Midterm 1

Thu Nguyen

05 May, 2019

Libraries

```r
library(ggplot2)
library(tidyverse)
library(ggthemes)
```

Problem 1: Student vs. Wilcoxon

Suppose we have a numerical sample of size n which we assume was generated iid from an underlying distribution F, unknown with a well-defined mean μ.

- Student's t-test is a test about the mean: Is μ equal to a given value μ_0?
- Wicoxon's signed-rank test is a test for symmetry: Is F symmetric about a given μ_0?

That being said, the t-test can be used to test whether F is symmetric about μ_0, based on the fact that 'symmetric about μ_0' implies that 'the mean is equal to μ_0'. However, the two are not equivalent, so that the t-test is not consistent against all alternatives. Conversely, for the signed-rank test to be useful as a test about the mean, we need to assume that F is symmetric about its mean. With this additional (and nontrivial) assumption on F, testing for symmetry about μ_0 is equivalent to testing whether the mean equal to μ_0. (Convince yourself of that.) In what follows, we place ourselves in that situation, so that we can directly compare the two tests. There is some theory on that. For example, it is known that when F is a normal distribution, in which case the t-test achieves the most power asymptotically (meaning in the large-sample limit), the signed-rank test performs almost as well. We want to evaluate that with simulations.

Since both tests are scale-free, we may take that F to be the normal distribution with mean μ and variance 1. We consider the two-sided setting where we test $\mu = 0$ versus $\mu_0 \neq 0$. For each $n \in \{10, 20, 50, 100, 20, 500\}$ do the following. For each μ in a grid of your choice, denoted $\mathcal{M}$ and of size 10, generate $X_1, \ldots, X_n \sim \mathcal{N}(\mu, 1)$ and apply the t-test and signed-rank test, both set at level $\alpha = 0.10$. Record whether they reject or not. Repeat this $B = 1,000$ times and compute the fraction of times each test rejects. This estimates the power of each test against the alternative μ. The end result is a plot where these estimated power curves for each of these two tests are overlaid. Use colors and a legend to identify the two curves. Make sure to choose $\mathcal{M}$ so that we can see the power go from about α to about 1, zooming in on the action.

Given $F \sim N(\mu, 1)$, we want to study the powers of *student t-test* vs. *signed rank Wilcoxon test*, where

$$power = P(reject | H_1 : true)$$

We can study the powers of the 2 tests by simulations: calculating the proportions of rejects per simulation.

$$H_0 : \mu = \mu_0$$
$$H_1 : \mu \neq \mu_0$$

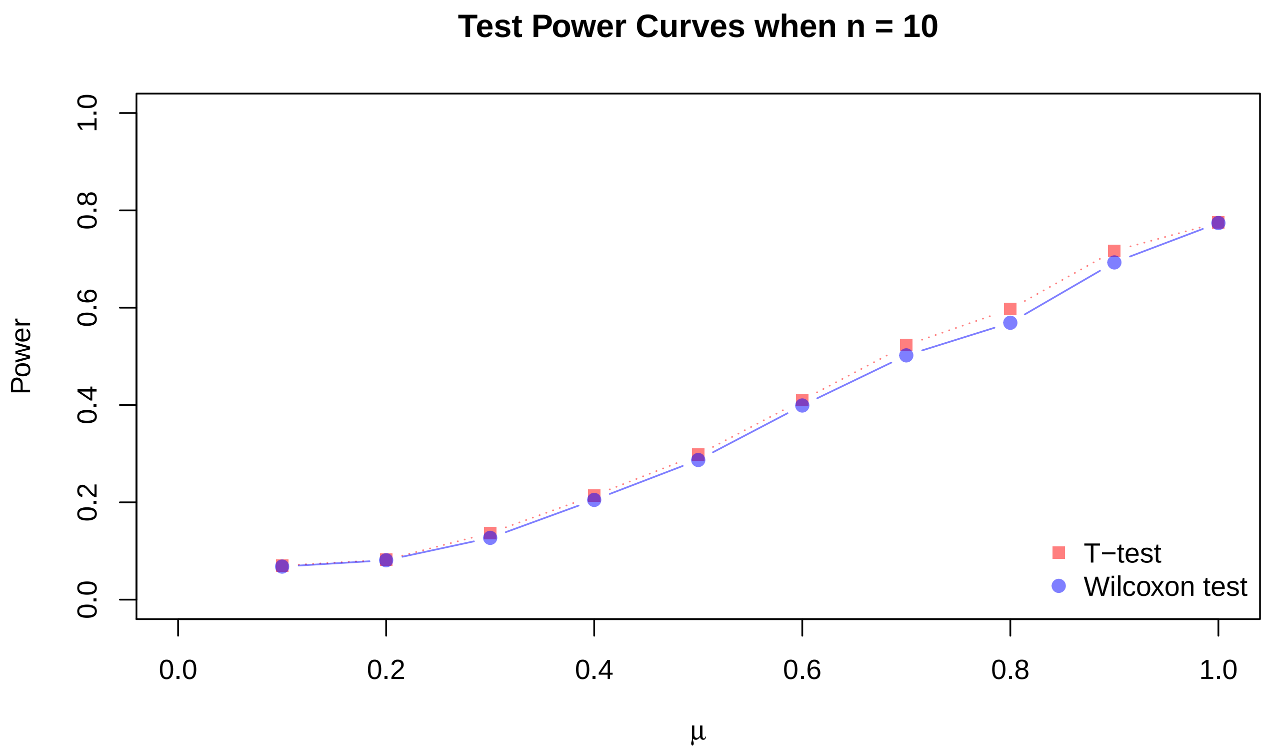
Number of simulation: $B = 1000$, with $\mu \in \mathcal{M} = \{.1, .2, \ldots, .9, 1\}$, and confidence level $\alpha = .1$.

