs. of  1 variable:
##  $ Hormone: num  17.2 16.7 14.5 10.3 18.2 ...
```

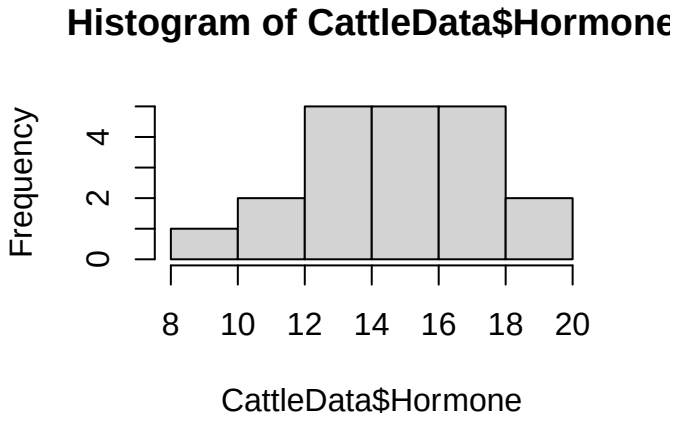
```
mean(CattleData$Hormone)
```

```
## [1] 14.62
```

```
sd(CattleData$Hormone)
```

```
## [1] 2.730506
```

```
hist(CattleData$Hormone)
```



2 CI using t.test

The 95% confidence interval is given by default.
To change this, can use the conf.level option.

Note that a test (p-value) is returned by default.
We will discuss the one-sample t-test in the next group of notes.

```
t.test(CattleData$Hormone)
```

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

```
## data:  CattleData$Hormone
## t = 23.945, df = 19, p-value = 1.18e-15
## alternative hypothesis: true mean is not equal to 0
## 95 percent confidence interval:
##  13.34208 15.89792
## sample estimates:
## mean of x
##     14.62
```

3 CI "by hand" (for illustration)

We use the mean and sd shown above.
Recall n = 20. Hence df = 19.
We construct a 95% CI, corresponding to alpha = 0.05.

```
14.62 - qt(0.975, df = 19)*2.73/sqrt(20)
```

```
## [1] 13.34232
```

```
14.62 + qt(0.975, df = 19)*2.73/sqrt(20)
```

```
## [1] 15.89768
```

CH5.3: One sample t-test

1 One-sample t-test

We use the one-sample t-test to make inference for a single population mean. Requires the assumption of normality or large sample size.

1.1 Cattle Data

Hormone levels were measured from a random sample of n = 20 cows.

```
CattleData <- read.csv("CH5_Cattle.csv")
str(CattleData)
```

```
## 'data.frame':    20 obs. of  1 variable:
##  $ Hormone: num  17.2 16.7 14.5 10.3 18.2 ...
```

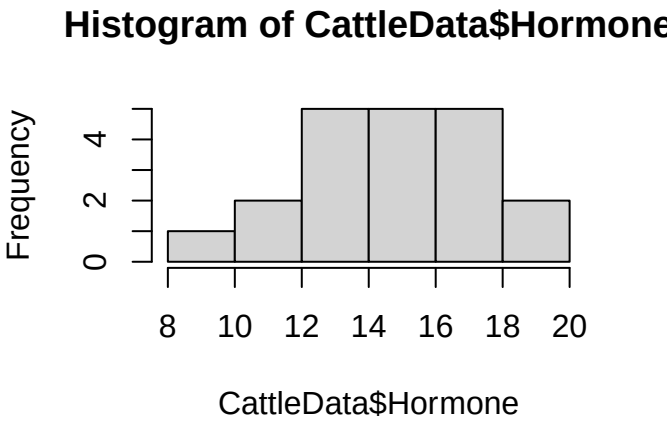
```
mean(CattleData$Hormone)
```

```
## [1] 14.62
```

```
sd(CattleData$Hormone)
```

```
## [1] 2.730506
```

```
hist(CattleData$Hormone)
```



1.2 Two-sided Test

Here we do a two-sided test of H0: $\mu = 12$ vs HA: $\mu \neq 12$.
Remember that the hypotheses should be motivated by the research question and can (should!) be specified before looking at the data.

```
t.test(CattleData$Hormone, mu = 12)
```

```
##
##  One Sample t-test
##
## data:  CattleData$Hormone
```

```
## t = 4.2911, df = 19, p-value = 0.0003943
## alternative hypothesis: true mean is not equal to 12
## 95 percent confidence interval:
##  13.34208 15.89792
## sample estimates:
## mean of x
##     14.62
```

1.3 One-sided Test

Now we do a one-sided test of H0: $\mu \leq 12$ vs HA: $\mu > 12$.
Remember that a two-sided test should be used "by default". One-sided tests should not be used unless there is some compelling reason.

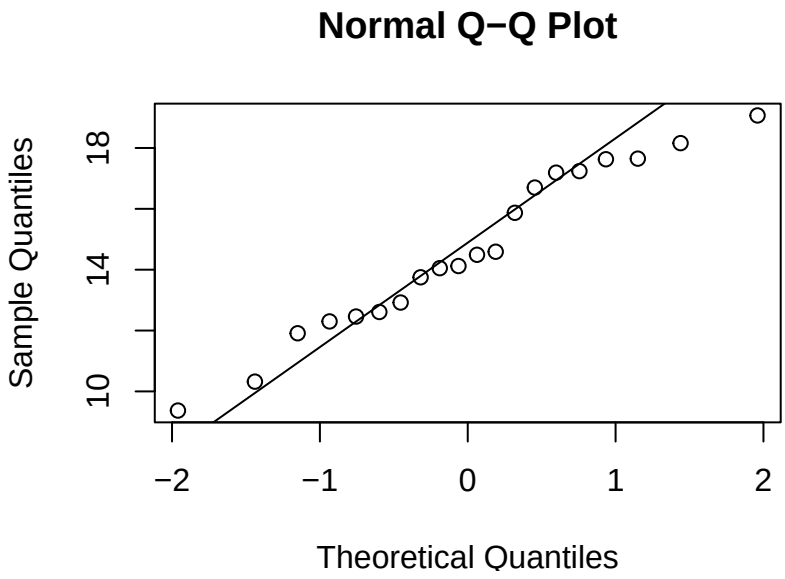
```
t.test(CattleData$Hormone, mu = 12, alternative =  "greater")
```

```
##
##  One Sample t-test
##
## data:  CattleData$Hormone
## t = 4.2911, df = 19, p-value = 0.0001971
## alternative hypothesis: true mean is greater than 12
## 95 percent confidence interval:
##  13.56426      Inf
## sample estimates:
## mean of x
##     14.62
```

2 Evaluating Normality

The QQplot is a graphical tool for assessing normality.
If the plot is roughly a straight line, it supports the idea that the data came from the normal distribution.

```
qqnorm(CattleData$Hormone)
qqline(CattleData$Hormone)
```



For both SW and KS tests, large p-values support normality.

```
shapiro.test(CattleData$Hormone)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  CattleData$Hormone
## W = 0.96181, p-value = 0.5807
```

```
ks.test(CattleData$Hormone, "pnorm", mean(CattleData$Hormone), sd(CattleData$Hormone) )
```

```
##
##  Exact one-sample Kolmogorov-Smirnov test
##
## data:  CattleData$Hormone
## D = 0.1269, p-value = 0.8647
## alternative hypothesis: two-sided
```

3 tidyverse (Optional)

Summary statistics, tidy output and summary plot using tidyverse and broom.
Recall that the %>% "pipe operator" is a feature of tidyverse. With ggplot2 we can build plots in layers.

```
library(tidyverse)
library(broom)
SumStats <- CattleData %>%
            summarise(n = n(),
                      mean = mean(Hormone),
                      sd = sd(Hormone),
                      SE = sd/sqrt(n))
SumStats
```

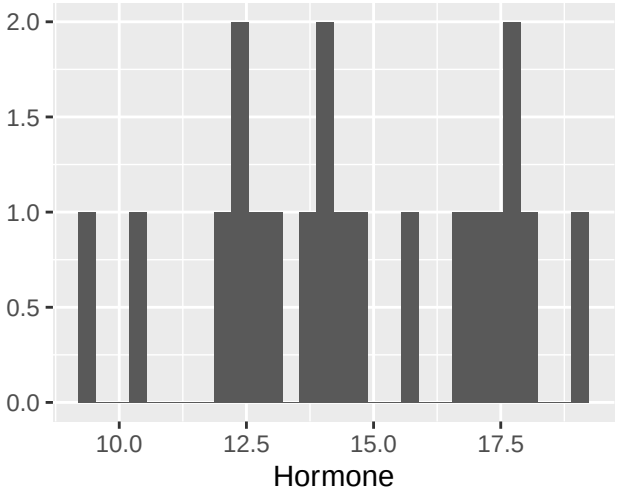
```
##    n  mean       sd        SE
## 1 20 14.62 2.730506 0.6105597
```

```
tidy(t.test(CattleData$Hormone,mu = 12))
```

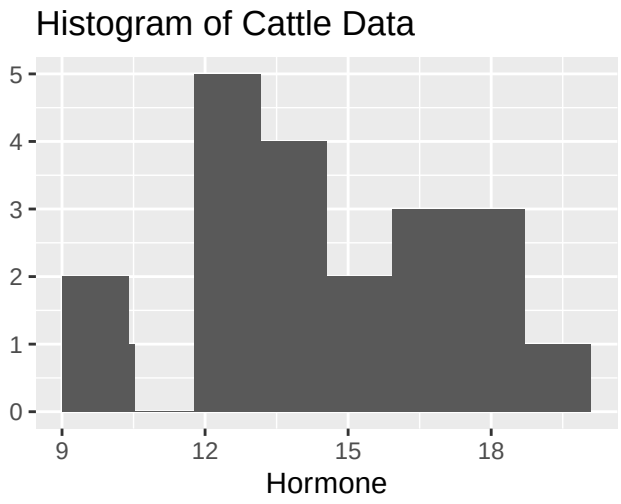
```
## # A tibble: 1 x 8
##   estimate statistic  p.value parameter conf.low conf.high method    alternative
##      <dbl>     <dbl>    <dbl>     <dbl>    <dbl>     <dbl> <chr>     <chr>
## 1     14.6      4.29 0.000394        19     13.3      15.9 One Samp~ two.sided
```

```
qplot(Hormone, data = CattleData)
```

```
## Warning: `qplot()` was deprecated in ggplot2 3.4.0.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```




```
qplot(Hormone, data = CattleData) +
  stat_bin(bins = 8) +
  ggtitle("Histogram of Cattle Data")
```


Histogram of Cattle Data
5
4
3
2
1
0
9
12
15
18
Hormone

4 Simulation for two-sided test and CI

For Illustration: This is not a basic data analysis example! We simulate data where the truth is known. Specifically, we generate data from the standard normal distribution (mean = 0, standard deviation = 1) and look at confidence intervals and hypothesis tests.
As a secondary goal of this example, we will illustrate an approach for running the same analysis multiple times.

4.1 Simulate data from Standard Normal

Hence μ = true mean = 0. In other words H0: $\mu = 0$ is true.

We use rnorm to simulate data from 1000 samples of size n = 25 using the standard normal distribution. set.seed() is used so that we can recreate the same results.

```
library(tidyverse)
library(broom)
set.seed(15672)
SimData <- data.frame(SampleID = rep(seq(1, 1000), 25), Y = rnorm(25000))
str(SimData)
```

```
## 'data.frame':    25000 obs. of  2 variables:
##  $ SampleID: int  1 2 3 4 5 6 7 8 9 10 ...
##  $ Y       : num  -1.063 1.922 -0.045 0.133 -1.187 ...
```

```
summary(SimData)
```

```
##     SampleID            Y
##  Min.   :   1.0   Min.   :-4.249126
##  1st Qu.: 250.8   1st Qu.:-0.679455
##  Median : 500.5   Median :-0.012982
##  Mean   : 500.5   Mean   :-0.006655
##  3rd Qu.: 750.2   3rd Qu.: 0.662290
##  Max.   :1000.0   Max.   : 4.085963
```

4.2 Look at t.test for SampleID 1

```
temp <- with(t.test(Y), data = subset(SimData, SampleID==1) )
temp
```

```
##
##  One Sample t-test
##
## data:  Y
## t = -1.8409, df = 24, p-value = 0.07804
## alternative hypothesis: true mean is not equal to 0
## 95 percent confidence interval:
##  -0.86032174  0.04914038
## sample estimates:
##  mean of x
## -0.4055907
```

```
tidy(temp)
```

```
## # A tibble: 1 x 8
##   estimate statistic p.value parameter conf.low conf.high method     alternative
##      <dbl>     <dbl>   <dbl>     <dbl>    <dbl>     <dbl> <chr>      <chr>
```

```
## 1   -0.406     -1.84  0.0780        24   -0.860    0.0491 One Sampl~ two.sided
```

```
summary(temp)
```

```
##             Length Class  Mode
## statistic   1      -none- numeric
## parameter   1      -none- numeric
## p.value     1      -none- numeric
## conf.int    2      -none- numeric
## estimate    1      -none- numeric
## null.value  1      -none- numeric
## stderr      1      -none- numeric
## alternative 1      -none- character
## method      1      -none- character
## data.name   1      -none- character
```

```
temp$statistic
```

```
##         t
## -1.840864
```

```
temp$p.value
```

```
## [1] 0.07803791
```

4.3 Now run t.test and CI for each SampleID

Use do from summarise from dplyr to run t.test for each SampleID.

```
OutData <- SimData %>%
           group_by(SampleID) %>%
           summarise(statistic = t.test(Y)$statistic,
                     p.value = t.test(Y)$p.value,
                     conf.low = t.test(Y)$conf.int[1],
                     conf.high = t.test(Y)$conf.int[2])
head(OutData)
```

```
## # A tibble: 6 x 5
##   SampleID statistic p.value conf.low conf.high
##      <int>     <dbl>   <dbl>    <dbl>     <dbl>
## 1        1   -1.84    0.0780   -0.860    0.0491
## 2        2   -0.152   0.880    -0.536    0.462
## 3        3   -0.223   0.825    -0.447    0.359
## 4        4   -0.855   0.401    -0.500    0.207
## 5        5   -0.139   0.890    -0.453    0.395
## 6        6   -0.0484  0.962    -0.384    0.366
```

4.4 Summarize Results

We will create flags or indicator variables to indicate if (1) a CI does NOT include 0 (true mean) and (2) p-value < 0.05. These are equivalent criteria corresponding to "false positives". We do this using mutate to create new variables and then summarize to count the number of occurrences.

As expected, we find that about 5% of tests (48/1000) return "false positives", corresponding to alpha = 0.05.

```
OutData <- OutData %>%
           mutate(CIFlag = if_else(conf.low > 0 | conf.high < 0, 1, 0),
                  PvalFlag = if_else(p.value < 0.05, 1, 0))
```

```
OutData %>%
  ungroup() %>%
  summarise(CountCI = sum(CIFlag),
            CountP = sum(PvalFlag))
```

```
## # A tibble: 1 x 2
##   CountCI CountP
##     <dbl>  <dbl>
## 1      48     48
```

```
OutData %>%
  filter(CIFlag == 1) %>%
  head()
```

```
## # A tibble: 6 x 7
##   SampleID statistic p.value conf.low conf.high CIFlag PvalFlag
##      <int>     <dbl>   <dbl>    <dbl>     <dbl>  <dbl>    <dbl>
## 1       17     -2.67  0.0133   -0.826   -0.106       1        1
## 2       30     -2.44  0.0225   -0.842   -0.0700      1        1
## 3       58      2.63  0.0146    0.116    0.953       1        1
## 4       62     -2.21  0.0370   -0.786   -0.0267      1        1
## 5       63     -2.17  0.0398   -0.744   -0.0193      1        1
## 6       64     -2.50  0.0195   -0.668   -0.0644      1        1
```

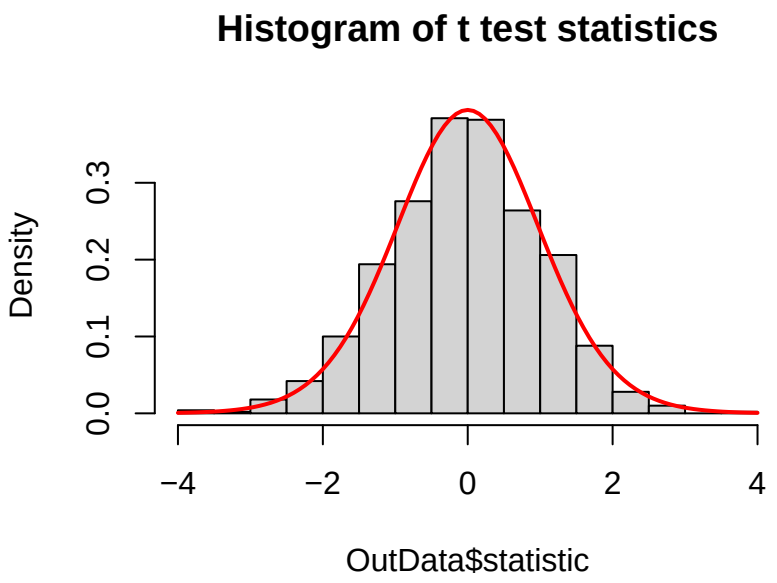
```
table(OutData$CIFlag, OutData$PvalFlag)
```

```
##
##       0   1
##   0 952   0
##   1   0  48
```

4.5 Test statistic and p-value distributions

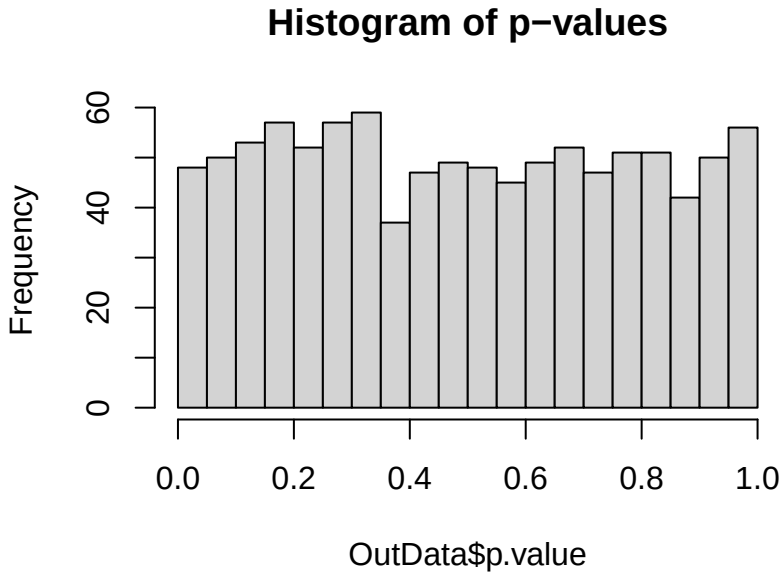
As expected, the t test statistics follow a t distribution.

```
hist(OutData$statistic, freq = FALSE, main = "Histogram of t test statistics")
#Overlay t distribution for comparison
curve(dt(x, df = 24), add = TRUE, col = "red", lwd = 2)
```