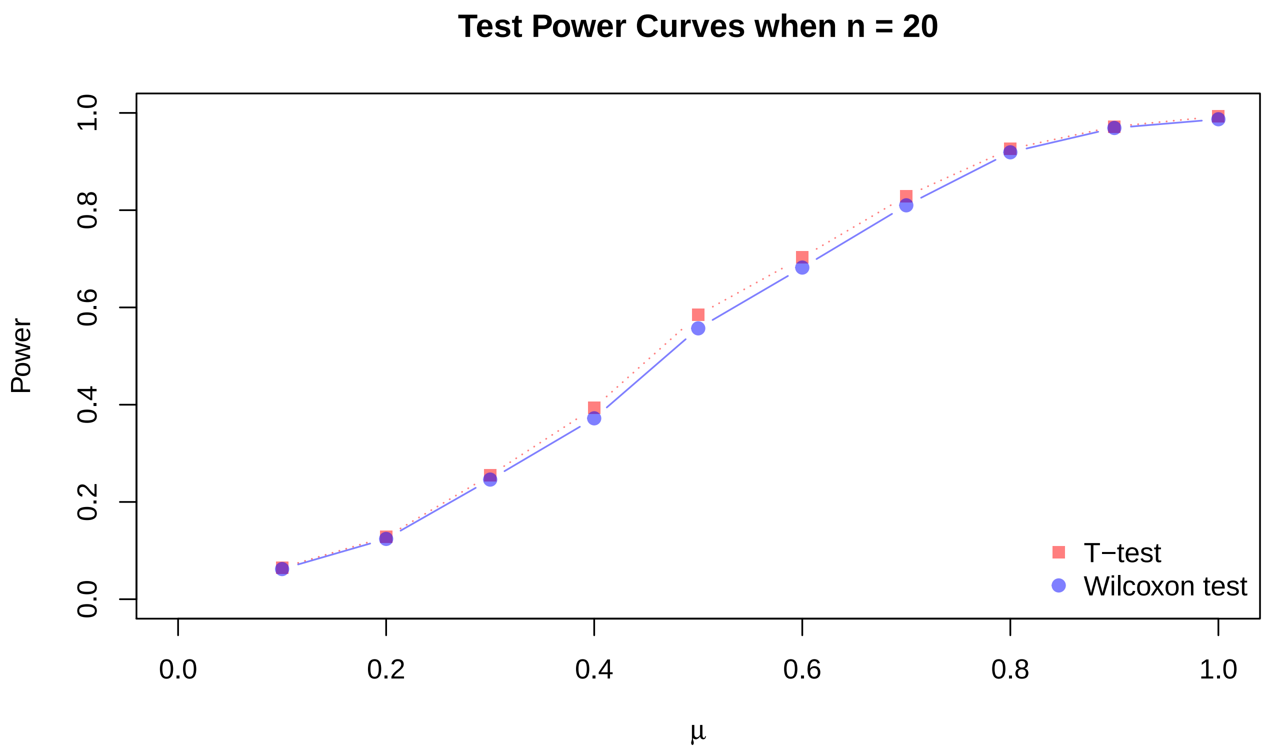
Set up

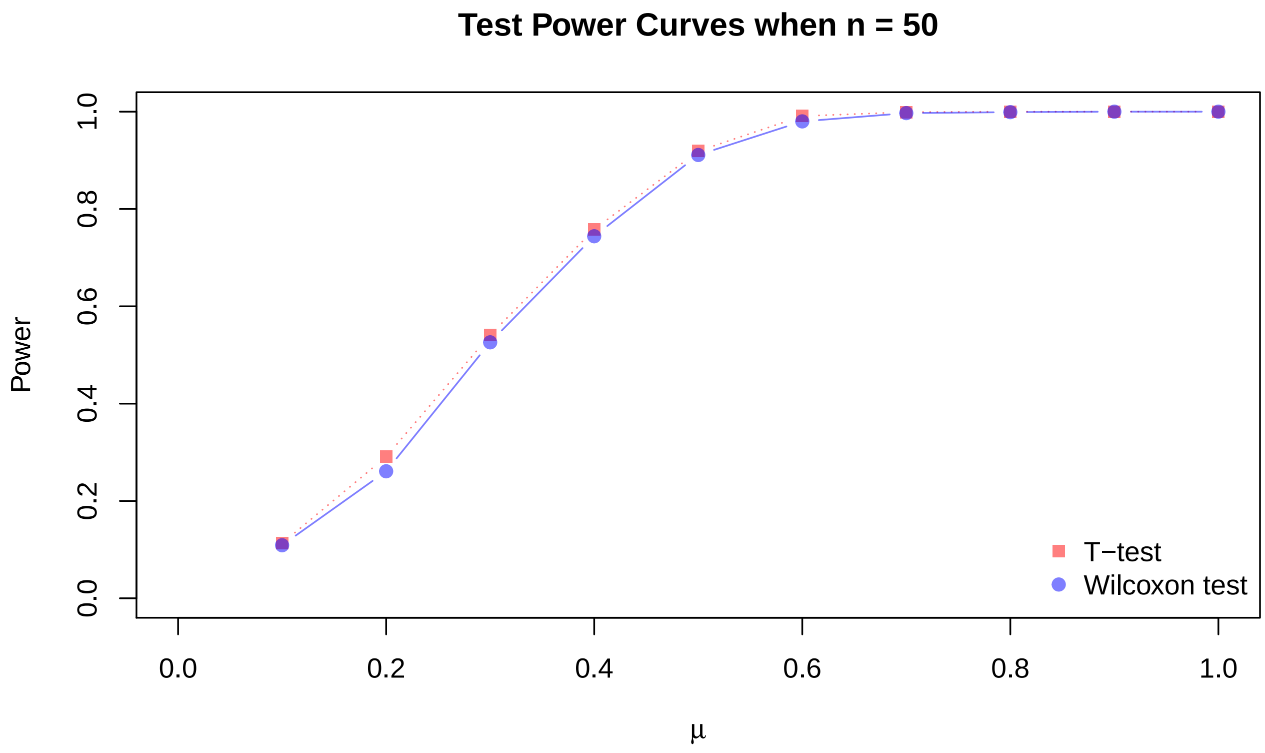
```r
# Different sample size n
ns <- c(10, 20, 50, 100, 200, 500)
# B: number of simulations
B <- 1000
# M: sequence of different values for mu
mu = seq(0.1, 1, .1)
# Confidence level
alpha = .1
```

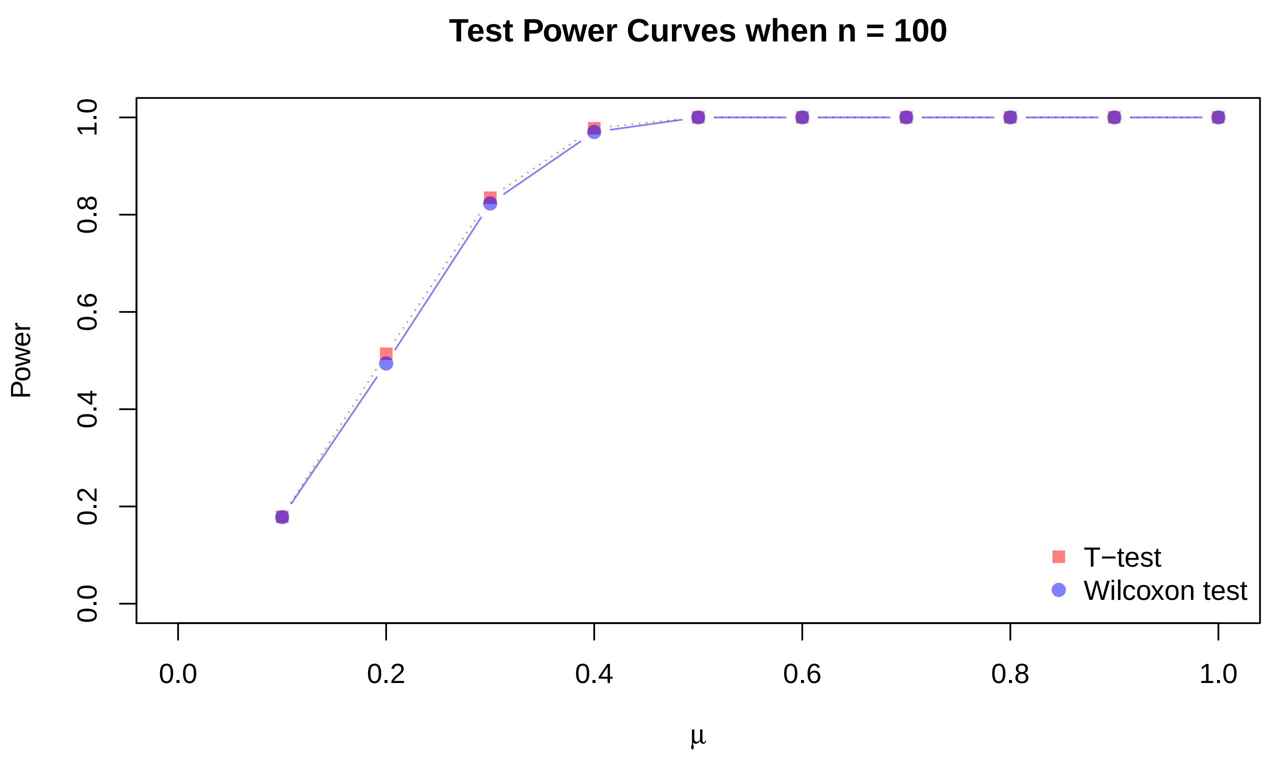
Power curves for *Student's t-test* and *Wilcoxon's signed-rank test*