P-values follow Uniform distribution.

```
hist(OutData$p.value, breaks = seq(from = 0, to = 1, by = 0.05), main = "Histogram of p-values")
```



CH5.4: sample size and power calculation

Confidence Interval Width

Calculate ME for sample sizes between 5-15

```
n <- seq(from = 5, to = 15, by = 1)
#Equivalent to: seq(5, 15, 1)
sd <- 4
alpha <- 0.05
ME <- qt(1-(alpha/2), df = n-1)*sd/sqrt(n)
out <- data.frame(n, ME)
out
```

```
##     n       ME
## 1   5 4.966656
## 2   6 4.197743
## 3   7 3.699383
## 4   8 3.344084
## 5   9 3.074672
## 6  10 2.861428
## 7  11 2.687237
## 8  12 2.541479
## 9  13 2.417176
## 10 14 2.309531
## 11 15 2.215126
```

```
rm(n, sd, alpha, ME, out)
```

Power for ONE-sided one-sample t-test

```
#Using power.t.test
power.t.test(n = 10, delta = 4, sd = 4,
             sig.level = 0.05, type = "one.sample",
             alternative = "one.sided")
```

```
##
##      One-sample t test power calculation
##
##               n = 10
##           delta = 4
##              sd = 4
##       sig.level = 0.05
##           power = 0.897517
##     alternative = one.sided
```

```
#For illustration: power "by hand" using noncentrality parameter
1 - pt(1.8333, df = 9, ncp = 3.16)
```

```
## [1] 0.8971111
```

Power for TWO-sided one-sample t-test

```
#Using power.t.test
power.t.test(n = 10, delta = 4, sd = 4,
             sig.level = 0.05, type = "one.sample",
             alternative = "two.sided")
```

```
##
##      One-sample t test power calculation
##
##               n = 10
##           delta = 4
##              sd = 4
##       sig.level = 0.05
##           power = 0.8030962
##     alternative = two.sided
```

```
#For illustration: power "by hand" using noncentrality parameter
1 - pt(2.262, df = 9, ncp = 3.16) + pt(-2.262, df = 9, ncp = 3.16)
```

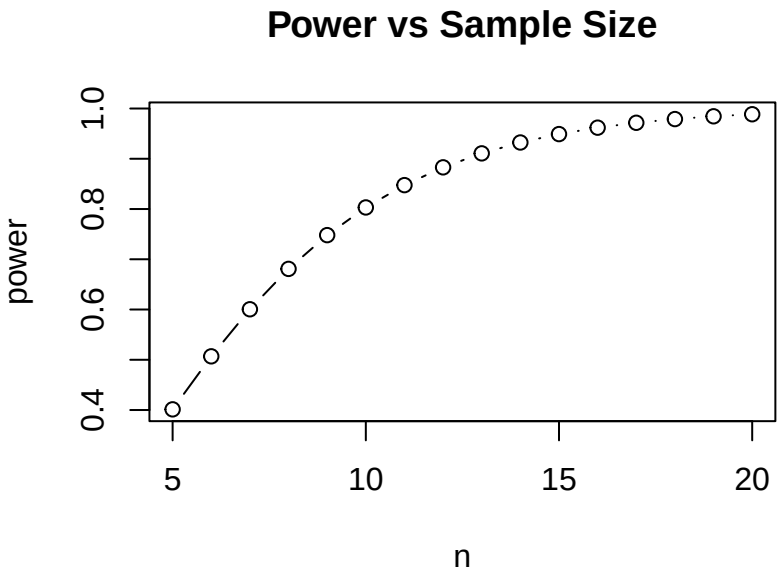
```
## [1] 0.802582
```

Graph of Power vs Sample Size

```
nvec<-seq(5, 20, 1)
powerout1 <- power.t.test(n = nvec, delta = 4, sd = 4,
                          sig.level = 0.05, type = "one.sample",
                          alternative = "two.sided")
powerout1
```

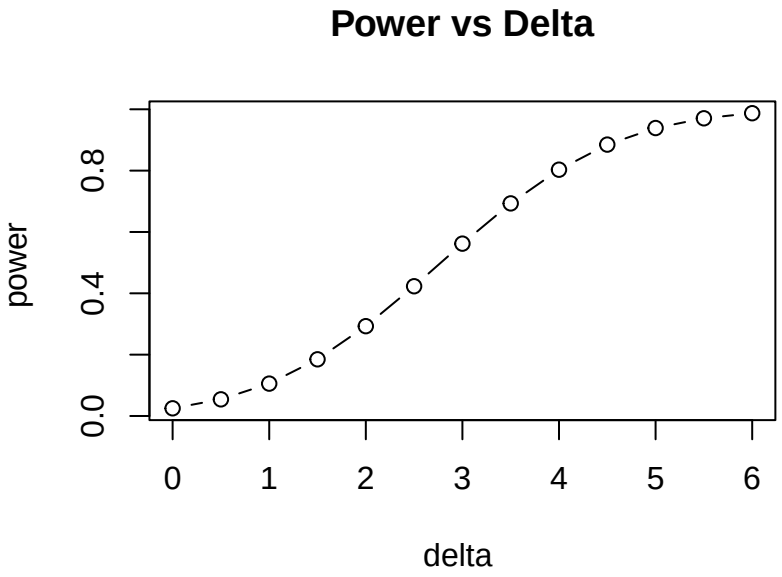
```
##
##      One-sample t test power calculation
##
##               n = 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20
##           delta = 4
##              sd = 4
##       sig.level = 0.05
##           power = 0.4013203, 0.5068167, 0.6004875, 0.6808301, 0.7480155, 0.8030962, 0.8475297, 0.882
##     alternative = two.sided
```

```
plot(powerout1$power ~ powerout1$n,
       type = "b", xlab = "n", ylab = "power",
       main = "Power vs Sample Size")
```



Graph of Power vs Delta (Difference between means)

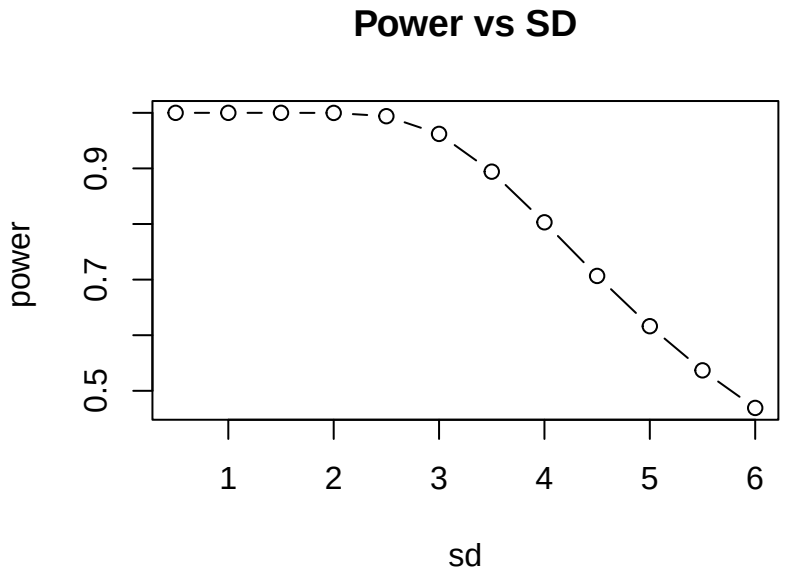
```
deltavec <- seq(0, 6, 0.5)
powerout2 <- power.t.test(n = 10, delta = deltavec, sd = 4,
                          sig.level = 0.05, type = "one.sample",
                          alternative = "two.sided")
plot(powerout2$power ~ powerout2$delta,
     type = "b", xlab = "delta ", ylab =  "power ",
     main = "Power vs Delta")
```



Graph of Power vs SD

```
sdvec <- seq(0.5, 6, 0.5)
powerout3 <- power.t.test(n = 10, delta = 4, sd = sdvec,
                          sig.level = 0.05, type = "one.sample",
                          alternative = "two.sided")
```

```
plot(powerout3$power ~ powerout3$sd,
     type = "b", xlab = "sd", ylab =  "power ",
     main = "Power vs SD")
```




```
rm(powerout1, powerout2, powerout3, nvec, deltavec, sdvec)
```

CH6.1: Two sample t-test and CI

```
library(knitr)
knitr::opts_chunk$set(message = FALSE, comment=NA )
```

We use the two-sample t-test and corresponding confidence interval to compare means using independent samples. Requires the assumption of normality or large sample size.

1 Rat Lead Data

Twenty rats were randomly assigned to two groups.
n1 = 10 rats in the control group received a standard diet.
n2 = 10 rats in the deficient group received a calcium deficient diet.
For both groups, a 0.15% lead-acetate solution was available to drink.
The amount of solution (Y) consumed by each rat was measured.

Notice that this data is in "long" format, with a column indicating the treatment group (trt) and another giving the response (y).

```
library(tidyverse)
RatData <- read.csv("CH6_RatLead.csv")
str(RatData)
```

```
'data.frame':   20 obs. of  2 variables:
 $ trt: Factor w/ 2 levels "control","deficient": 1 1 1 1 1 1 1 1 1 1 ...
 $ y  : num  5.4 6.2 3.1 3.8 6.5 5.8 6.4 4.5 4.9 4 ...
```

```
RatData
```

```
         trt    y
1    control  5.4
2    control  6.2
3    control  3.1
4    control  3.8
5    control  6.5
6    control  5.8
7    control  6.4
8    control  4.5
9    control  4.9
10   control  4.0
11 deficient  8.8
12 deficient  9.5
13 deficient 10.6
14 deficient  9.6
15 deficient  7.5
16 deficient  6.9
17 deficient  7.4
18 deficient  6.5
19 deficient 10.5
20 deficient  8.3
```

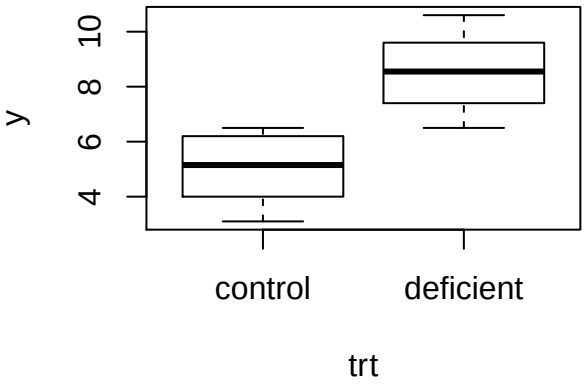
2 Summary Statistics and Graphics

In this example, we use the summarize() and group_by() functions from tidyverse to calculate summary statistics for each group. Another option is to use the aggregate function.

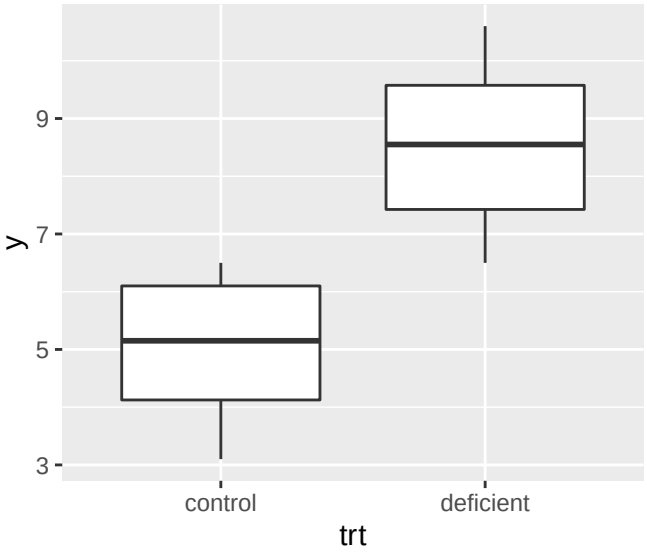
```
SumStats <- RatData %>%
            group_by(trt) %>%
            summarise(n = n(),
                      mean = mean(y),
                      sd = sd(y),
                      se = sd/sqrt(n))
SumStats
```

```
# A tibble: 2 x 5
  trt           n  mean    sd    se
  <fct>     <int> <dbl> <dbl> <dbl>
1 control      10  5.06  1.19 0.376
2 deficient    10  8.56  1.47 0.465
```

```
boxplot(y ~ trt, data = RatData)
```




```
## same plot with ggplot (in tidyverse package suite)
ggplot(RatData, aes(trt,y)) + geom_boxplot()
```



3 Pooled two-sample t-test and CI (Equal variances)

Note that for this example the data is in "long" format.

```
t.test(y ~ trt, var.equal = TRUE, data = RatData)
```

```
    Two Sample t-test

data:  y by trt
t = -5.8507, df = 18, p-value = 1.532e-05
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -4.756813 -2.243187
sample estimates:
  mean in group control mean in group deficient
                   5.06                    8.56
```

4 Welch-Sattertwaite t-test and CI (Unequal variances)

Note that for this example the data is in "long" format.
Note also that the default for t.test is Welch-Satterthwaite t-test (NOT assuming equal variances).

```
t.test(y ~ trt, data = RatData)
```

```
    Welch Two Sample t-test

data:  y by trt
t = -5.8507, df = 17.241, p-value = 1.822e-05
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -4.760793 -2.239207
sample estimates:
  mean in group control mean in group deficient
                   5.06                    8.56
```

5 Reformatting (For Illustration!)

We use the pivot_wider and pivot_longer functions from tidyverse to do some example reformatting.

The function t.test can handle data in either long or wide format. So it is not really necessary to reformat here. We note that long format is usually preferred.

5.1 Reformat Long to Wide

```
WideData <- RatData %>%
            mutate(ID = c(rep(1:10),rep(1:10))) %>%
            pivot_wider(names_from = trt, values_from = y)
str(WideData)
```

```
Classes 'tbl_df', 'tbl' and 'data.frame':   10 obs. of  3 variables:
 $ ID       : int  1 2 3 4 5 6 7 8 9 10
 $ control  : num  5.4 6.2 3.1 3.8 6.5 5.8 6.4 4.5 4.9 4
 $ deficient: num  8.8 9.5 10.6 9.6 7.5 6.9 7.4 6.5 10.5 8.3
```

```
WideData
```

```
# A tibble: 10 x 3
      ID control deficient
   <int>   <dbl>     <dbl>
 1     1     5.4       8.8
 2     2     6.2       9.5
 3     3     3.1      10.6
 4     4     3.8       9.6
 5     5     6.5       7.5
 6     6     5.8       6.9
 7     7     6.4       7.4
 8     8     4.5       6.5
 9     9     4.9      10.5
10    10     4         8.3
```

5.2 Rerun Two-sample t-test using Wide format

Note that for this example the data is in "wide" format.

```
t.test(WideData$control, WideData$deficient)
```

```
    Welch Two Sample t-test

data:  WideData$control and WideData$deficient
t = -5.8507, df = 17.241, p-value = 1.822e-05
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -4.760793 -2.239207
sample estimates:
mean of x mean of y
     5.06      8.56
```

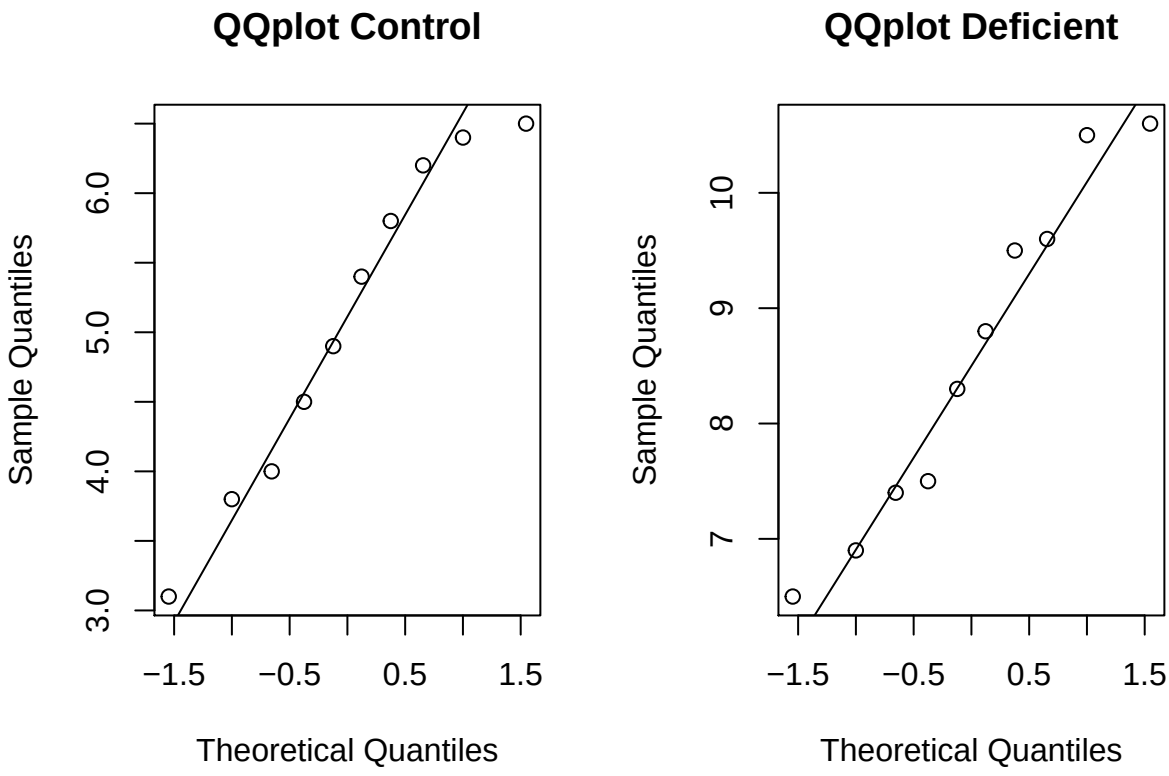
5.3 Reformat Wide to Long

```
LongData <- WideData %>%
            pivot_longer(control:deficient, names_to = "trt", values_to = "y")
str(LongData)
```

```
Classes 'tbl_df', 'tbl' and 'data.frame':   20 obs. of  3 variables:
 $ ID : int  1 1 2 2 3 3 4 4 5 5 ...
 $ trt: chr  "control" "deficient" "control" "deficient" ...
 $ y  : num  5.4 8.8 6.2 9.5 3.1 10.6 3.8 9.6 6.5 7.5 ...
```

6 Evaluating Normality

```
par(mfrow=c(1,2))
qqnorm(WideData$control, main = "QQplot Control")
qqline(WideData$control)
qqnorm(WideData$deficient, main = "QQplot Deficient")
qqline(WideData$deficient)
```



CH6.2: Sample Size and Power for Two Sample t-test

1 Confidence Interval Width

Calculate ME for sample sizes between 5-10.

```
n <- seq(5, 10, 1)
sd <- 4
alpha <- 0.05
ME <- qt(1-alpha/2, 2*n-2)*sd*sqrt(2/n)
out <- data.frame(n1 = n, n2 = n, ME)
out
```

```
##   n1 n2       ME
## 1  5  5 5.833780
## 2  6  6 5.145666
## 3  7  7 4.658498
## 4  8  8 4.289573
## 5  9  9 3.997332
## 6 10 10 3.758244
```