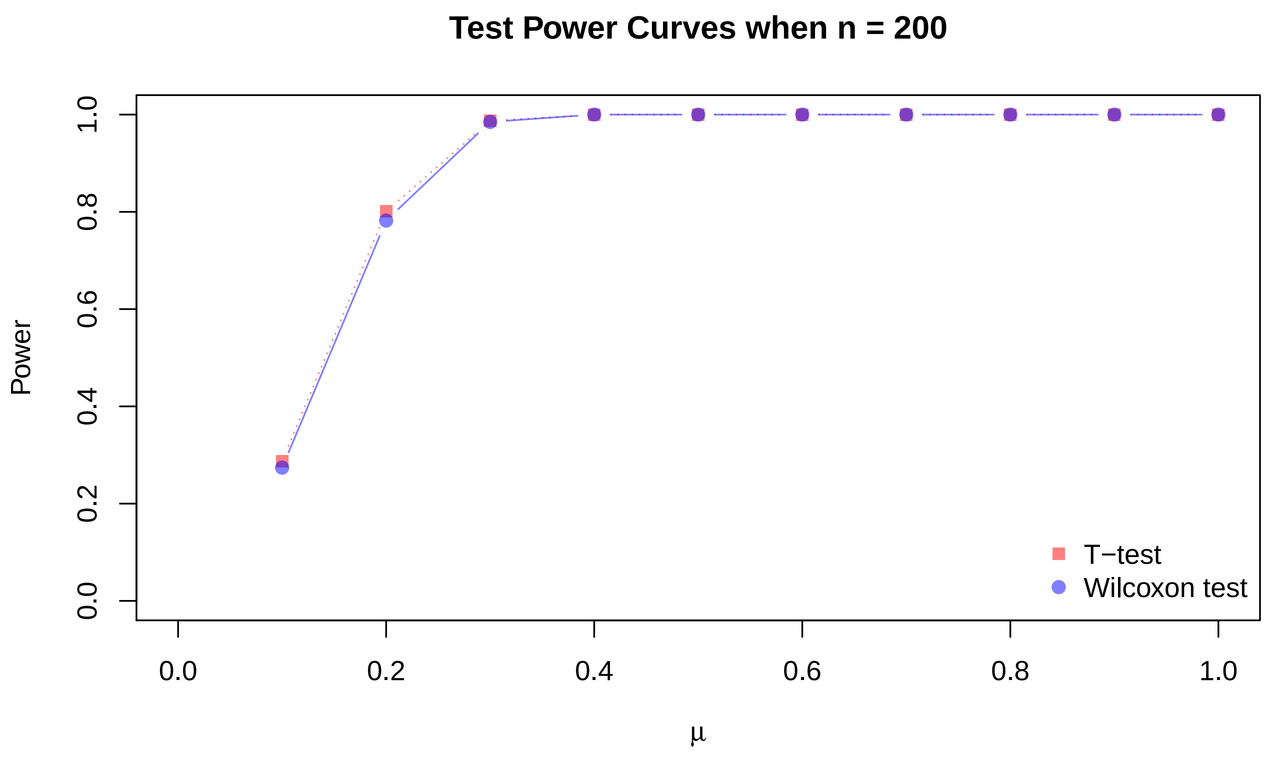
```r
par(mfrow = c(1,1))

# Loop of different sample size n
for (j in 1:length(ns)) {
  # Assign specific n
  n <- ns[j]

  # t: vector of powers for t-test
  t <- numeric(length(mu))
  # w: vector of powers for wilcoxon-test
  w <- numeric(length(mu))

  # Loop of different values for mu
  for (i in 1:length(mu)) {
    # Assign specific mu
    mu_0 <- mu[i]

    # ttest: vector to store of Rejects/NotRejects for t-test
    ttest <- numeric(B)
    # wtest: vector to store of Rejects/NotRejects for wilcoxon-test
    wtest <- numeric(B)

    # Loop of 1000 simulations
    for (b in 1:B) {
      # Generate sequence of n random variables
      x <- rnorm(n, mu_0, 1)
      # Test-statistic for t-test
      tstat <- t.test(x)$p.value
      # t-test: if yes
      if (tstat < alpha/2) {
        ttest[b] <- 1
      }
      # Test-statistic for wilcoxon-test
      wstat <- wilcox.test(x)$p.value
      # wilcoxon-test: if yes
      if (wstat < alpha/2) {
        wtest[b] <- 1
      }
    }

    # Power of t-test
    t[i] <- sum(ttest)/length(ttest)
    # Power of wilcoxon-test
    w[i] <- sum(wtest)/length(wtest)
  }

  # Plot's label
  mes = paste('Test Power Curves when n =', n)

  # Plot
  plot(t~mu, xlab = expression(mu), ylab = 'Power',
       type = 'b', lty = 3, col = rgb(1,0,0,.5), lwd=1, pch=15,
       xlim = c(0,1), ylim = c(0,1),
       main = mes)
  lines(w~mu, col = rgb(0,0,1,.5), lwd = 1, pch = 19, type ='b')

  # Legend for plot
  legend('bottomright',
         legend = c('T-test', 'Wilcoxon test'),
         col = c(rgb(1,0,0,.5), rgb(0,0,1,.5)), pch = c(15, 19), bty = 'n')
}
```

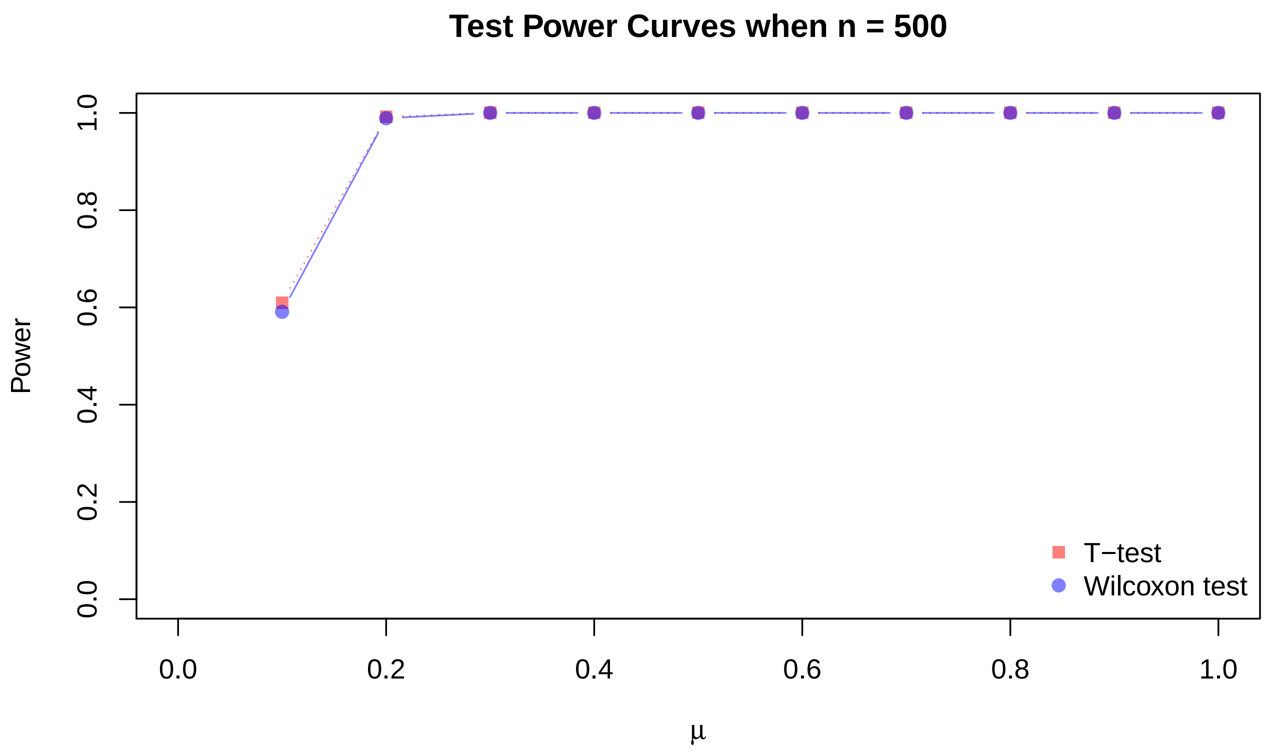



Test Power Curves when n = 20
Power
μ
T−test
Wilcoxon test
0.0
0.2
0.4
0.6
0.8
1.0


Test Power Curves when n = 50
Power
μ
T−test
Wilcoxon test
0.0
0.2
0.4
0.6
0.8
1.0


Test Power Curves when n = 100
Power
μ
0.0
0.2
0.4
0.6
0.8
1.0
T−test
Wilcoxon test


Test Power Curves when n = 200
Power
μ
0.0
0.2
0.4
0.6
0.8
1.0
T−test
Wilcoxon test


Test Power Curves when n = 500
Power
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0
μ
T−test
Wilcoxon test

Problem 2: Fungi in brassica plants

Consider the following article about how different brassica plants are affected by different types of Rhizoctonia fungi.1 Read enough of the article to understand the premise and the main findings. Otherwise, we will focus on the data given in Table 6 on how different brassica species are affected by different types of Rhizoctonia fungi.

A. Write a function `tableObsExp(dat)` taking in a two-column data frame, with each column representing a factor, and then outputting a table of observed and expected (under no association) of counts - similar to what Table 6 in that article looks like.
B. Enter the observed counts from Table 6 (likely by hand, as the data do not seem directly downloadable) and apply your function to recover a similar table.
C. Continuing with the same dataset, produce a couple of plots using functions in the ggplot2.
D. Finally, ask a question and formalize it into a hypothesis testing problem. Perform a test and offer some brief comments.

A. Function `tableObsExp(dat)`

Given a 2-column table, each of which representing a factor, we are testing for if the 2 factors are Independent of each other, which is equivalent to the claim that they have no association. Assuming they are independent, using the laws of Probability for joint density, we have:

$$\mathbf{E}_{ij} = \frac{1}{n} R_i C_j$$

where E_{ij} is the expected count for the i^{th} row, j^{th} column cell, n is the total number of observations, R_i is the total number of observations from row i, and C_j is the total number of observations from row j.