2 Power for One-sided two-sample t-test

```
#Using power.t.test
power.t.test(n = 9, delta = 6, sd = 4, sig.level = 0.05,
             type = "two.sample", alternative = "one.sided")
```

```
##
##      Two-sample t test power calculation
##
##               n = 9
##           delta = 6
##              sd = 4
##       sig.level = 0.05
##           power = 0.9189915
##     alternative = one.sided
##
## NOTE: n is number in *each* group
```

```
#For illustration: power "by hand" using noncentrality parameter
1-pt(1.746, df = 16, ncp = 3.18)
```

```
## [1] 0.9186899
```

3 Power for Two-sided two-sample t-test

```
#Using power.t.test
power.t.test(n = 9, delta = 6, sd = 4, sig.level = 0.05,
```

```
            type = "two.sample", alternative = "two.sided")
```

```
##
##      Two-sample t test power calculation
##
##               n = 9
##           delta = 6
##              sd = 4
##       sig.level = 0.05
##           power = 0.8476098
##     alternative = two.sided
##
## NOTE: n is number in *each* group
```

```
#For illustration: power "by hand" using noncentrality parameter
pt(-2.120, df = 16, ncp = 3.18)+(1-pt(2.120, df = 16, ncp = 3.18))
```

```
## [1] 0.8471516
```

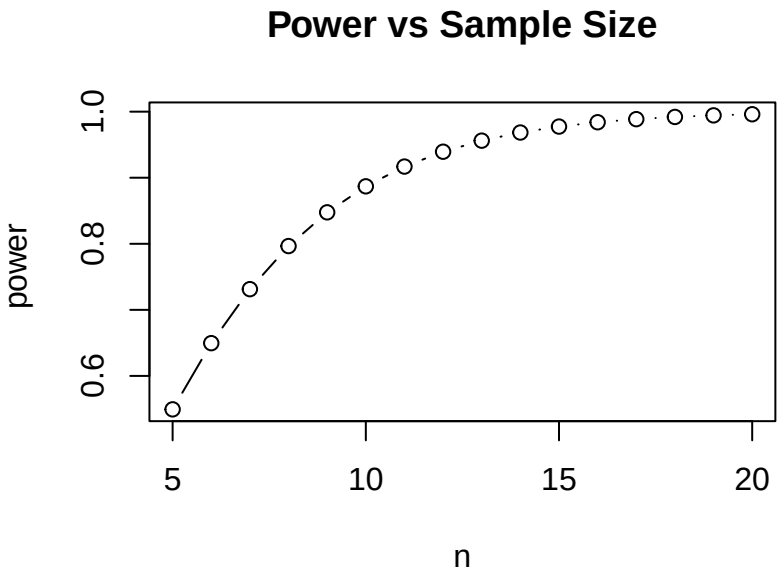
4 Graphs of Power

4.1 Graph of Power vs Sample Size

```
nvec <- seq(5, 20, 1)
powerout1 <-  power.t.test(n = nvec, delta = 6, sd = 4, sig.level = 0.05,
                           type = "two.sample", alternative = "two.sided")
powerout1
```

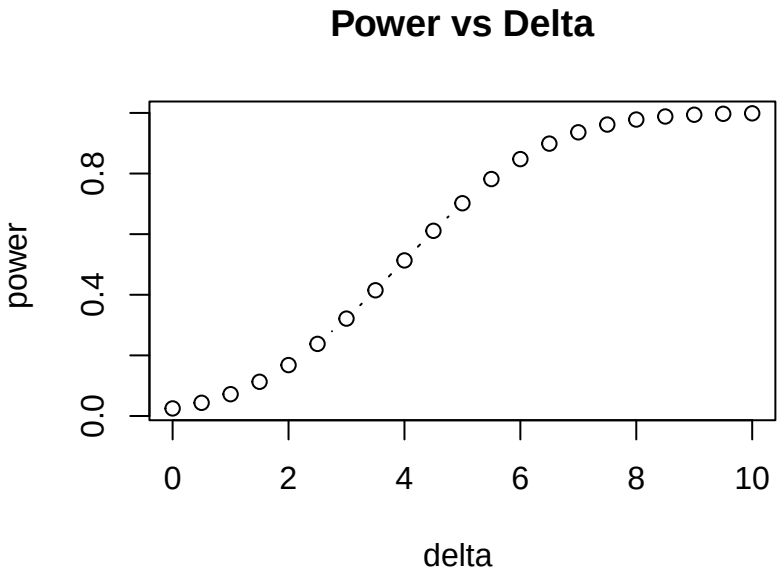
```
##
##      Two-sample t test power calculation
##
##               n = 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20
##           delta = 6
##              sd = 4
##       sig.level = 0.05
##           power = 0.5493642, 0.6495744, 0.7313279, 0.7965441, 0.8476098, 0.8869701, 0.9168991, 0.939
##     alternative = two.sided
##
## NOTE: n is number in *each* group
```

```
plot(powerout1$power ~ powerout1$n,
      type = "b", xlab = "n", ylab = "power ",
      main = "Power vs Sample Size")
```



4.2 Graph of Power vs Delta (Difference between means)

```
deltavec <- seq(0, 10, 0.5)
powerout2 <- power.t.test(n = 9, delta = deltavec, sd = 4, sig.level = 0.05,
                          type = "two.sample", alternative = "two.sided")
plot(powerout2$power ~ powerout2$delta,
     type = "b", xlab = "delta ", ylab = "power ",
     main = "Power vs Delta")
```



CH6.5: Paired t and Wilcoxon Signed Rank tests

1 Dogs Example

This data is from Ott & Longnecker. A study was conducted to investigate the effect of benzedrine on the heart rate of dogs. A total of n = 14 dogs each got both placebo (P) and benzedrine (B) in a randomized order. The response variable is heart rate (BPM). Note that this is **paired** data because each dog experiences both treatments.

While it is not required for analysis, we start by creating a new column/variable giving the difference for each dog. This is helpful for checking normality of the differences.

```
DogData <- read.csv("CH6_Dogs.csv")
str(DogData)
```

```
## 'data.frame':    14 obs. of  3 variables:
##  $ Dog: int  1 2 3 4 5 6 7 8 9 10 ...
##  $ P  : int  250 271 243 252 266 272 293 296 301 298 ...
##  $ B  : int  258 285 245 250 268 278 280 305 319 308 ...
```

```
DogData$Diff <- DogData$P - DogData$B
str(DogData)
```

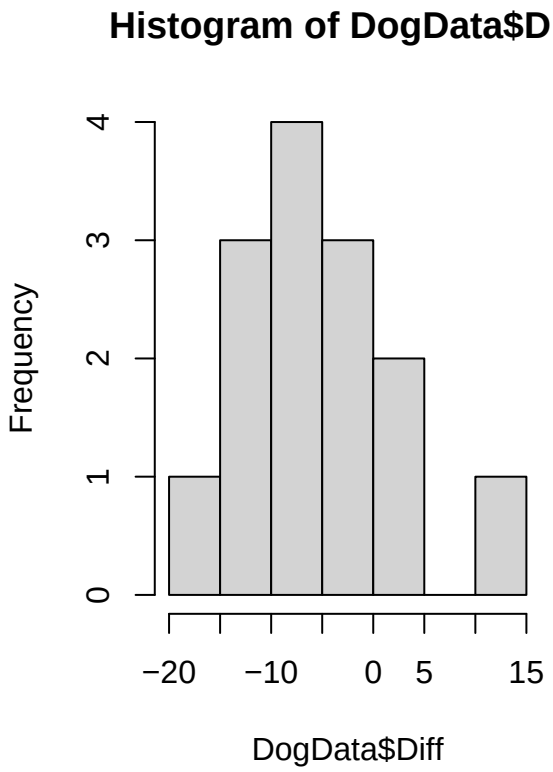
```
## 'data.frame':    14 obs. of  4 variables:
##  $ Dog : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ P   : int  250 271 243 252 266 272 293 296 301 298 ...
##  $ B   : int  258 285 245 250 268 278 280 305 319 308 ...
##  $ Diff: int  -8 -14 -2 2 -2 -6 13 -9 -18 -10 ...
```

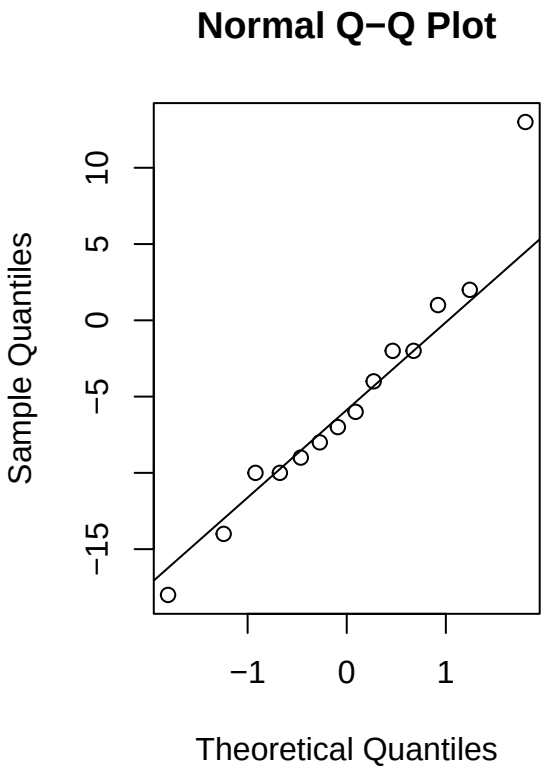
```
mean(DogData$Diff); sd(DogData$Diff)
```

```
## [1] -5.285714
```

```
## [1] 7.630189
```

```
par(mfrow=c(1,2))
hist(DogData$Diff)
qqnorm(DogData$Diff); qqline(DogData$Diff)
```





2 Paired t-test

The paired t-test is used to compare means using paired data. This test requires the assumption of normality of the differences or large sample size.

2.1 Paired t-test (Approach #1)

Primarily for illustration. We run a one-sample t-test using the differences calculated above.

```
t.test(DogData$Diff)
```

```
##
##  One Sample t-test
##
## data:  DogData$Diff
## t = -2.592, df = 13, p-value = 0.02234
## alternative hypothesis: true mean is not equal to 0
## 95 percent confidence interval:
##  -9.6912541 -0.8801745
## sample estimates:
## mean of x
## -5.285714
```

2.2 Paired t-test (Approach #2)

Exactly the same results as above.

```
t.test(DogData$P, DogData$B, paired = TRUE)
```

```
##
```

```
##  Paired t-test
##
## data:  DogData$P and DogData$B
## t = -2.592, df = 13, p-value = 0.02234
## alternative hypothesis: true mean difference is not equal to 0
## 95 percent confidence interval:
##  -9.6912541 -0.8801745
## sample estimates:
## mean difference
##       -5.285714
```

CH7: Inference for comparing Variances

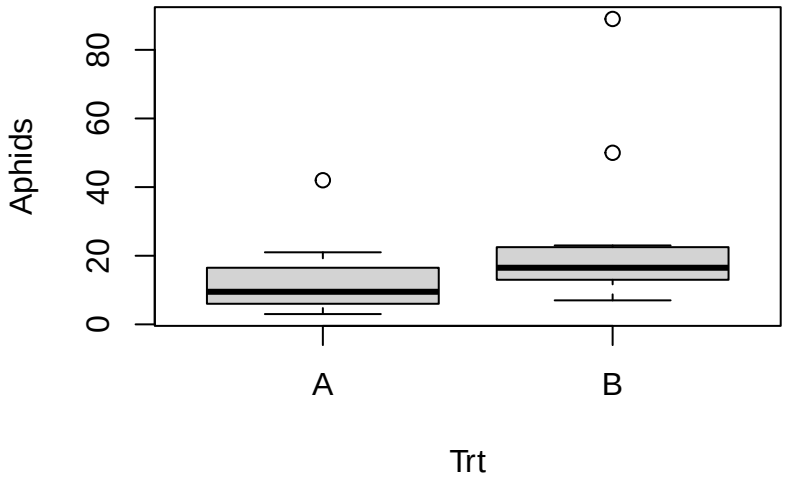
1 Aphid Example

We use this data to illustrate two different tests of variances: 1. The F-test is used to compare two variances. This test requires assumption of normality, so just for illustration here. 2. Levene's test (or Brown-Forsythe test) is more commonly used to compare two (or more) variances, which do NOT require the assumption of normality. Hence, a more reasonable choice for this data.

```
library(car)
library(coin)
Aphids <- read.csv("CH7_Aphids.csv")
str(Aphids)
```

```
## 'data.frame':    24 obs. of  2 variables:
##  $ Trt   : chr  "A" "A" "A" "A" ...
##  $ Aphids: int  21 12 6 9 10 3 6 19 4 7 ...
```

```
boxplot(Aphids ~ Trt, data = Aphids)
```



2 F-test comparing variances

For illustration only. This test requires the assumption of normality, which is not reasonable based on boxplots above (with several outliers).

```
var.test(Aphids ~ Trt, data = Aphids)
```

```
##
##  F test to compare two variances
##
## data:  Aphids by Trt
## F = 0.21818, num df = 11, denom df = 11, p-value = 0.01804
## alternative hypothesis: true ratio of variances is not equal to 1
## 95 percent confidence interval:
```

```
##  0.06281032 0.75790603
## sample estimates:
## ratio of variances
##          0.2181841
```

3 Levene's test comparing variances

This test does NOT require the assumption of normality. It is a commonly used test to compare 2 (or more) variances. The leveneTest() function is from the car package. By default, center = "median" is used and called the Brown-Forsythe test. When we use center = "mean" this corresponds to the "traditional" Levene's test.

```
leveneTest(Aphids ~ Trt, data = Aphids)
```

```
## Warning in leveneTest.default(y = y, group = group, ...): group coerced to
## factor.
```

```
## Levene's Test for Homogeneity of Variance (center = median)
##       Df F value Pr(>F)
## group  1  0.7575 0.3935
##       22
```

```
leveneTest(Aphids ~ Trt, data = Aphids, center = "mean")
```

```
## Warning in leveneTest.default(y = y, group = group, ...): group coerced to
## factor.
```

```
## Levene's Test for Homogeneity of Variance (center = "mean")
##       Df F value Pr(>F)
## group  1  1.9425 0.1773
##       22
```

CH8.1: ANOVA for comparison of several means

1 Rice Example: One-way ANOVA

One-way ANOVA is used to compare means when there are more than two groups.

In this example, the effects of four acids on the growth of rice seedlings are compared in a completely randomized design. Seedling shoot dry weights are compared after 7 days in solution. The goal of the study is to compare mean weight for the four acid treatments.

Pairwise comparisons and diagnostics added to this Rice example with Ch8p2_R

```
library(tidyverse)
rice <- read.csv("CH8_Rice.csv")
str(rice)
```

```
## 'data.frame':    20 obs. of  2 variables:
##  $ trt   : chr  "control" "control" "control" "control" ...
##  $ weight: num  4.23 4.38 4.25 4.3 4.25 3.75 3.68 3.81 3.84 3.76 ...
```

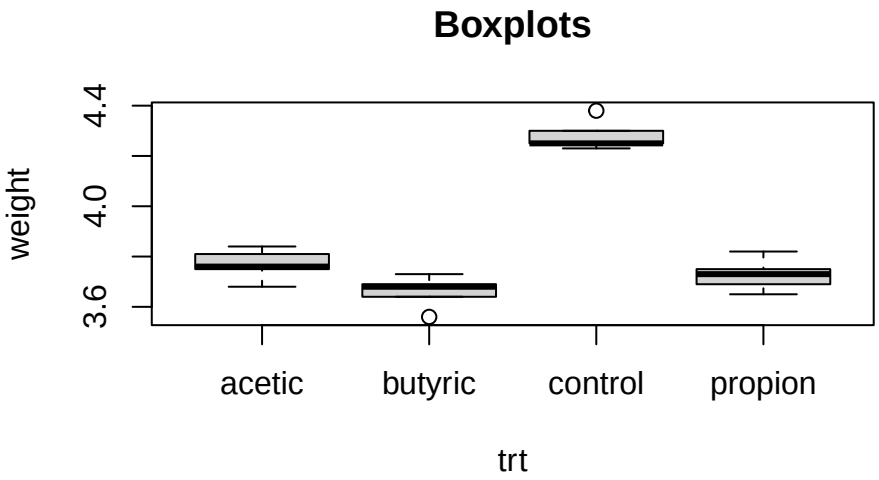
1.1 Summary Statistics and Graphics

Recall that the summarise() and group_by() functions are from the dplyr package.

```
SumStats <- summarise(group_by(rice, trt),
                      n = n(),
                      mean = mean(weight),
                      sd = sd(weight),
                      se = sd/sqrt(n))
SumStats
```

```
## # A tibble: 4 x 5
##   trt         n  mean     sd     se
##   <chr>   <int> <dbl>  <dbl>  <dbl>
## 1 acetic      5  3.77 0.0614 0.0275
## 2 butyric     5  3.66 0.0644 0.0288
## 3 control     5  4.28 0.0606 0.0271
## 4 propion     5  3.73 0.0642 0.0287
```

```
boxplot(weight ~ trt, data = rice, main = "Boxplots")
```



1.2 One-way ANOVA model and table

Note that trt should be defined as.factor. We checked this above using the str() function. We can use either lm() or aov() to fit the one-way ANOVA model. The resulting ANOVA table will be the same. The summary() output for the lm object is not directly needed to address common research questions, but shown here for illustration.

```
#Using lm
LMFit <- lm(weight ~ trt, data = rice)
summary(LMFit)
```

```
##
## Call:
## lm(formula = weight ~ trt, data = rice)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -0.1000 -0.0335 -0.0030  0.0330  0.0980
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  3.76800    0.02803 134.443  < 2e-16 ***
## trtbutyric  -0.10800    0.03964  -2.725    0.015 *
## trtcontrol   0.51400    0.03964  12.968 6.63e-10 ***
## trtpropion  -0.04000    0.03964  -1.009    0.328
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.06267 on 16 degrees of freedom
## Multiple R-squared:  0.951,  Adjusted R-squared:  0.9418
## F-statistic: 103.5 on 3 and 16 DF,  p-value: 1.083e-10
```

```
anova(LMFit)
```

```
## Analysis of Variance Table
##
## Response: weight
##           Df  Sum Sq Mean Sq F value    Pr(>F)
## trt        3 1.21985 0.40662  103.53 1.083e-10 ***
## Residuals 16 0.06284 0.00393
```

```
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
LMFit
```

```
##
## Call:
## lm(formula = weight ~ trt, data = rice)
##
## Coefficients:
## (Intercept)   trtbutyric   trtcontrol   trtpropion
##       3.768       -0.108        0.514       -0.040
```

or

```
#Using aov
AovFit <- aov(weight ~ trt, data = rice)
summary(AovFit)
```

```
##             Df Sum Sq Mean Sq F value   Pr(>F)
## trt          3 1.2199  0.4066   103.5 1.08e-10 ***
## Residuals   16 0.0628  0.0039
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

CH8.2: More with one-way ANOVA and Kruskal-Wallis test

1 Power for One-way ANOVA

power.anova.test() works for balanced scenarios, where the sample size is the same for each group. Notice that the between and within group VARIANCES are needed for this function!

For this example with t = 4 groups, we conjecture that:
1. The true means are 1,2,3,4.
2. The true variance (within group) is 4.

1.1 Calculate the power with n=10 per group

```
var(c(1,2,3,4))
```

```
## [1] 1.666667
```

```
power.anova.test(groups = 4, n = 10,
                 between.var = 1.66667, within.var = 4,
                 sig.level = 0.05)
```

```
##
##      Balanced one-way analysis of variance power calculation
##
##          groups = 4
##               n = 10
##     between.var = 1.66667
##      within.var = 4
##       sig.level = 0.05
##           power = 0.8119587
##
## NOTE: n is number in each group
```

1.2 Calculate the sample size required to achieve 90% power

Notice this calculation returns a non-integer value - round up to an integer!

```
power.anova.test(groups = 4, power=0.90,
                 between.var = 1.66667, within.var = 4,
                 sig.level = 0.05)
```

```
##
##      Balanced one-way analysis of variance power calculation
##
##          groups = 4
##               n = 12.36347
##     between.var = 1.66667
##      within.var = 4
##       sig.level = 0.05
##           power = 0.9
```

```
##
## NOTE: n is number in each group
```

2 Diagnostics (for checking model assumptions)

In our example, five herbicides are compared in their ability to limit the number of poppy plants in oats. The five herbicide treatments are randomly assigned to twenty plots. The results, in number of poppy plants per 3.75 sqft of oats.

```
library(tidyverse)
library(MASS)
library(car)
poppies<-read.csv("CH8_Poppies.csv")
str(poppies)
```

```
## 'data.frame':    20 obs. of  2 variables:
##  $ Trt   : chr  "A" "A" "A" "A" ...
##  $ Plants: int  438 442 319 380 538 422 377 315 77 61 ...
```

```
SumStats <- summarise(group_by(poppies, Trt),
                      n = n(),
                      mean = mean(Plants),
                      sd = sd(Plants),
                      se = sd/sqrt(n))
SumStats
```

```
## # A tibble: 5 x 5
##   Trt       n  mean    sd    se
##   <chr> <int> <dbl> <dbl> <dbl>
## 1 A         4 395.   57.9  29.0
## 2 B         4 413    94.2  47.1
## 3 C         4  86.8  48.0  24.0
## 4 D         4  37.8  33.5  16.8
## 5 E         4  35.2  28.0  14.0
```

2.1 Analysis on the original scale

```
Fit1 <- lm(Plants ~ Trt, data = poppies)
anova(Fit1)
```

```
## Analysis of Variance Table
##
## Response: Plants
##           Df Sum Sq Mean Sq F value    Pr(>F)
## Trt        4 597514  149378  45.451 3.271e-08 ***
## Residuals 15  49299    3287
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

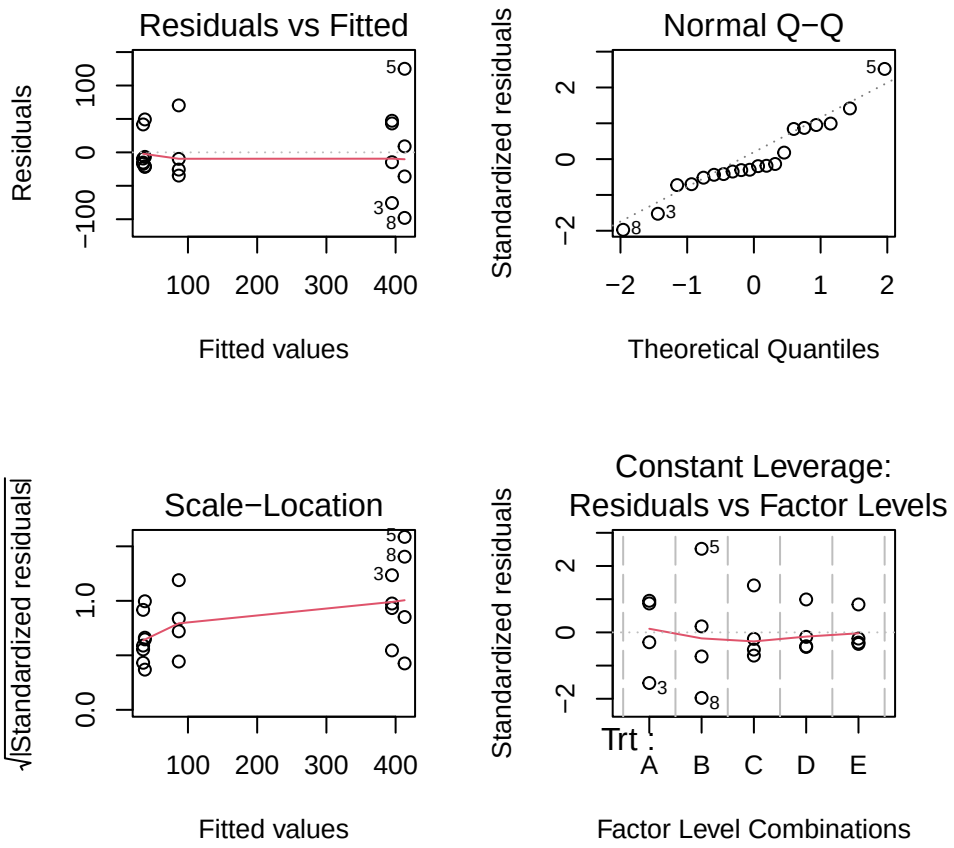
2.2 Checking model assumptions

The assumptions for one-way ANOVA include (1) random samples, independent observations, (2) normally distributed residuals and (3) equality of variances.

When the plot() function is applied to an lm (or aov) object, diagnostic plots are returned. The plots of Resids vs Fitted and QQplot are of primary interest. We will not discuss the other two plots. Recall that

leveneTest() is from the car package.

```
par(mfrow=c(2,2))
plot(Fit1)
```




```
shapiro.test(residuals(Fit1))
```

```
##
##  Shapiro-Wilk normality test
##
## data:  residuals(Fit1)
## W = 0.94308, p-value = 0.2739
```

```
leveneTest(Plants ~ factor(Trt), data = poppies)
```

```
## Levene's Test for Homogeneity of Variance (center = median)
##       Df F value Pr(>F)
## group  4  1.2175 0.3446
##       15
```

3 Transformations

3.1 common transformations: sqrt and log

```
poppies$sqrtPlants <- sqrt(poppies$Plants)
poppies$logPlants  <- log(poppies$Plants)
```

3.1.1 Analysis on the SQRT scale

```
Fit2 <- lm(sqrtPlants ~ Trt, data = poppies)
anova(Fit2)
```

```
## Analysis of Variance Table
##
## Response: sqrtPlants
##           Df Sum Sq Mean Sq F value    Pr(>F)
## Trt        4 866.96 216.739  45.527 3.233e-08 ***
## Residuals 15  71.41   4.761
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

3.1.2 Analysis on the LOG (natural log) scale

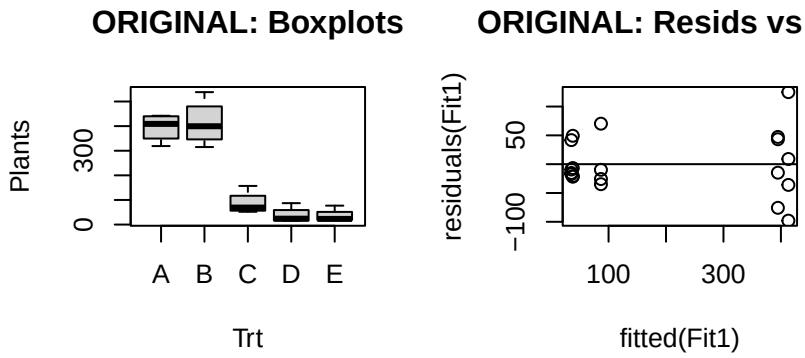
```
Fit3 <- lm(logPlants ~ Trt, data = poppies)
anova(Fit3)
```

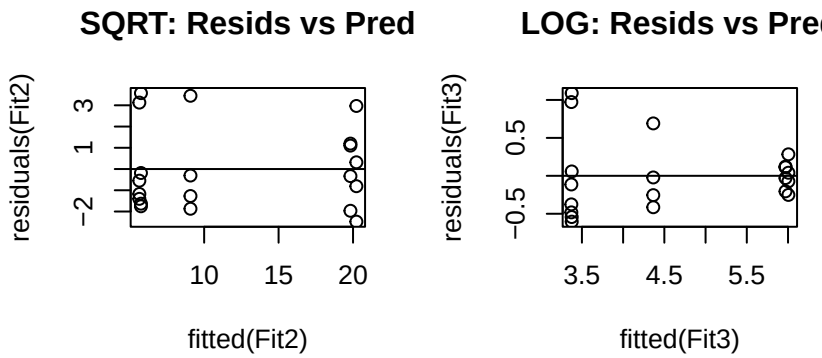
```
## Analysis of Variance Table
##
## Response: logPlants
##           Df  Sum Sq Mean Sq F value    Pr(>F)
## Trt        4 27.6283  6.9071  25.173 1.669e-06 ***
## Residuals 15  4.1157  0.2744
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

3.1.3 Plots

The plot() function could also have been used to generate diagnostic plots for each analysis. Based on the diagnostic plots, the square root transform looks reasonable.

```
par(mfrow=c(2,2))
boxplot(Plants ~ Trt, data = poppies, main = "ORIGINAL: Boxplots")
plot(residuals(Fit1) ~ fitted(Fit1), main = "ORIGINAL: Resids vs Pred"); abline(h = 0)
plot(residuals(Fit2) ~ fitted(Fit2), main = "SQRT: Resids vs Pred"); abline(h = 0)
plot(residuals(Fit3) ~ fitted(Fit3),main="LOG: Resids vs Pred"); abline(h = 0)
```

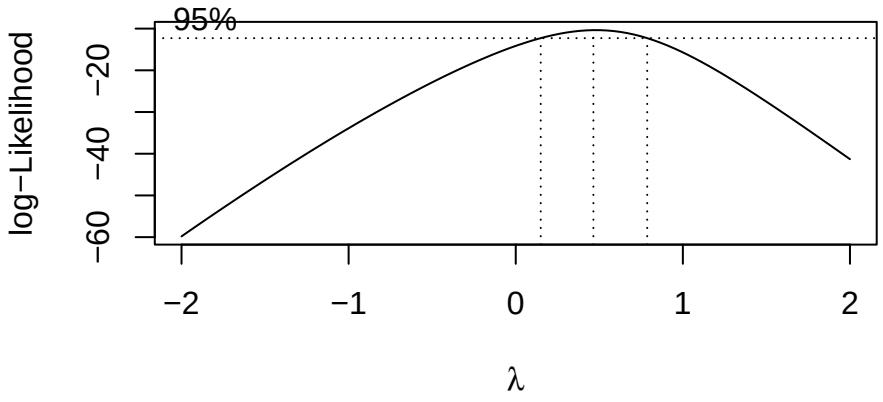




3.2 Box-Cox

The boxcox() function is from the MASS package. For this example, $\lambda = 0.5$ is suggested by the Box-Cox summary graph. This supports the use of the square root transformation.

```
boxcox(Plants ~ Trt, data = poppies)
```



4 Kruskal-Wallis

Kruskal-Wallis is a non-parametric alternative to the one-way ANOVA that does not require the assumption of normality.

```
kruskal.test(Plants ~ Trt, data = poppies)
```

```
##
##  Kruskal-Wallis rank sum test
##
## data:  Plants by Trt
## Kruskal-Wallis chi-squared = 14.929, df = 4, p-value = 0.004851
```

CH9.1: Multiple Comparisons

1 Clover Example

Red Clover was inoculated with t = 6 (bacteria) strains. The response variable is nitrogen (N) content. The goal of the study is to compare means for the six treatments. Five pots of each of six treatments completely randomized in a greenhouse experiment (t = 6 trts, n = 5 pots/trt, nT=30).

```
library(tidyverse)
library(emmeans)
library(multcomp)
Clover <- read.csv("CH9_clover.csv")
str(Clover)
```

```
## 'data.frame':    30 obs. of  2 variables:
##  $ Strain: chr  "3DOk1" "3DOk1" "3DOk1" "3DOk1" ...
##  $ N     : num  19.4 32.1 32.6 33 27 14.4 14.2 11.8 14.3 11.6 ...
```

```
table(Clover$Strain)
```

```
##
##  3DOk1 3DOk13  3DOk4  3DOk5  3DOk7 compos
##      5      5      5      5      5      5
```

Using tidverse:
1. We define Strain as.factor.
2. We reorder factor levels so that the control (compos) group is first. This is helpful for dunnett's method, but there are other reasons you may want to reorder the factor levels.

```
Clover <- Clover %>%
          mutate(Strain = as_factor(Strain)) %>%
          mutate(Strain = fct_relevel(Strain, "compos"))
str(Clover)
```

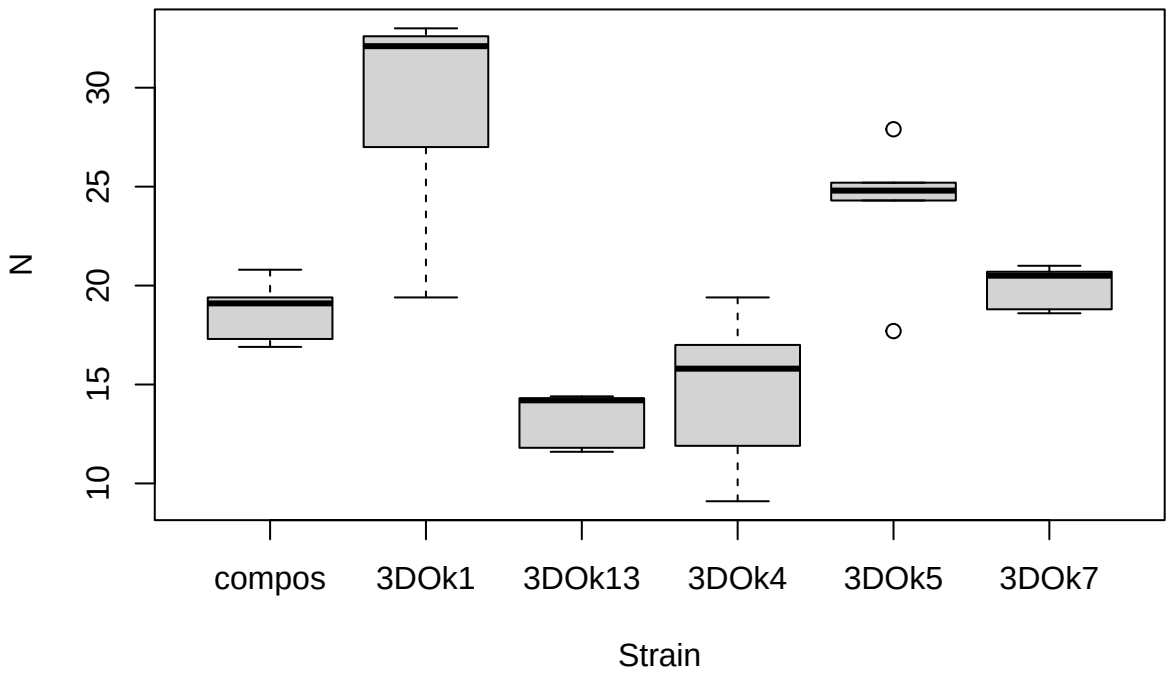
```
## 'data.frame':    30 obs. of  2 variables:
##  $ Strain: Factor w/ 6 levels "compos","3DOk1",..: 2 2 2 2 2 3 3 3 3 3 ...
##  $ N     : num  19.4 32.1 32.6 33 27 14.4 14.2 11.8 14.3 11.6 ...
```

```
table(Clover$Strain)
```

```
##
## compos  3DOk1 3DOk13  3DOk4  3DOk5  3DOk7
##      5      5      5      5      5      5
```

2 Basic One-way ANOVA analysis

```
boxplot(N ~ Strain, data = Clover, main = "Boxplots")
```




```
SumStats <- Clover %>%
            group_by(Strain) %>%
            summarise(
               n    = n(),
               mean = mean(N),
               sd   = sd(N),
               se   = sd / sqrt(n) )
SumStats
```

```
## # A tibble: 6 x 5
##   Strain     n  mean    sd    se
##   <fct>  <int> <dbl> <dbl> <dbl>
## 1 compos     5  18.7  1.60 0.716
## 2 3DOk1      5  28.8  5.80 2.59
## 3 3DOk13     5  13.3  1.43 0.638
## 4 3DOk4      5  14.6  4.12 1.84
## 5 3DOk5      5  24.0  3.78 1.69
## 6 3DOk7      5  19.9  1.13 0.505
```

```
OneWayFit <- lm(N ~ Strain, data = Clover)
anova(OneWayFit)
```

```
## Analysis of Variance Table
##
## Response: N
##           Df Sum Sq Mean Sq F value    Pr(>F)
## Strain     5 847.05 169.409   14.37 1.485e-06 ***
## Residuals 24 282.93  11.789
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

3 Summary statistics using emmeans()

```
emout <- emmeans(OneWayFit, ~ Strain)
emout
```

```
##  Strain emmean   SE df lower.CL upper.CL
##  compos   18.7 1.54 24     15.5     21.9
##  3DOk1    28.8 1.54 24     25.7     32.0
##  3DOk13   13.3 1.54 24     10.1     16.4
##  3DOk4    14.6 1.54 24     11.5     17.8
##  3DOk5    24.0 1.54 24     20.8     27.1
##  3DOk7    19.9 1.54 24     16.8     23.1
##
## Confidence level used: 0.95
```

4 Pairwise comparisons using emmeans

4.1 Fisher's/UNadjusted Pairwise Comparisons

4.1.1 Simultaneous tests

```
pairs(emout, adjust = "none")
```

```
##  contrast        estimate   SE df t.ratio p.value
##  compos - 3DOk1    -10.12 2.17 24  -4.660  0.0001
##  compos - 3DOk13     5.44 2.17 24   2.505  0.0194
##  compos - 3DOk4      4.06 2.17 24   1.870  0.0738
##  compos - 3DOk5     -5.28 2.17 24  -2.431  0.0229
##  compos - 3DOk7     -1.22 2.17 24  -0.562  0.5794
##  3DOk1 - 3DOk13     15.56 2.17 24   7.166  <.0001
##  3DOk1 - 3DOk4      14.18 2.17 24   6.530  <.0001
##  3DOk1 - 3DOk5       4.84 2.17 24   2.229  0.0354
##  3DOk1 - 3DOk7       8.90 2.17 24   4.099  0.0004
##  3DOk13 - 3DOk4     -1.38 2.17 24  -0.636  0.5311
##  3DOk13 - 3DOk5    -10.72 2.17 24  -4.937  <.0001
##  3DOk13 - 3DOk7     -6.66 2.17 24  -3.067  0.0053
##  3DOk4 - 3DOk5      -9.34 2.17 24  -4.301  0.0002
##  3DOk4 - 3DOk7      -5.28 2.17 24  -2.431  0.0229
##  3DOk5 - 3DOk7       4.06 2.17 24   1.870  0.0738
```

4.1.2 Simultaneous Confidence Intervals