The function `tableObsExp(dat)` will return (1) a table of observed counts, (2) a table of expected counts, and (3) a vector of all the counts, in the order of Observed counts and Expected counts, to be used for part D.

```r
# Part A ---------------------------------------------------------------

tableObsExp <- function(dat) {
  # Observed counts matrix
  observedMatrix <- table(dat)
  print('Observed counts from data')
  print(observedMatrix)

  #--------------------------------------------
  # Expected counts

  # Get unique names per column
  col1 <- unique(dat[,1])
  col2 <- unique(dat[,2])

  # Number of unique features per column from 2-column dataframe
  n1 <- length(col1)
  n2 <- length(col2)

  # Total obsevations
  n <- nrow(dat)

  # Get total counts for each row and column
  temp <-addmargins(observedMatrix)
  sumCol <- temp[nrow(temp), 1:(ncol(temp)-1)]
  sumRow <- temp[1:(nrow(temp)-1), ncol(temp)]

  # Vector of Expected counts: E_ij = (R_i * C_j) / n
  # Empty vector to store all expected counts
  exptvector <- vector()
  # Step 1: loop through each row
  for (i in 1:n1) {
    # Step 2: loop through each column
    for (j in 1:n2) {
      # Expected count
      Eij <- round(sumRow[[i]] * sumCol[[j]] / n,2)
      # Add to exptvector to keep track of all counts
      exptvector <- append(exptvector, Eij)
    }
  }

  # Transform vector to matrix form
  expectedMatrix <- matrix(exptvector, nrow = 7, byrow = T)

  # Rename rows and columns
  rownames(expectedMatrix) <- col1
  colnames(expectedMatrix) <- col2


  print(cat('\n'))
  # Print
  print('Expected counts table under Null Hypothesis: Variables are Independent')
  print(expectedMatrix)

  return(c(observedMatrix, expectedMatrix))
}
```

B. Apply on Table 6 from the article

Prepare the data from Table 6, arrange into a 2-column data.frame: Column 1: Plants, Column 2: AGs.

```r
# Part B -------------------------------------------------------------

# Gather table of counts from Table 6
# Features: col1 for Plants, col2 for AG
col1 <- c('Mustard cabbage', 'White cabbage', 'Chinese flowering cabbage',
          'Chinese cabbage', 'Pak choi', 'Broccoli', 'Turnip cabbage')
col2 <- c('1-IA', '1-IB', '1-ID', '2-2', '1-IG', '4-HGI', '7', 'A', 'Fc')

# Counts matrix as seen from Table 6
countsmatrix <- matrix(c(14,0,5,0,0,1,0,1,0,
                         8,8,0,0,0,3,0,2,0,
                         9,0,12,1,1,4,0,0,0,
                         0,1,0,0,0,3,0,0,1,
                         10,0,0,0,0,0,0,1,0,
                         0,4,0,0,0,0,0,0,0,
                         3,0,0,4,0,1,1,0,0),
                       nrow = 7, byrow = T)
```

```r
# Default data.frame, to be added in
df <- data.frame()

# Step 1: loop through column 1
for (i in 1:length(col1)) {
  # Get feature name
  featname <- col1[i]
  # Get distribution of column 1 feature i
  counts <- countsmatrix[i,]
  # Total occurances of column 1 feature i
  n1 <- sum(counts)
  # Generate cells of column 1 feature i * n1 times
  col1feat <- rep(featname, n1)
  # Step 2: loop through column 2
  # Generate respective counts for column 2 features
  col2feat <- vector()
  for (j in 1:length(col2)) {
    if (counts[j] > 0) {
      col2name <- col2[j]
      temp <- rep(col2name, counts[j])
      col2feat <- append(col2feat, temp)
    }
  }
  temp_df <- data.frame(cbind(col1feat, col2feat))
  df <- rbind(df, temp_df)
}
```

Applying function `tableObsExp(dat)` onto the data.frame above

```r
tableObsExp(df)[0]
```