```
confint(pairs(emout, adjust = "none"))
```

```
##  contrast        estimate   SE df lower.CL upper.CL
##  compos - 3DOk1    -10.12 2.17 24  -14.602   -5.638
##  compos - 3DOk13     5.44 2.17 24    0.958    9.922
##  compos - 3DOk4      4.06 2.17 24   -0.422    8.542
##  compos - 3DOk5     -5.28 2.17 24   -9.762   -0.798
##  compos - 3DOk7     -1.22 2.17 24   -5.702    3.262
##  3DOk1 - 3DOk13     15.56 2.17 24   11.078   20.042
##  3DOk1 - 3DOk4      14.18 2.17 24    9.698   18.662
##  3DOk1 - 3DOk5       4.84 2.17 24    0.358    9.322
##  3DOk1 - 3DOk7       8.90 2.17 24    4.418   13.382
```

```
##  3DOk13 - 3DOk4     -1.38 2.17 24   -5.862    3.102
##  3DOk13 - 3DOk5    -10.72 2.17 24  -15.202   -6.238
##  3DOk13 - 3DOk7     -6.66 2.17 24  -11.142   -2.178
##  3DOk4 - 3DOk5      -9.34 2.17 24  -13.822   -4.858
##  3DOk4 - 3DOk7      -5.28 2.17 24   -9.762   -0.798
##  3DOk5 - 3DOk7       4.06 2.17 24   -0.422    8.542
##
## Confidence level used: 0.95
```

4.2 Bonferonni Adjusted Pairwise Comparisons

4.2.1 Simultaneous tests

```
pairs(emout, adjust = "bonferroni")
```

```
##  contrast        estimate   SE df t.ratio p.value
##  compos - 3DOk1    -10.12 2.17 24  -4.660  0.0015
##  compos - 3DOk13     5.44 2.17 24   2.505  0.2914
##  compos - 3DOk4      4.06 2.17 24   1.870  1.0000
##  compos - 3DOk5     -5.28 2.17 24  -2.431  0.3431
##  compos - 3DOk7     -1.22 2.17 24  -0.562  1.0000
##  3DOk1 - 3DOk13     15.56 2.17 24   7.166  <.0001
##  3DOk1 - 3DOk4      14.18 2.17 24   6.530  <.0001
##  3DOk1 - 3DOk5       4.84 2.17 24   2.229  0.5317
##  3DOk1 - 3DOk7       8.90 2.17 24   4.099  0.0062
##  3DOk13 - 3DOk4     -1.38 2.17 24  -0.636  1.0000
##  3DOk13 - 3DOk5    -10.72 2.17 24  -4.937  0.0007
##  3DOk13 - 3DOk7     -6.66 2.17 24  -3.067  0.0793
##  3DOk4 - 3DOk5      -9.34 2.17 24  -4.301  0.0037
##  3DOk4 - 3DOk7      -5.28 2.17 24  -2.431  0.3431
##  3DOk5 - 3DOk7       4.06 2.17 24   1.870  1.0000
##
## P value adjustment: bonferroni method for 15 tests
```

4.2.2 Simultaneous Confidence Intervals

```
confint(pairs(emout, adjust = "bonferroni"))
```

```
##  contrast        estimate   SE df lower.CL upper.CL
##  compos - 3DOk1    -10.12 2.17 24   -17.20   -3.044
##  compos - 3DOk13     5.44 2.17 24    -1.64   12.516
##  compos - 3DOk4      4.06 2.17 24    -3.02   11.136
##  compos - 3DOk5     -5.28 2.17 24   -12.36    1.796
##  compos - 3DOk7     -1.22 2.17 24    -8.30    5.856
##  3DOk1 - 3DOk13     15.56 2.17 24     8.48   22.636
##  3DOk1 - 3DOk4      14.18 2.17 24     7.10   21.256
##  3DOk1 - 3DOk5       4.84 2.17 24    -2.24   11.916
##  3DOk1 - 3DOk7       8.90 2.17 24     1.82   15.976
##  3DOk13 - 3DOk4     -1.38 2.17 24    -8.46    5.696
##  3DOk13 - 3DOk5    -10.72 2.17 24   -17.80   -3.644
##  3DOk13 - 3DOk7     -6.66 2.17 24   -13.74    0.416
##  3DOk4 - 3DOk5      -9.34 2.17 24   -16.42   -2.264
##  3DOk4 - 3DOk7      -5.28 2.17 24   -12.36    1.796
##  3DOk5 - 3DOk7       4.06 2.17 24    -3.02   11.136
```

```
##
## Confidence level used: 0.95
## Conf-level adjustment: bonferroni method for 15 estimates
```

4.3 Tukey Adjusted Pairwise Comparisons

4.3.1 Simultaneous tests

Note that Tukey adjustment is done by default!

```
pairs(emout)
```

```
##  contrast        estimate   SE df t.ratio p.value
##  compos - 3DOk1    -10.12 2.17 24  -4.660  0.0012
##  compos - 3DOk13     5.44 2.17 24   2.505  0.1622
##  compos - 3DOk4      4.06 2.17 24   1.870  0.4435
##  compos - 3DOk5     -5.28 2.17 24  -2.431  0.1852
##  compos - 3DOk7     -1.22 2.17 24  -0.562  0.9926
##  3DOk1 - 3DOk13     15.56 2.17 24   7.166  <.0001
##  3DOk1 - 3DOk4      14.18 2.17 24   6.530  <.0001
##  3DOk1 - 3DOk5       4.84 2.17 24   2.229  0.2617
##  3DOk1 - 3DOk7       8.90 2.17 24   4.099  0.0049
##  3DOk13 - 3DOk4     -1.38 2.17 24  -0.636  0.9871
##  3DOk13 - 3DOk5    -10.72 2.17 24  -4.937  0.0006
##  3DOk13 - 3DOk7     -6.66 2.17 24  -3.067  0.0528
##  3DOk4 - 3DOk5      -9.34 2.17 24  -4.301  0.0030
##  3DOk4 - 3DOk7      -5.28 2.17 24  -2.431  0.1852
##  3DOk5 - 3DOk7       4.06 2.17 24   1.870  0.4435
##
## P value adjustment: tukey method for comparing a family of 6 estimates
```

4.3.2 Simultaneous Confidence Intervals

```
confint(pairs(emout))
```

```
##  contrast        estimate   SE df lower.CL upper.CL
##  compos - 3DOk1    -10.12 2.17 24   -16.83  -3.4058
##  compos - 3DOk13     5.44 2.17 24    -1.27  12.1542
##  compos - 3DOk4      4.06 2.17 24    -2.65  10.7742
##  compos - 3DOk5     -5.28 2.17 24   -11.99   1.4342
##  compos - 3DOk7     -1.22 2.17 24    -7.93   5.4942
##  3DOk1 - 3DOk13     15.56 2.17 24     8.85  22.2742
##  3DOk1 - 3DOk4      14.18 2.17 24     7.47  20.8942
##  3DOk1 - 3DOk5       4.84 2.17 24    -1.87  11.5542
##  3DOk1 - 3DOk7       8.90 2.17 24     2.19  15.6142
##  3DOk13 - 3DOk4     -1.38 2.17 24    -8.09   5.3342
##  3DOk13 - 3DOk5    -10.72 2.17 24   -17.43  -4.0058
##  3DOk13 - 3DOk7     -6.66 2.17 24   -13.37   0.0542
##  3DOk4 - 3DOk5      -9.34 2.17 24   -16.05  -2.6258
##  3DOk4 - 3DOk7      -5.28 2.17 24   -11.99   1.4342
##  3DOk5 - 3DOk7       4.06 2.17 24    -2.65  10.7742
##
## Confidence level used: 0.95
## Conf-level adjustment: tukey method for comparing a family of 6 estimates
```

5 CLD

The CLD display provides a succinct summary of the results of the pairwise comparisons. But it relies entirely on a "bright line" p-value cutoff (0.05 by default).

The cld() function is from the multcomp package. Note that the multcompView package may also be needed.

5.1 Fisher's/UNadjusted CLD

```
cld(emout,adjust = "none")
```

```
##  Strain emmean   SE df lower.CL upper.CL .group
##  3DOk13   13.3 1.54 24     10.1     16.4  1
##  3DOk4    14.6 1.54 24     11.5     17.8  12
##  compos   18.7 1.54 24     15.5     21.9   23
##  3DOk7    19.9 1.54 24     16.8     23.1    34
##  3DOk5    24.0 1.54 24     20.8     27.1     4
##  3DOk1    28.8 1.54 24     25.7     32.0      5
##
## Confidence level used: 0.95
## significance level used: alpha = 0.05
## NOTE: If two or more means share the same grouping letter,
##       then we cannot show them to be different.
##       But we also did not show them to be the same.
```

5.2 Bonferonni Adjusted CLD

```
cld(emout,adjust = "bonferroni")
```

```
##  Strain emmean   SE df lower.CL upper.CL .group
##  3DOk13   13.3 1.54 24     8.85     17.7  1
##  3DOk4    14.6 1.54 24    10.23     19.1  1
##  compos   18.7 1.54 24    14.29     23.1  12
##  3DOk7    19.9 1.54 24    15.51     24.3  12
##  3DOk5    24.0 1.54 24    19.57     28.4   23
##  3DOk1    28.8 1.54 24    24.41     33.2    3
##
## Confidence level used: 0.95
## Conf-level adjustment: bonferroni method for 6 estimates
## P value adjustment: bonferroni method for 15 tests
## significance level used: alpha = 0.05
## NOTE: If two or more means share the same grouping letter,
##       then we cannot show them to be different.
##       But we also did not show them to be the same.
```

5.3 Tukey Adjusted CLD

```
cld(emout)
```

```
##  Strain emmean   SE df lower.CL upper.CL .group
##  3DOk13   13.3 1.54 24     10.1     16.4  1
##  3DOk4    14.6 1.54 24     11.5     17.8  1
##  compos   18.7 1.54 24     15.5     21.9  12
##  3DOk7    19.9 1.54 24     16.8     23.1  12
```

```
##  3DOk5    24.0 1.54 24     20.8     27.1   23
##  3DOk1    28.8 1.54 24     25.7     32.0    3
##
## Confidence level used: 0.95
## P value adjustment: tukey method for comparing a family of 6 estimates
## significance level used: alpha = 0.05
## NOTE: If two or more means share the same grouping letter,
##       then we cannot show them to be different.
##       But we also did not show them to be the same.
```

CH9: Dunnett's Method and Contrast Comparisons

1 Clover Example: Dunnett's Method

Red Clover was inoculated with t = 6 (bacteria) strains. The response variable is nitrogen (N) content. The goal of the study is to compare means for the six treatments. Five pots of each of six treatments completely randomized in a greenhouse experiment (t = 6 trts, n = 5 pots/trt, nT=30).

```
library(tidyverse)
library(emmeans)
library(multcomp)
Clover <- read.csv("CH9_clover.csv")
str(Clover)
```

```
## 'data.frame':    30 obs. of  2 variables:
##  $ Strain: chr  "3DOk1" "3DOk1" "3DOk1" "3DOk1" ...
##  $ N     : num  19.4 32.1 32.6 33 27 14.4 14.2 11.8 14.3 11.6 ...
```

```
table(Clover$Strain)
```

```
##
##  3DOk1 3DOk13  3DOk4  3DOk5  3DOk7 compos
##      5      5      5      5      5      5
```

Using tidverse:
1. We define Strain as.factor.
2. We reorder factor levels so that the control (compos) group is first. This is helpful for dunnett's method, but there are other reasons you may want to reorder the factor levels.

```
Clover <- Clover %>%
          mutate(Strain = as_factor(Strain)) %>%
          mutate(Strain = fct_relevel(Strain, "compos"))
str(Clover)
```

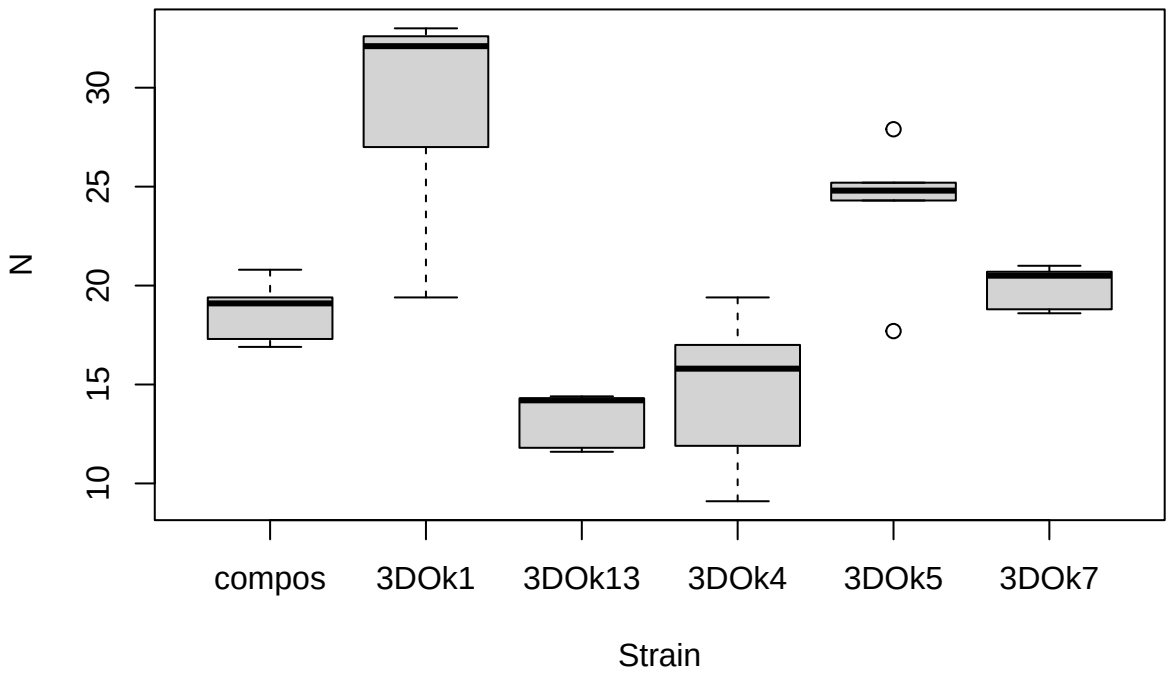
```
## 'data.frame':    30 obs. of  2 variables:
##  $ Strain: Factor w/ 6 levels "compos","3DOk1",..: 2 2 2 2 2 3 3 3 3 3 ...
##  $ N     : num  19.4 32.1 32.6 33 27 14.4 14.2 11.8 14.3 11.6 ...
```

```
table(Clover$Strain)
```

```
##
## compos  3DOk1 3DOk13  3DOk4  3DOk5  3DOk7
##      5      5      5      5      5      5
```

1.1 Basic One-way ANOVA analysis

```
boxplot(N ~ Strain, data = Clover, main = "Boxplots")
```




```
SumStats <- Clover %>%
            group_by(Strain) %>%
            summarise(
               n    = n(),
               mean = mean(N),
               sd   = sd(N),
               se   = sd / sqrt(n) )
SumStats
```

```
## # A tibble: 6 x 5
##   Strain     n  mean    sd    se
##   <fct>  <int> <dbl> <dbl> <dbl>
## 1 compos     5  18.7  1.60 0.716
## 2 3DOk1      5  28.8  5.80 2.59
## 3 3DOk13     5  13.3  1.43 0.638
## 4 3DOk4      5  14.6  4.12 1.84
## 5 3DOk5      5  24.0  3.78 1.69
## 6 3DOk7      5  19.9  1.13 0.505
```

```
OneWayFit <- lm(N ~ Strain, data = Clover)
anova(OneWayFit)
```

```
## Analysis of Variance Table
##
## Response: N
##           Df Sum Sq Mean Sq F value    Pr(>F)
## Strain     5 847.05 169.409   14.37 1.485e-06 ***
## Residuals 24 282.93  11.789
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

1.2 Dunnett Adjusted Pairwise Comparisons

Note that the factor levels were reordered above, so that control is listed first!

```
emmeans(OneWayFit, dunnett ~ Strain)
```

```
## $emmeans
##  Strain emmean   SE df lower.CL upper.CL
##  compos   18.7 1.54 24     15.5     21.9
##  3DOk1    28.8 1.54 24     25.7     32.0
##  3DOk13   13.3 1.54 24     10.1     16.4
##  3DOk4    14.6 1.54 24     11.5     17.8
##  3DOk5    24.0 1.54 24     20.8     27.1
##  3DOk7    19.9 1.54 24     16.8     23.1
##
## Confidence level used: 0.95
##
## $contrasts
##  contrast        estimate   SE df t.ratio p.value
##  3DOk1 - compos     10.12 2.17 24   4.660  0.0005
##  3DOk13 - compos    -5.44 2.17 24  -2.505  0.0776
##  3DOk4 - compos     -4.06 2.17 24  -1.870  0.2543
##  3DOk5 - compos      5.28 2.17 24   2.431  0.0902
##  3DOk7 - compos      1.22 2.17 24   0.562  0.9349
##
## P value adjustment: dunnettx method for 5 tests
```

2 Wheat Example: Contrasts for One-Way ANOVA

For many analyses, the ANOVA table and pairwise comparisons of means address all research questions. However, in some cases contrasts are required to address additional comparisons of interest.

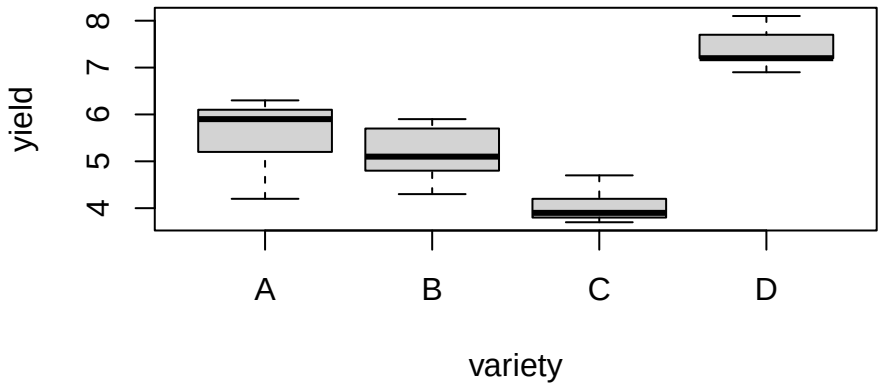
In this example, we analyze a one-way completely randomized design with 4 treatments (A, B, C, D) and 5 observations per treatment.

```
library(tidyverse)
library(emmeans)
wheat <- read.csv("CH8_Wheat.csv")
str(wheat)
```

```
## 'data.frame':    20 obs. of  2 variables:
##  $ variety: chr  "A" "A" "A" "A" ...
##  $ yield  : num  6.3 5.9 4.2 5.2 6.1 4.3 4.8 5.7 5.9 5.1 ...
```

2.1 Basic One-way ANOVA analysis

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




```
SumStats <- summarise(group_by(wheat, variety),
                n    = n(),
                mean = mean(yield),
                sd   = sd(yield),
                se   = sd / sqrt(n) )
SumStats
```

```
## # A tibble: 4 x 5
##   variety     n  mean    sd    se
##   <chr>   <int> <dbl> <dbl> <dbl>
## 1 A           5  5.54 0.856 0.383
## 2 B           5  5.16 0.654 0.293
## 3 C           5  4.06 0.404 0.181
## 4 D           5  7.42 0.476 0.213
```

```
OneWayFit <- lm(yield ~ variety, data = wheat)
anova(OneWayFit)
```

```
## Analysis of Variance Table
##
## Response: yield
##           Df Sum Sq Mean Sq F value   Pr(>F)
## variety    3 29.346  9.7818  25.227 2.64e-06 ***
## Residuals 16  6.204  0.3878
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
emmout <- emmeans(OneWayFit, "variety")
emmout
```

```
##  variety emmean    SE df lower.CL upper.CL
##  A         5.54 0.278 16     4.95     6.13
##  B         5.16 0.278 16     4.57     5.75
##  C         4.06 0.278 16     3.47     4.65
##  D         7.42 0.278 16     6.83     8.01
##
## Confidence level used: 0.95
```

```
pairs(emmout, adjust = "none")
```

```
##  contrast estimate    SE df t.ratio p.value
##  A - B        0.38 0.394 16   0.965  0.3490
##  A - C        1.48 0.394 16   3.758  0.0017
##  A - D       -1.88 0.394 16  -4.774  0.0002
```

```
##  B - C        1.10 0.394 16   2.793  0.0130
##  B - D       -2.26 0.394 16  -5.739  <.0001
##  C - D       -3.36 0.394 16  -8.532  <.0001
```

2.2 Contrasts

The order of coefficients is the order of sample means. Can check the contrast estimates using the sample means or emmeans.

```
contrast(emmout,list(
          #Contrast1:  A vs B
          AvB = c(1, -1, 0, 0),
          #Contrast2: A+B vs C+D
          ABvCD = c(0.5, 0.5, -0.5, -0.5),
          #Contrast3: D vs others
          DvOthers = c(-1/3, -1/3, -1/3, 1),
          #Contrast4: C vs (A+B)
          CvAB = c(-0.5, -0.5, 1, 0)
          ))
```

```
##  contrast estimate    SE df t.ratio p.value
##  AvB          0.38 0.394 16   0.965  0.3490
##  ABvCD       -0.39 0.278 16  -1.400  0.1805
##  DvOthers     2.50 0.322 16   7.775  <.0001
##  CvAB        -1.29 0.341 16  -3.782  0.0016
```

CH10.3: McNemar's Test and Chi-square GOF Test

1 McNemar's Test

McNemar's test is used to compare paired proportions.

1.1 Opinions Example

This data is from the General Social Survey (GSS) but taken from Agresti.
Subjects (n = 1144) were asked whether, to help the environment, they would be willing to (1) pay higher taxes (TaxInc) and (2) accept a cut in living standards (LSDec). Because each subject was asked both questions, we have paired data for which the responses not independent.

```
OpinionsData <- read.csv("CH10_Opinions.csv")
str(OpinionsData)
```

```
## 'data.frame':    1144 obs. of  3 variables:
##  $ Person: int  1 2 3 4 5 6 7 8 9 10 ...
##  $ LSDec : chr  "yes" "yes" "yes" "yes" ...
##  $ TaxInc: chr  "yes" "yes" "yes" "yes" ...
```

```
head(OpinionsData)
```

```
##   Person LSDec TaxInc
## 1      1   yes    yes
## 2      2   yes    yes
## 3      3   yes    yes
## 4      4   yes    yes
## 5      5   yes    yes
## 6      6   yes    yes
```

1.2 Summary Tables

The table() function from base R can be handy for summarizing categorical variables.

```
LSDecTable <- table(OpinionsData$LSDec)
LSDecTable
```

```
##
##  no yes
## 810 334
```

```
prop.table(LSDecTable)
```

```
##
##       no      yes
## 0.708042 0.291958
```

```
TaxIncTable <- table(OpinionsData$TaxInc)
TaxIncTable
```

```
##
```

```
##  no yes
## 785 359
```

```
prop.table(TaxIncTable)
```

```
##
##        no       yes
## 0.6861888 0.3138112
```

29.2% are willing to accept a living standard decrease.
31.4% are willing to accept a tax increase.

```
OpinionsTable <- table(OpinionsData$TaxInc, OpinionsData$LSDec)
OpinionsTable
```

```
##
##        no yes
##   no  678 107
##   yes 132 227
```

1.3 McNemar's Test (from summarized data)

We can run McNemar's test using the table constructed above.

```
mcnemar.test(OpinionsTable)
```

```
##
##  McNemar's Chi-squared test with continuity correction
##
## data:  OpinionsTable
## McNemar's chi-squared = 2.41, df = 1, p-value = 0.1206
```

Or we can enter the summary data "by hand" by creating a matrix.

```
OpinionsMatrix <- matrix(c(678, 107,
                           132, 227), byrow = TRUE, nrow = 2)
OpinionsMatrix
```

```
##      [,1] [,2]
## [1,]  678  107
## [2,]  132  227
```

```
mcnemar.test(OpinionsMatrix)
```

```
##
##  McNemar's Chi-squared test with continuity correction
##
## data:  OpinionsMatrix
## McNemar's chi-squared = 2.41, df = 1, p-value = 0.1206
```

1.4 McNemar's Test (from raw data)

```
mcnemar.test(x = OpinionsData$TaxInc, y = OpinionsData$LSDec)
```

```
##
##  McNemar's Chi-squared test with continuity correction
##
## data:  OpinionsData$TaxInc and OpinionsData$LSDec
## McNemar's chi-squared = 2.41, df = 1, p-value = 0.1206
```

2 Chisquare GOF Test

The chi-square goodness of fit (GOF) test is used to compare observed proportions (or counts) for a single categorical variable to some expected probabilities under H0. The more common test is the chi-square test for contingency tables (see the next section).

In this example from Ott&Longnecker, we test H0: π_1=9/16, π_2=3/16, π_3=3/16, π_4=1/16. These null hypothesized probabilities are motivated by Mendel's laws.

Note: In practice, the hypothesized probabilities (or proportions) would be motivated by the research question.

```
chisq.test(c(773, 231, 238, 59), p = c(9/16, 3/16, 3/16, 1/16), correct = FALSE)
```

```
##
##  Chi-squared test for given probabilities
##
## data:  c(773, 231, 238, 59)
## X-squared = 9.2714, df = 3, p-value = 0.02589
```

2.1 Expected Counts

```
out <- chisq.test(c(773, 231, 238, 59), p = c(9/16, 3/16, 3/16, 1/16), correct = FALSE)
out$expected
```

```
## [1] 731.8125 243.9375 243.9375  81.3125
```

CH10.4 Chi-Square test & Fishers Exact for Contingency Tables

1 Chi-Square test & Fishers Exact for Contingency Tables

The χ^2-square test for contingency tables is used to test for an association between row and column variables and for the equality of proportions, which can also be used to check the equality of proportions. If sample size is small (see warning generated for the Birds Example), then Fisher's Exact test (FET) is preferred for the equality of proportions.

Notes:

1. For most of these examples, we start from a summarized table of counts (constructed using the matrix() function). But in practice, it is much more common for data to start in a data.frame. However, it is relatively easy to summarize into counts or tables using table() in R. See the Birds data for an example.
2. In these examples, I use correct = FALSE with the chisq.test function to match hand calculations from the notes. But in practice, I am fine with the default continuity correction!

1.1 Skiers Example

```
Skiers <- matrix(c(109, 31, 122, 17), nrow = 2, byrow = TRUE)
colnames(Skiers) <- c("NoCold", "YesCold")
rownames(Skiers) <- c("Placebo", "VitC")
Skiers
```

```
##         NoCold YesCold
## Placebo    109      31
## VitC       122      17
```

```
prop.table(Skiers, 1)
```

```
##            NoCold   YesCold
## Placebo 0.7785714 0.2214286
## VitC    0.8776978 0.1223022
```

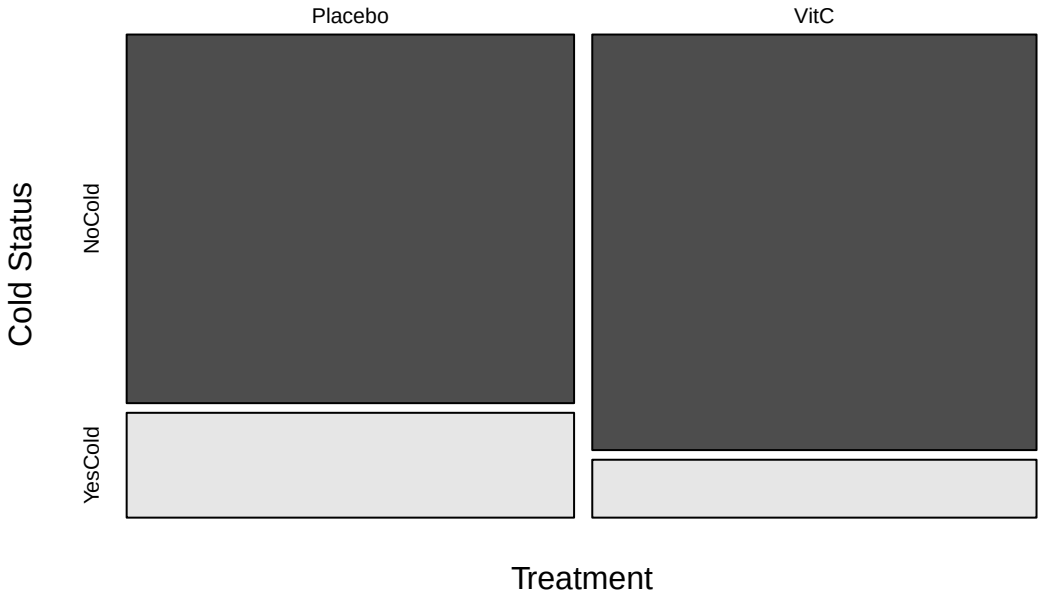
```
chisq.test(Skiers, correct = FALSE)
```

```
##
##  Pearson's Chi-squared test
##
## data:  Skiers
## X-squared = 4.8114, df = 1, p-value = 0.02827
```

```
#Look at Expected Values
SkierTest<-chisq.test(Skiers, correct = FALSE)
SkierTest$expected
```

```
##          NoCold  YesCold
## Placebo 115.914 24.08602
## VitC    115.086 23.91398
```

```r
mosaicplot(Skiers, color = TRUE, xlab = "Treatment", ylab = "Cold Status")
```



1.2 Rat Tumor Example

```r
Tumors <- matrix(c(90, 10, 81, 19, 86, 14), nrow = 3, byrow = TRUE)
colnames(Tumors) <- c("NoTumor", "SomeTumors")
rownames(Tumors) <- c("Ctrl", "High", "Low")
Tumors
```

```
##      NoTumor SomeTumors
## Ctrl      90         10
## High      81         19
## Low       86         14
```

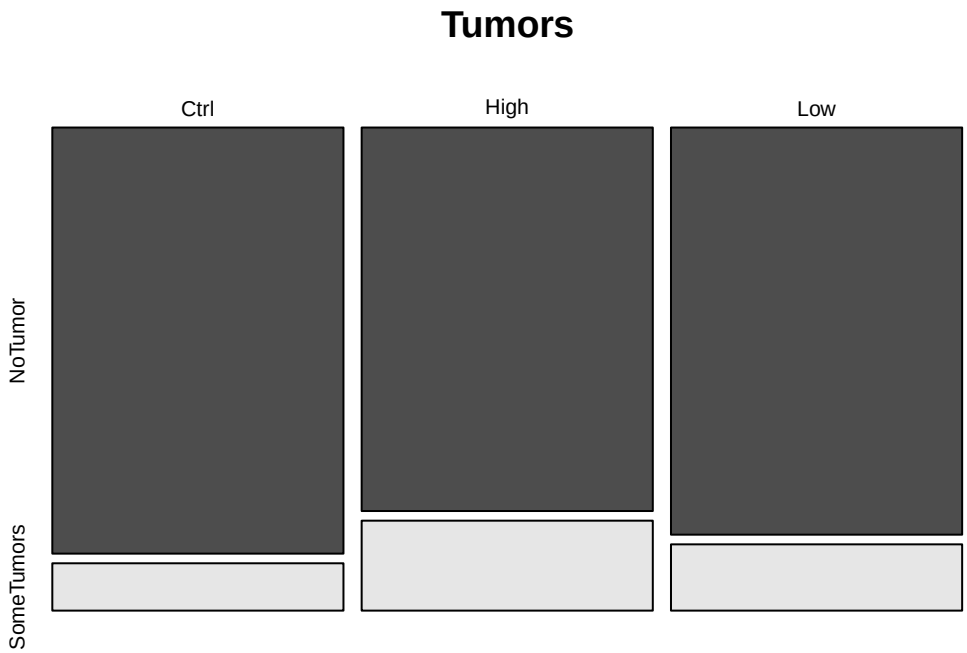
```r
prop.table(Tumors, 1)
```

```
##      NoTumor SomeTumors
## Ctrl    0.90       0.10
## High    0.81       0.19
## Low     0.86       0.14
```

```r
chisq.test(Tumors, correct = FALSE)
```

```
##
##  Pearson's Chi-squared test
##
## data:  Tumors
## X-squared = 3.3119, df = 2, p-value = 0.1909
```

```
mosaicplot(Tumors, color = TRUE)
```



1.3 Opinions Example

```
Guns <- matrix(c(66, 311, 236, 784), nrow = 2, byrow = TRUE)
colnames(Guns)<-c("NoDP", "YesDP")
rownames(Guns)<-c("NoGR", "YesGR")
Guns
```

```
##       NoDP YesDP
## NoGR    66   311
## YesGR  236   784
```

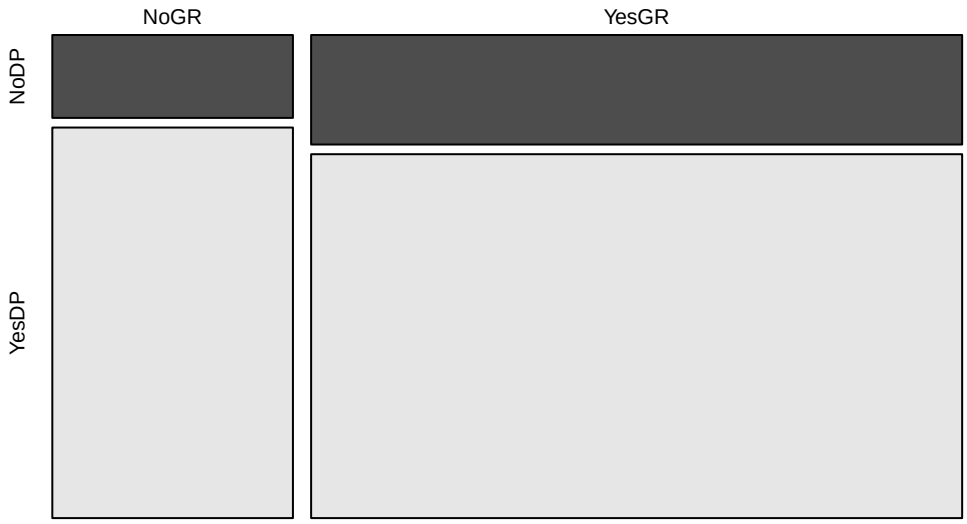
```
prop.table(Guns, 1)
```

```
##            NoDP     YesDP
## NoGR  0.1750663 0.8249337
## YesGR 0.2313725 0.7686275
```

```
chisq.test(Guns, correct = FALSE)
```

```
##
##  Pearson's Chi-squared test
##
## data:  Guns
## X-squared = 5.1503, df = 1, p-value = 0.02324
```

```
mosaicplot(Guns, color = TRUE)
```




```
rm(Guns)
```

1.4 Birds Example

Approach 1: Since the data is in a data.frame, we can construct a summary table and then pass the table to the fisher.test() function. This is the approach we used in previous examples.

```
Birds <- read.csv("CH10_Birds.csv")
str(Birds)
```

```
## 'data.frame':    22 obs. of  3 variables:
##  $ ID  : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ Type: chr  "Blue" "Blue" "Blue" "Blue" ...
##  $ Disc: chr  "Yes" "Yes" "Yes" "Yes" ...
```

```
BirdTable <- with(table(Type, Disc), data = Birds)
BirdTable
```

```
##       Disc
## Type   No Yes
##   Blue  6   4
##   Gold  9   3
```

```
fisher.test(BirdTable)
```

```
##
##  Fisher's Exact Test for Count Data
##
## data:  BirdTable
## p-value = 0.6517
## alternative hypothesis: true odds ratio is not equal to 1
## 95 percent confidence interval:
##  0.05404558 4.33256378
## sample estimates:
## odds ratio
##  0.5163825
```

Approach 2: We can run FET and chi-square test without first creating the summary table. Note however that the with() function is handy here. Also since the sample sizes are small, the chi-square test is just for illustration here!

```
with(fisher.test(x = Type, y = Disc), data = Birds)
```

```
##
##  Fisher's Exact Test for Count Data
##
## data:  Type and Disc
## p-value = 0.6517
## alternative hypothesis: true odds ratio is not equal to 1
## 95 percent confidence interval:
##  0.05404558 4.33256378
## sample estimates:
## odds ratio
##  0.5163825
```

```
with(chisq.test(x = Type, y = Disc, correct = FALSE), data = Birds)
```

```
## Warning in chisq.test(x = Type, y = Disc, correct = FALSE): Chi-squared
## approximation may be incorrect
```

```
##
##  Pearson's Chi-squared test
##
## data:  Type and Disc
## X-squared = 0.56571, df = 1, p-value = 0.452
```

```
rm(Birds, BirdTable)
```

CH10.5: Odds Ratios

1 Odds Ratios

Odds ratio and its corresponding CI are an alternative to presenting and testing proportions. This is especially useful for case-control studies. We return to the some of the same examples that we used to illustrate the chi-square test for contingency tables. In addition, we look at the Birth control data representing a case-control study.

Notes:

1. When using the oddsratio() function in R, it helps to have (1) reference/control group in first row and (2) "event" in last column. For more details see the function help.
2. Notice that the oddsratio() function provides the results of the chi-square test and Fisher's Exact Test.