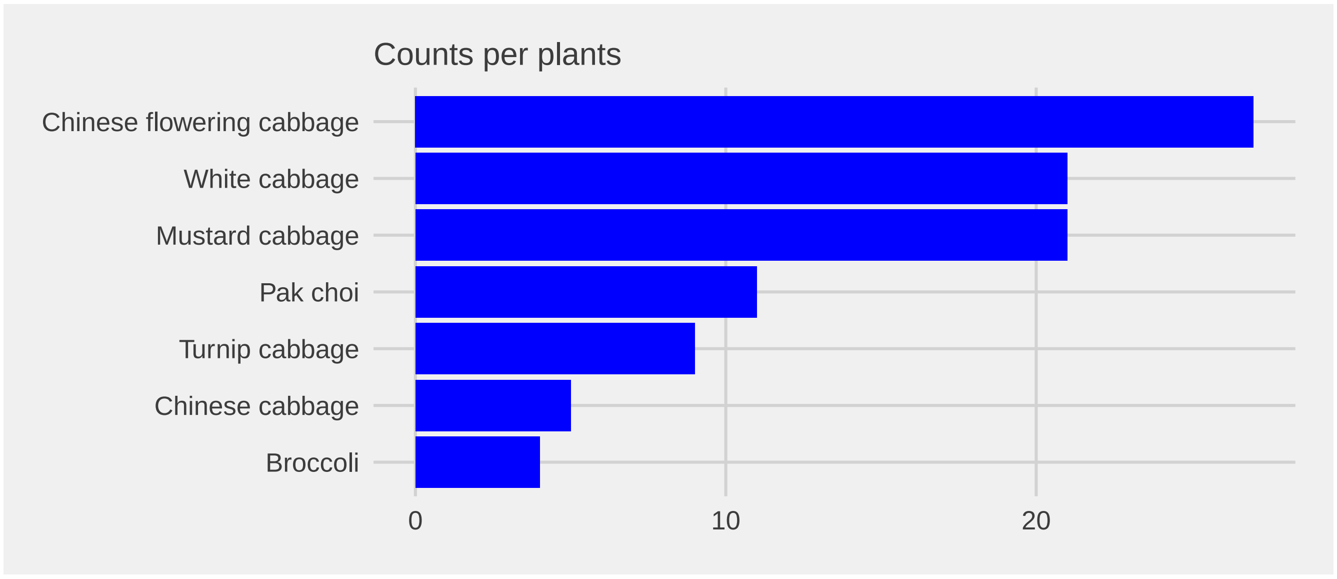
```
## [1] "Observed counts from data"
##                              col2feat
## col1feat                       1-IA 1-ID 4-HGI  A 1-IB 1-IG 2-2 Fc  7
##   Mustard cabbage                14    5     1  1    0    0   0  0  0
##   White cabbage                   8    0     3  2    8    0   0  0  0
##   Chinese flowering cabbage       9   12     4  0    0    1   1  0  0
##   Chinese cabbage                 0    0     3  0    1    0   0  1  0
##   Pak choi                       10    0     0  1    0    0   0  0  0
##   Broccoli                        0    0     0  0    4    0   0  0  0
##   Turnip cabbage                  3    0     1  0    0    0   4  0  1
##
## NULL
## [1] "Expected counts table under Null Hypothesis: Variables are Independent"
##                              1-IA 1-ID 4-HGI    A 1-IB  2-2 1-IG   Fc    7
## Mustard cabbage              9.43 3.64  2.57 0.86 2.79 0.21 1.07 0.21 0.21
## White cabbage                9.43 3.64  2.57 0.86 2.79 0.21 1.07 0.21 0.21
## Chinese flowering cabbage 12.12 4.68  3.31 1.10 3.58 0.28 1.38 0.28 0.28
## Chinese cabbage              2.24 0.87  0.61 0.20 0.66 0.05 0.26 0.05 0.05
## Pak choi                     4.94 1.91  1.35 0.45 1.46 0.11 0.56 0.11 0.11
## Broccoli                     1.80 0.69  0.49 0.16 0.53 0.04 0.20 0.04 0.04
## Turnip cabbage               4.04 1.56  1.10 0.37 1.19 0.09 0.46 0.09 0.09
```

```
## numeric(0)
```

C. Plots

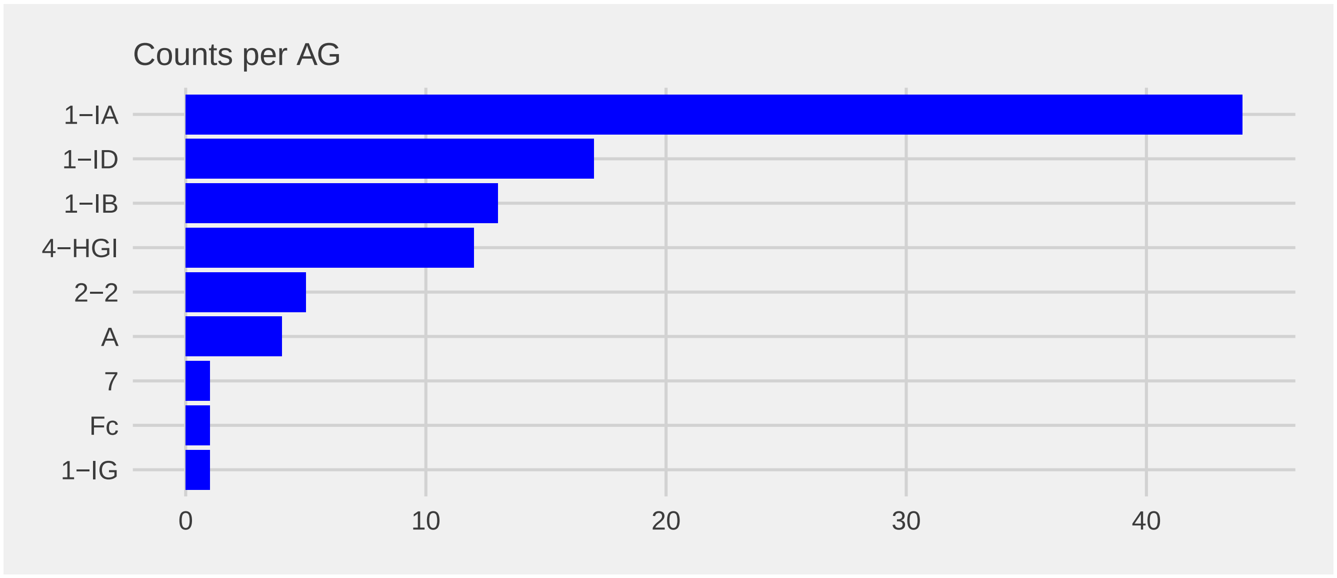
Plot 1: Counts per plants

```r
# Part C -------------------------------------------------------------
# Plot 1: Counts per plants
df %>%
  select(col1feat) %>%
  group_by(col1feat) %>%
  summarise(counts = n()) %>%
  arrange(desc(counts)) %>%
  ggplot(aes(x = reorder(col1feat, counts), y = counts)) +
  geom_col(fill = 'blue') +
  coord_flip() +
  labs(subtitle = 'Counts per plants') +
  theme_fivethirtyeight()
```