1.1 Skiers Example

```
library(epitools)
Skiers <- matrix(c(109, 31, 122, 17), nrow = 2, byrow = TRUE)
colnames(Skiers) <- c("NoCold", "YesCold")
rownames(Skiers) <- c("Placebo", "VitC")
oddsratio(Skiers, method = "wald")
```

```
## $data
##         NoCold YesCold Total
## Placebo    109      31   140
## VitC       122      17   139
## Total      231      48   279
##
## $measure
##                           NA
## odds ratio with 95% C.I.  estimate     lower     upper
##                   Placebo 1.0000000        NA        NA
##                   VitC    0.4899524 0.2569419 0.9342709
##
## $p.value
##          NA
## two-sided midp.exact fisher.exact chi.square
##   Placebo         NA           NA         NA
##   VitC    0.02951602   0.03849249 0.02827186
##
## $correction
## [1] FALSE
##
## attr(,"method")
## [1] "Unconditional MLE & normal approximation (Wald) CI"
```

1.2 Rat Tumors Example

```
Tumors <- matrix(c(90, 10, 81, 19, 86, 14), nrow = 3, byrow = TRUE)
colnames(Tumors) <- c("NoTumor", "SomeTumors")
rownames(Tumors) <- c("Ctrl", "High", "Low")
oddsratio(Tumors, method = "wald")
```

```
## $data
##       NoTumor SomeTumors Total
## Ctrl       90         10   100
## High       81         19   100
## Low        86         14   100
## Total     257         43   300
##
## $measure
##                          NA
## odds ratio with 95% C.I. estimate     lower    upper
##                     Ctrl 1.000000        NA       NA
##                     High 2.111111 0.9275180 4.805071
##                     Low  1.465116 0.6177245 3.474957
##
## $p.value
##          NA
## two-sided midp.exact fisher.exact chi.square
##      Ctrl         NA           NA         NA
##      High 0.07510514    0.1069786 0.07069593
##      Low  0.39553616    0.5146243 0.38408825
##
## $correction
## [1] FALSE
##
## attr(,"method")
## [1] "Unconditional MLE & normal approximation (Wald) CI"
```

1.3 Birth Control Example

```
BC <- matrix(c(132,35,34,23), nrow = 2, byrow = TRUE)
colnames(BC) <- c("NoMI", "YesMI")
rownames(BC) <- c("NoBC", "YesBC")
oddsratio(BC, method = "wald")
```

```
## $data
##       NoMI YesMI Total
## NoBC   132    35   167
## YesBC   34    23    57
## Total  166    58   224
##
## $measure
##                          NA
## odds ratio with 95% C.I. estimate    lower    upper
##                     NoBC  1.000000       NA       NA
##                     YesBC 2.551261 1.335615 4.873357
##
## $p.value
```

```
##          NA
## two-sided  midp.exact fisher.exact  chi.square
##     NoBC           NA           NA          NA
##     YesBC 0.005478672  0.005190049 0.003902078
##
## $correction
## [1] FALSE
##
## attr(,"method")
## [1] "Unconditional MLE & normal approximation (Wald) CI"
```

2 Berkeley Example

We use a classic data set where we consider combining 2x2 contingency tables (gender by admission for several departments). The Berkeley admissions study is an example of Simpson's Paradox. We use Breslow-Day test to test for equality of odds ratios comparing across departments.

Notes:

1. We make use of the available R data set. Hence there is no need to import or create data, simply load using data()! But note that the layout of the data is not what we might have preferred. However, the calculated odds ratios still correspond to the odds of Admission for Males vs Females.
2. This example uses several functions that we will NOT see again after this section. For example margin.table.

```
data(UCBAdmissions)
UCBAdmissions
```

```
## , , Dept = A
##
##           Gender
## Admit      Male Female
##   Admitted  512     89
##   Rejected  313     19
##
## , , Dept = B
##
##           Gender
## Admit      Male Female
##   Admitted  353     17
##   Rejected  207      8
##
## , , Dept = C
##
##           Gender
## Admit      Male Female
##   Admitted  120    202
##   Rejected  205    391
##
## , , Dept = D
##
##           Gender
## Admit      Male Female
##   Admitted  138    131
##   Rejected  279    244
```

```
##
## , , Dept = E
##
##           Gender
## Admit      Male Female
##   Admitted   53     94
##   Rejected  138    299
##
## , , Dept = F
##
##           Gender
## Admit      Male Female
##   Admitted   22     24
##   Rejected  351    317
```

```
class(UCBAdmissions)
```

```
## [1] "table"
```

2.1 Combined Analysis (Ignoring Department)

```
library(epitools)
CombineDepts <- margin.table(UCBAdmissions, c(1, 2))
CombineDepts
```

```
##           Gender
## Admit      Male Female
##   Admitted 1198    557
##   Rejected 1493   1278
```

```
prop.table(CombineDepts, 2)
```

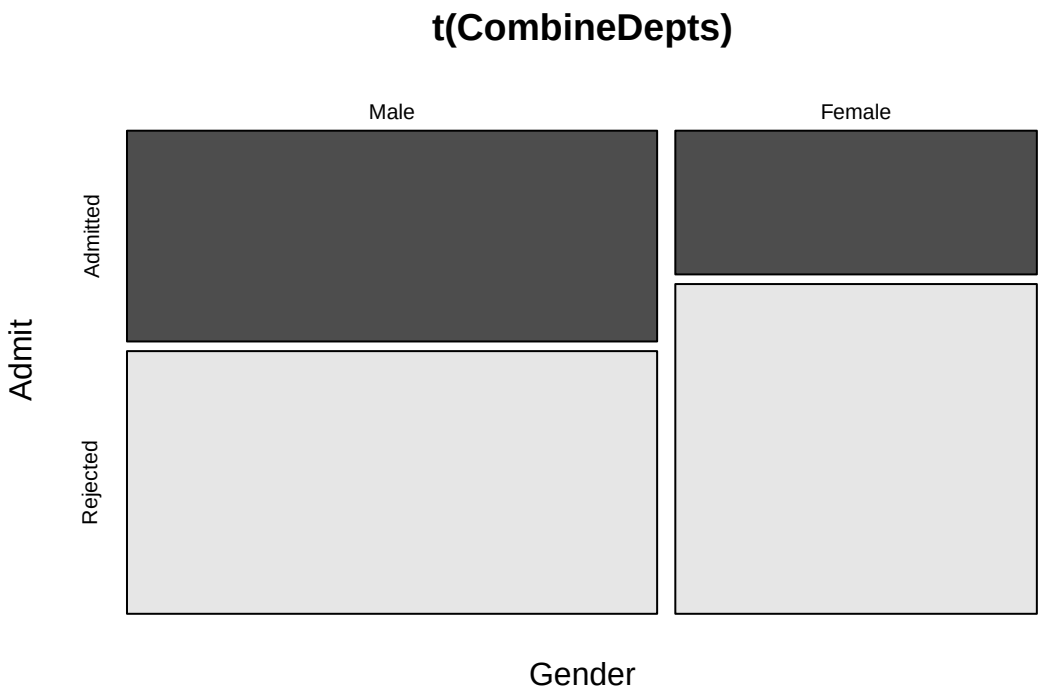
```
##           Gender
## Admit           Male    Female
##   Admitted 0.4451877 0.3035422
##   Rejected 0.5548123 0.6964578
```

```
oddsratio(CombineDepts, method = "wald")
```

```
## $data
##           Gender
## Admit      Male Female Total
##   Admitted 1198    557  1755
##   Rejected 1493   1278  2771
##   Total    2691   1835  4526
##
## $measure
##           odds ratio with 95% C.I.
## Admit      estimate    lower    upper
##   Admitted  1.00000       NA       NA
##   Rejected  1.84108 1.624377 2.086693
##
## $p.value
##           two-sided
## Admit      midp.exact fisher.exact chi.square
##   Admitted         NA           NA         NA
```

```
##   Rejected          0 4.835903e-22 7.8136e-22
##
## $correction
## [1] FALSE
##
## attr(,"method")
## [1] "Unconditional MLE & normal approximation (Wald) CI"
```

```
mosaicplot(t(CombineDepts), color = TRUE)
```



2.2 Analysis BY Department

Here we use the cmh.test() function from the lawstat package to calculate odds ratios by department. But notice we can also use the oddsratio() function to calculate the odds ratio for a single department.

```
library(lawstat)
cmh.test(UCBAdmissions)
```

```
##
##  Cochran-Mantel-Haenszel Chi-square Test
##
## data:  UCBAdmissions
## CMH statistic = 1.52461, df = 1.00000, p-value = 0.21692, MH Estimate =
## 0.90470, Pooled Odd Ratio = 1.84108, Odd Ratio of level 1 = 0.34921,
## Odd Ratio of level 2 = 0.80250, Odd Ratio of level 3 = 1.13306, Odd
## Ratio of level 4 = 0.92128, Odd Ratio of level 5 = 1.22163, Odd Ratio
## of level 6 = 0.82787
```

```
oddsratio(UCBAdmissions[,,1], method = "wald")
```

```
## $data
##           Gender
## Admit      Male Female Total
##   Admitted  512     89   601
```

```
##   Rejected  313     19   332
##   Total     825    108   933
##
## $measure
##           odds ratio with 95% C.I.
## Admit      estimate     lower     upper
##   Admitted 1.000000        NA        NA
##   Rejected 0.349212 0.2086756 0.5843954
##
## $p.value
##           two-sided
## Admit        midp.exact fisher.exact   chi.square
##   Admitted           NA           NA           NA
##   Rejected 1.534042e-05 1.669189e-05 3.280404e-05
##
## $correction
## [1] FALSE
##
## attr(,"method")
## [1] "Unconditional MLE & normal approximation (Wald) CI"
```

But it is also important to note that other things vary by department including admission rate and proportion of male applicants.
Note that the 3 departments with highest proportion female applicants also have the lowest admission rates!

```
Admit.by.Dept <- margin.table(UCBAdmissions, c(1, 3))
prop.table(Admit.by.Dept, 2)
```

```
##           Dept
## Admit                A          B          C          D          E          F
##   Admitted 0.64415863 0.63247863 0.35076253 0.33964646 0.25171233 0.06442577
##   Rejected 0.35584137 0.36752137 0.64923747 0.66035354 0.74828767 0.93557423
```

```
Gender.by.Dept <- margin.table(UCBAdmissions, c(2,3))
prop.table(Gender.by.Dept, 2)
```

```
##         Dept
## Gender             A          B          C          D          E          F
##   Male   0.88424437 0.95726496 0.35403050 0.52651515 0.32705479 0.52240896
##   Female 0.11575563 0.04273504 0.64596950 0.47348485 0.67294521 0.47759104
```

2.3 Breslow-Day Test for Equality of Odds Ratios

```
library(DescTools)
BreslowDayTest(UCBAdmissions)
```

```
##
##  Breslow-Day test on Homogeneity of Odds Ratios
##
## data:  UCBAdmissions
## X-squared = 18.826, df = 5, p-value = 0.002071
```

We Reject H0 and conclude that the odds ratios are not the same for all departments. This is not surprising given the noticeable differences in the odds ratios shown above.

3 Drug Clinic Example

Another example of combining 2x2 tables. This time using data from a designed experiment looking at response (improvement or no improvement) versus drug (active or placebo) at 3 study locations. We use Breslow-Day test to test for equality of odds ratios comparing across study locations. We use the Cochran-Mantel-Haenszel test to combine information across locations.

3.1 Create the Data Array

```
Drugs <- array(c(40,15,10,35,
                    35,20,15,30,
                 43,31,7,19),
      dim = c(2, 2, 3),
      dimnames = list(  Trt = c("Drug", "Plac"),
                Response = c("Imp", "NoImp"),
                Clinic = c("1", "2", "3")))
Drugs
```

```
## , , Clinic = 1
##
##       Response
## Trt    Imp NoImp
##   Drug  40    10
##   Plac  15    35
##
## , , Clinic = 2
##
##       Response
## Trt    Imp NoImp
##   Drug  35    15
##   Plac  20    30
##
## , , Clinic = 3
##
##       Response
## Trt    Imp NoImp
##   Drug  43     7
##   Plac  31    19
```

3.2 Breslow-Day Test for Equality of Odds Ratios

```
library(DescTools)
BreslowDayTest(Drugs)
```

```
##
##  Breslow-Day test on Homogeneity of Odds Ratios
##
## data:  Drugs
## X-squared = 2.8164, df = 2, p-value = 0.2446
```

3.3 CMH Test

```r
library(lawstat)
cmh.test(Drugs)
```

```
##
##  Cochran-Mantel-Haenszel Chi-square Test
##
## data:  Drugs
## CMH statistic = 3.8943e+01, df = 1.0000e+00, p-value = 4.3630e-10, MH
## Estimate = 4.8981e+00, Pooled Odd Ratio = 4.6932e+00, Odd Ratio of
## level 1 = 9.3333e+00, Odd Ratio of level 2 = 3.5000e+00, Odd Ratio of
## level 3 = 3.7650e+00
```

CH11.2: SLR, Transformation, Lack of Fit

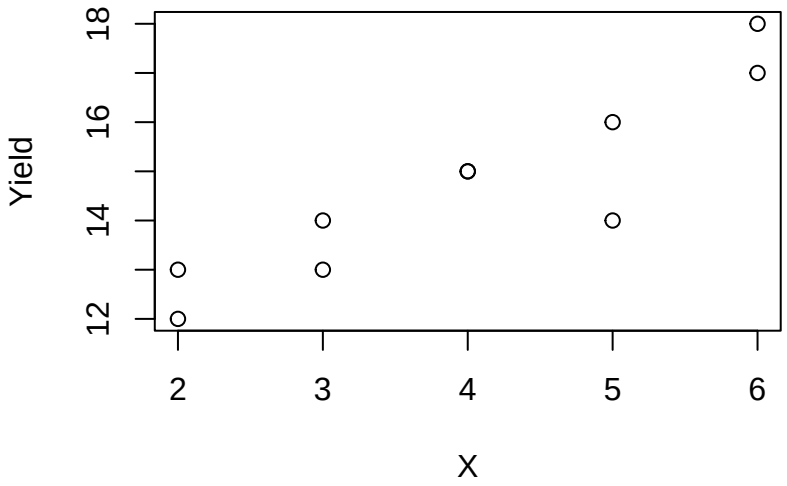
1 Corn Example: Simple Linear Regression

(Simple linear) regression is used to model the linear relationship between a numerical response variable and a single numerical predictor. In this example, corn yield is the response and fertilizer (X) is the predictor.

```
Corn <- read.csv("CH11_Corn.csv")
str(Corn)
```

```
## 'data.frame':    10 obs. of  2 variables:
##  $ Yield: int  12 13 13 14 15 15 14 16 17 18
##  $ X    : int  2 2 3 3 4 4 5 5 6 6
```

```
### Scatterplot
plot(Yield ~ X, data = Corn)
```



Note: The scatter plot is a summary plot that gives information about the data and relationship between variables, but may also provide information for checking assumptions.

1.1 Regression

```
Fit <- lm(Yield ~ X, data = Corn)
summary(Fit)
```

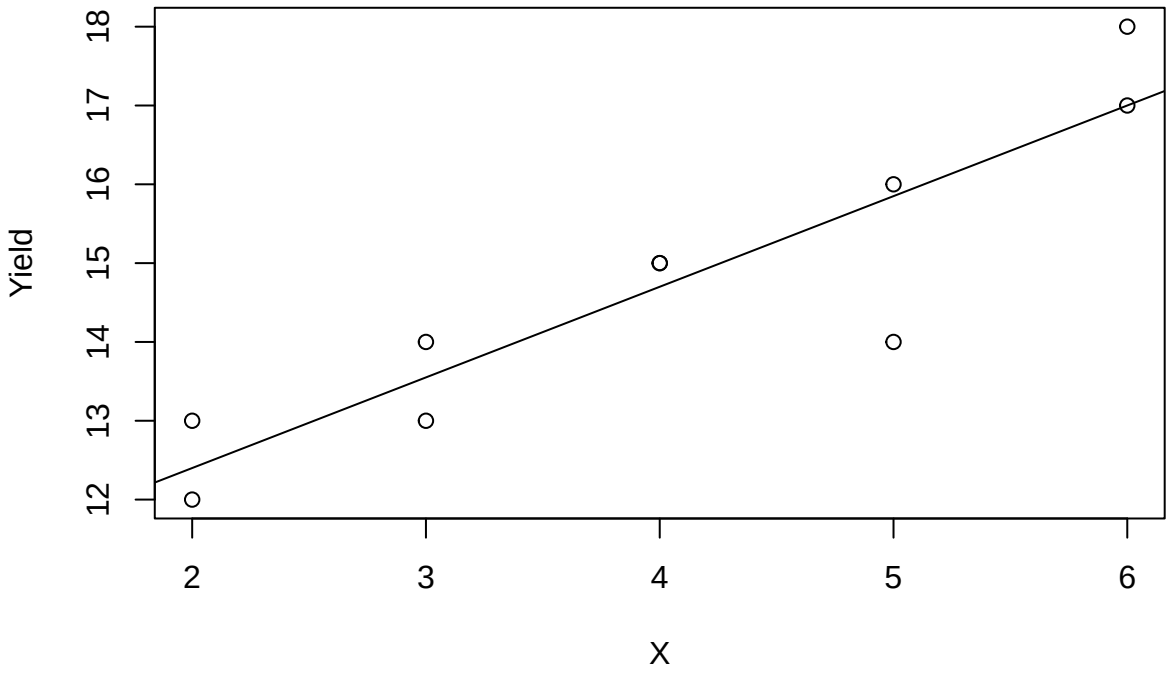
```
##
## Call:
## lm(formula = Yield ~ X, data = Corn)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -1.8500 -0.3000  0.2250  0.4125  1.0000
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  10.1000     0.7973   12.67 1.42e-06 ***
## X             1.1500     0.1879    6.12 0.000283 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.8404 on 8 degrees of freedom
## Multiple R-squared:  0.824,  Adjusted R-squared:  0.802
## F-statistic: 37.45 on 1 and 8 DF,  p-value: 0.0002832
```

```
Fit
```

```
##
## Call:
## lm(formula = Yield ~ X, data = Corn)
##
## Coefficients:
## (Intercept)            X
##       10.10         1.15
```

```
### Overlay fitted regression line in the scatter plot
plot(Yield ~ X, data = Corn)
abline(Fit, data = Corn)
```

```
## Warning in int_abline(a = a, b = b, h = h, v = v, untf = untf, ...): "data" is
## not a graphical parameter
```




```
### Confidence Intervals for intercept and slope
confint(Fit, level = 0.95)
```

```
##                 2.5 %    97.5 %
## (Intercept) 8.2615130 11.938487
## X           0.7166645  1.583336
```

```
### Inference for Mean response and Future Response
newdata <- data.frame(X = 5.5) # only one data point in the new dataset

# Confidence interval for mean value of Y at X=5.5
predict(Fit, newdata, interval = "confidence", level = 0.90)
```

```
##      fit      lwr      upr
## 1 16.425 15.70461 17.14539
```

```
# Prediction interval for future response value of Y at X=5.5
predict(Fit, newdata, interval = "predict",  level = 0.90)
```

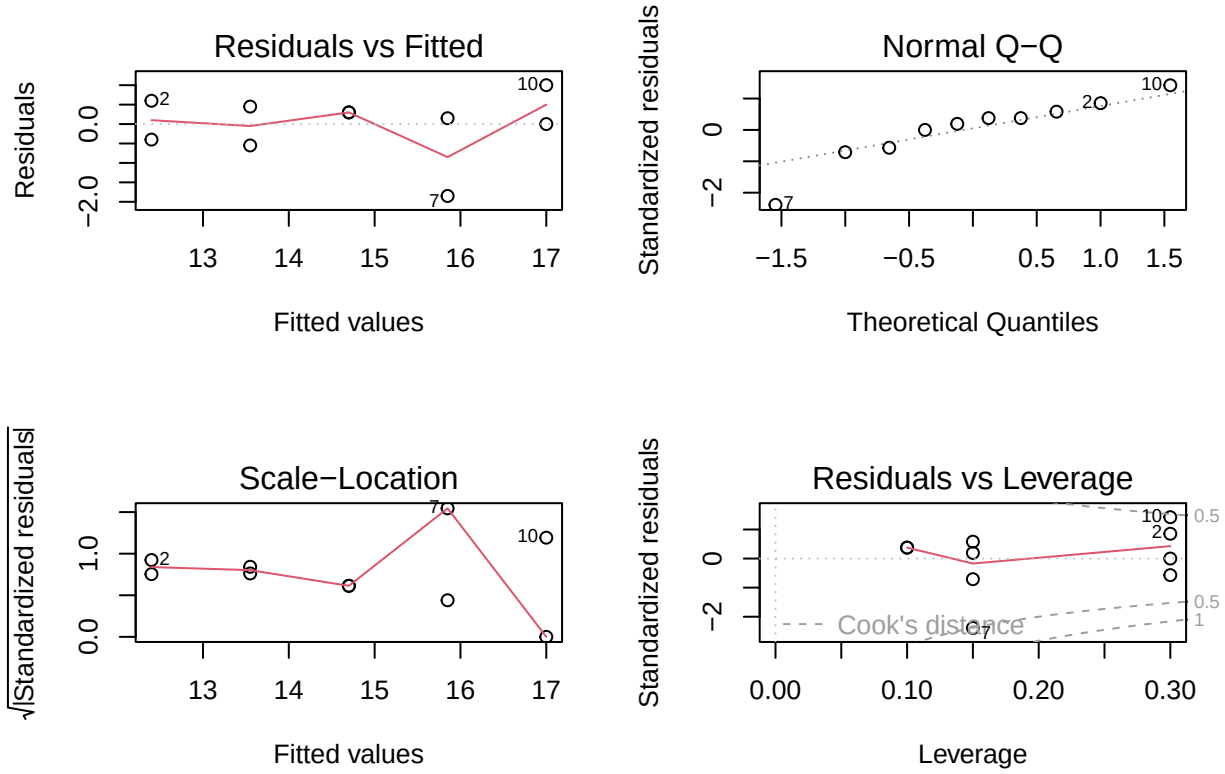
```
##      fit      lwr      upr
## 1 16.425 14.70421 18.14579
```

```
### ANOVA
anova(Fit)
```

```
## Analysis of Variance Table
##
## Response: Yield
##           Df Sum Sq Mean Sq F value    Pr(>F)
## X          1  26.45 26.4500  37.451 0.0002832 ***
## Residuals  8   5.65  0.7062
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

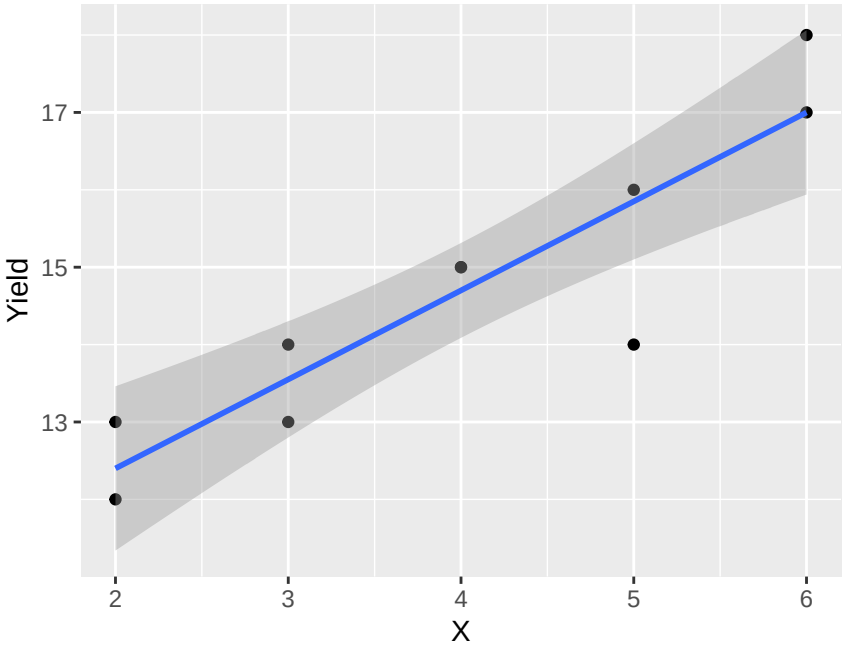
```
### diagnostic plots
par(mfrow=c(2,2))
```

```
plot(Fit)
```



1.2 tidyverse

```
library(tidyverse)
library(broom)
qplot(X, Yield, data = Corn) + geom_smooth(method = lm)
```



As we have seen before, tidy() from the broom package can be used to create "tidy" output.

glance() computes per-model statistics, such as R^2 and AIC.

The augment() method adds fitted values and residuals to the original data.

```
tidy(Fit)
```

```
## # A tibble: 2 x 5
##   term        estimate std.error statistic    p.value
##   <chr>          <dbl>     <dbl>     <dbl>      <dbl>
## 1 (Intercept)    10.1      0.797     12.7  0.00000142
## 2 X               1.15     0.188      6.12 0.000283
```

```
glance(Fit)
```

```
## # A tibble: 1 x 12
##   r.squ~1 adj.r~2 sigma stati~3 p.value    df logLik   AIC   BIC devia~4 df.re~5
##     <dbl>   <dbl> <dbl>   <dbl>   <dbl> <dbl>  <dbl> <dbl> <dbl>   <dbl>   <int>
## 1   0.824   0.802 0.840    37.5 2.83e-4     1  -11.3  28.7  29.6    5.65       8
## # ... with 1 more variable: nobs <int>, and abbreviated variable names
## #   1: r.squared, 2: adj.r.squared, 3: statistic, 4: deviance, 5: df.residual
```

```
head(augment(Fit))
```

```
## # A tibble: 6 x 8
##   Yield     X .fitted .resid  .hat .sigma .cooksd .std.resid
##   <int> <int>   <dbl>  <dbl> <dbl>  <dbl>   <dbl>      <dbl>
## 1    12     2    12.4 -0.400  0.3   0.880 0.0694      -0.569
## 2    13     2    12.4  0.600  0.3   0.857 0.156        0.853
## 3    13     3    13.6 -0.550  0.15  0.870 0.0445      -0.710
## 4    14     3    13.6  0.450  0.15  0.879 0.0298       0.581
## 5    15     4    14.7  0.300  0.1   0.890 0.00787      0.376
## 6    15     4    14.7  0.300  0.1   0.890 0.00787      0.376
```

```
rm(Corn, Fit, newdata)
```

2 Stopping Distance Example: Transformations

Transforming data can be used to satisfy model assumptions (linearity, equal variance and normality). In this example, we look at vehicle stopping distance versus speed (prior to braking).

```
Stop <-read.csv("CH11_StopDistance.csv")
str(Stop)
```

```
## 'data.frame':    50 obs. of  2 variables:
##  $ Speed: int  4 4 4 7 7 8 9 10 10 10 ...
##  $ Dist : int  2 10 16 4 22 16 10 18 26 34 ...
```

2.1 Model1 Dist ~ Speed

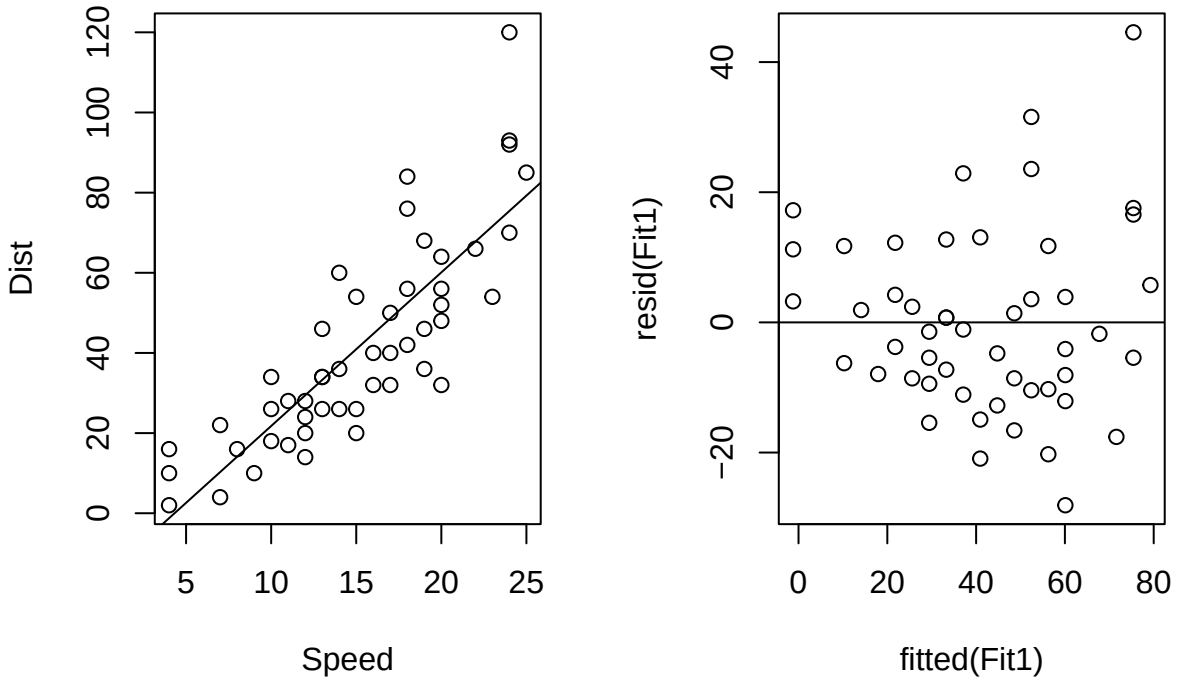
Note the "cornucopia" shape in the plot of resids vs fitted values. This indicates that regression assumptions are NOT satisfied.

```
Fit1 <- lm(Dist ~ Speed, data = Stop)
summary(Fit1)
```

```
##
```

```
## Call:
## lm(formula = Dist ~ Speed, data = Stop)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -28.098  -9.227  -1.599   9.856  44.571
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -16.5599     5.9790   -2.77  0.00795 **
## Speed         3.8329     0.3701   10.36 7.96e-14 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 14.32 on 48 degrees of freedom
## Multiple R-squared:  0.6908, Adjusted R-squared:  0.6844
## F-statistic: 107.2 on 1 and 48 DF,  p-value: 7.961e-14
```

```
par(mfrow=c(1,2))
plot(Dist ~ Speed, data = Stop)
abline(coef(Fit1))
plot(resid(Fit1) ~ fitted(Fit1))
abline(h = 0)
```



2.2 Model2 sqrt(Dist) ~ Speed

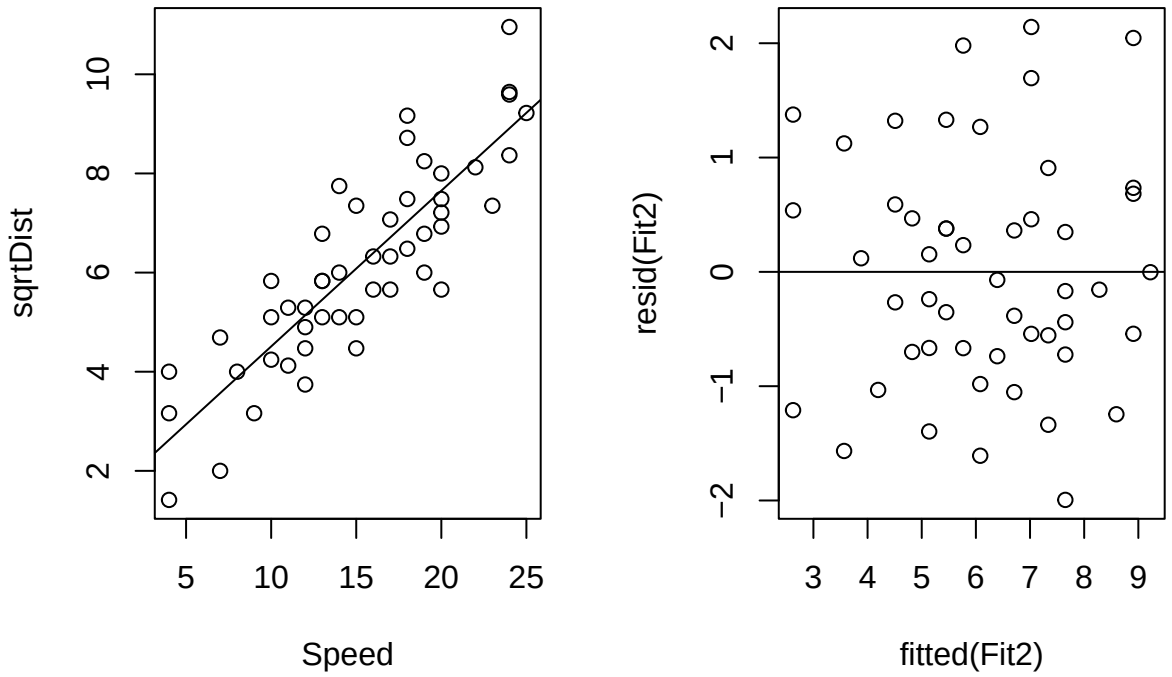
The plot of resids vs fitted for this model suggests that regression assumptions are now satisfied.

```
Stop$sqrtDist <- sqrt(Stop$Dist)
Fit2 <- lm(sqrtDist ~ Speed, data = Stop)
summary(Fit2)
```

```
##
```

```
## Call:
## lm(formula = sqrtDist ~ Speed, data = Stop)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -1.9947 -0.6922 -0.1130  0.5767  2.1420
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  1.36730    0.42748   3.199  0.00245 **
## Speed        0.31421    0.02646  11.874 6.84e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.024 on 48 degrees of freedom
## Multiple R-squared:  0.746,  Adjusted R-squared:  0.7407
## F-statistic:   141 on 1 and 48 DF,  p-value: 6.841e-16
```

```
par(mfrow=c(1,2))
plot(sqrtDist ~ Speed, data = Stop)
abline(coef(Fit2))
plot(resid(Fit2) ~ fitted(Fit2))
abline(h = 0)
```



2.3 Model3 Dist ~ Speed2

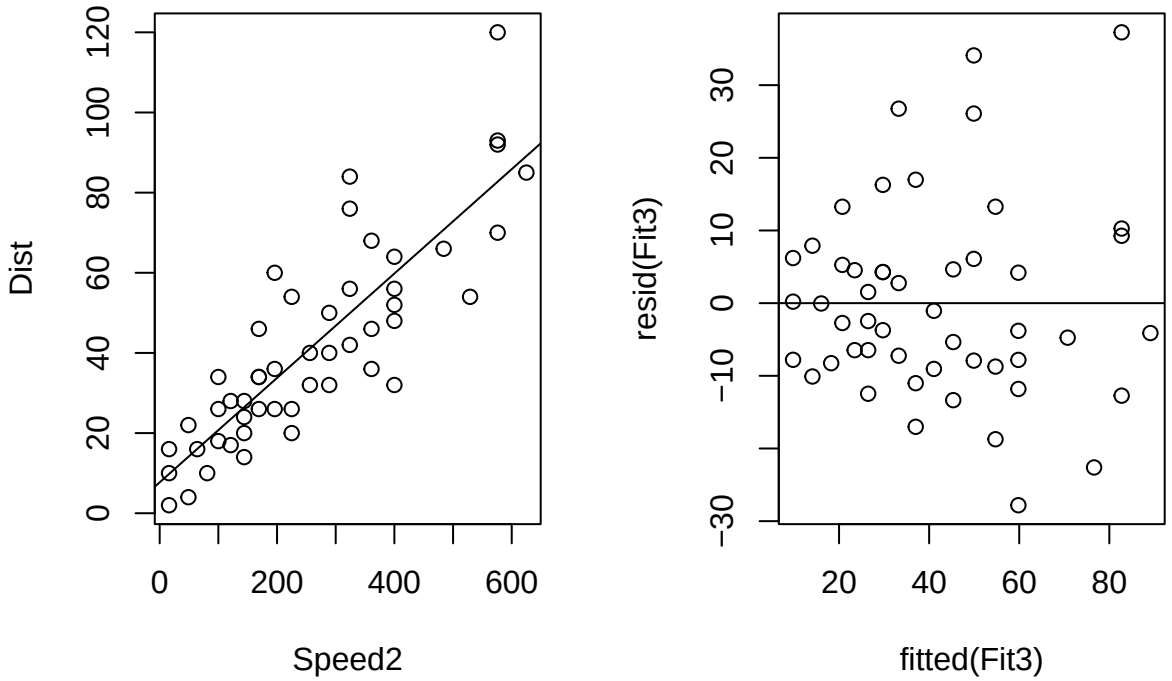
Note the "megaphone" shape in the plot of resids vs fitted values. This indicates that regression assumptions are NOT satisfied.

```
Stop$Speed2 <- Stop$Speed*Stop$Speed
Fit3 <- lm(Dist ~ Speed2, data = Stop)
summary(Fit3)
```

```
##
```

```
## Call:
## lm(formula = Dist ~ Speed2, data = Stop)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -27.809  -8.173  -2.601   5.883  37.267
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  7.71108    3.57651   2.156   0.0361 *
## Speed2       0.13025    0.01159  11.241 4.79e-15 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 13.51 on 48 degrees of freedom
## Multiple R-squared:  0.7247, Adjusted R-squared:  0.719
## F-statistic: 126.4 on 1 and 48 DF,  p-value: 4.792e-15
```

```
par(mfrow=c(1,2))
plot(Dist ~ Speed2, data = Stop)
abline(coef(Fit3))
plot(resid(Fit3) ~ fitted(Fit3))
abline(h = 0)
```




```
rm(Stop, Fit1, Fit2, Fit3)
```

3 Corn Example: Lack of Fit Test

```
Corn <- read.csv("CH11_Corn.csv")
str(Corn)
```

```
## 'data.frame':    10 obs. of  2 variables:
##  $ Yield: int  12 13 13 14 15 15 14 16 17 18
##  $ X    : int  2 2 3 3 4 4 5 5 6 6
```

3.1 Regression

```
RegFit <- lm(Yield ~ X, data = Corn)
anova(RegFit)
```

```
## Analysis of Variance Table
##
## Response: Yield
##           Df Sum Sq Mean Sq F value    Pr(>F)
## X          1  26.45 26.4500  37.451 0.0002832 ***
## Residuals  8   5.65  0.7062
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

3.2 ANOVA

```
ANOVAFit <- lm(Yield ~ as.factor(X), data = Corn)
anova(ANOVAFit)
```

```
## Analysis of Variance Table
##
## Response: Yield
##              Df Sum Sq Mean Sq F value  Pr(>F)
## as.factor(X)  4   28.6    7.15  10.214 0.01267 *
## Residuals     5    3.5    0.70
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

3.3 Lack of Fit test

```
anova(RegFit, ANOVAFit)
```

```
## Analysis of Variance Table
##
## Model 1: Yield ~ X
## Model 2: Yield ~ as.factor(X)
##   Res.Df  RSS Df Sum of Sq      F Pr(>F)
## 1      8 5.65
## 2      5 3.50  3      2.15 1.0238 0.4564
```