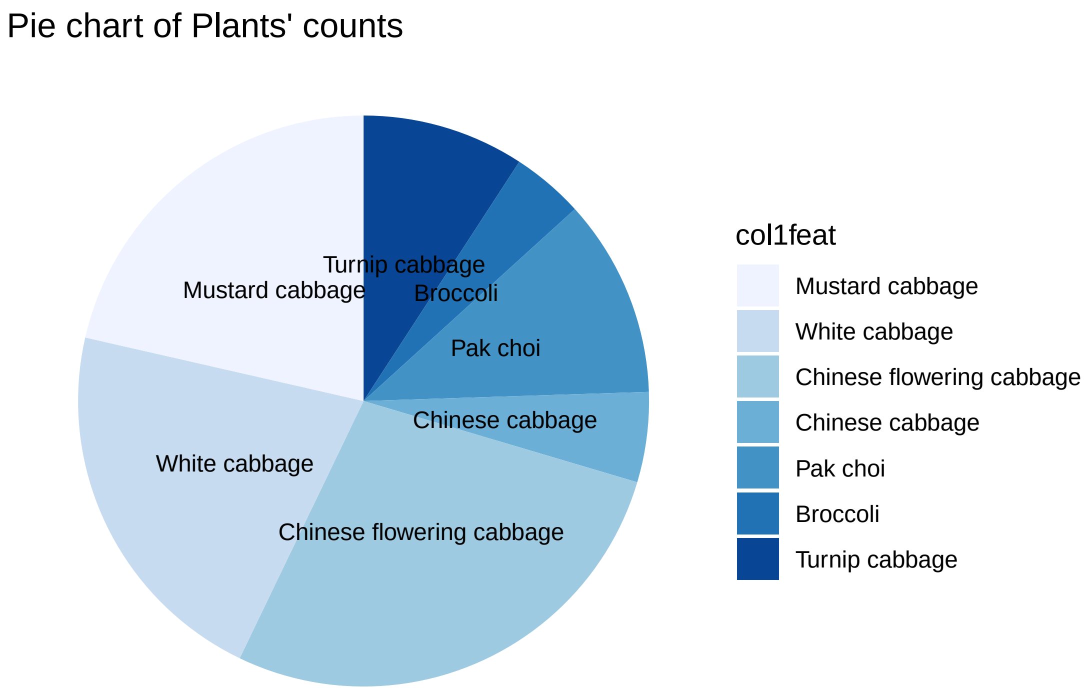
Plot 2: Counts per AG

```r
# Plot 2: Counts per AG
df %>%
  select(col2feat) %>%
  group_by(col2feat) %>%
  summarise(counts = n()) %>%
  arrange(desc(counts)) %>%
  ggplot(aes(x = reorder(col2feat, counts), y = counts)) +
  geom_col(fill = 'blue') +
  coord_flip() +
  labs(subtitle = 'Counts per AG') +
  theme_fivethirtyeight()
```



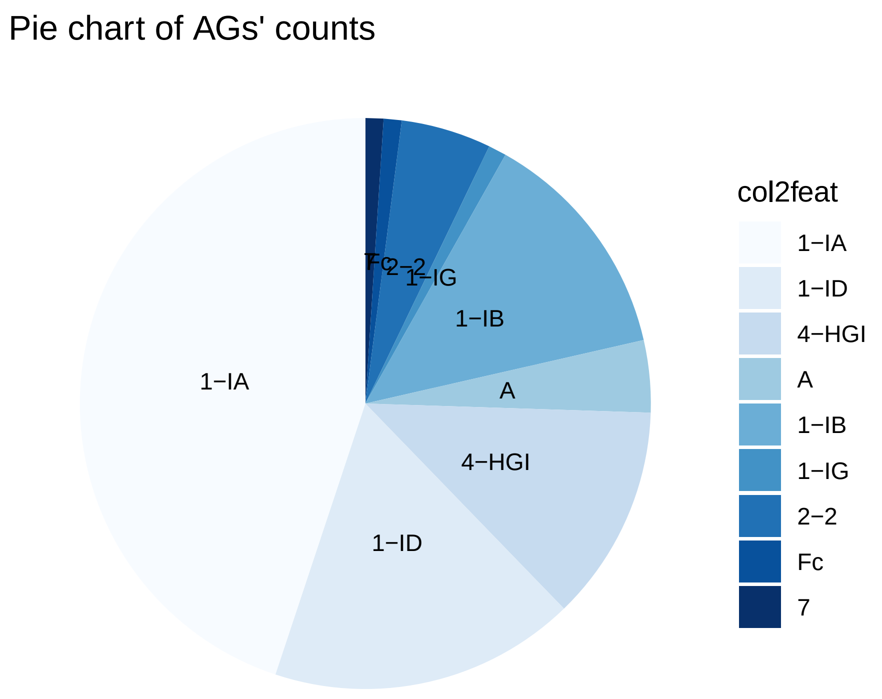
Plot 3: Pie chart for plants

```r
df %>%
  select(col1feat) %>%
  group_by(col1feat) %>%
  summarise(Percent = n()/nrow(.) * 100) %>%
  arrange(desc(Percent)) %>%
  ggplot(aes(x = '', y = Percent, fill = col1feat)) +
  geom_bar(width = 1, stat = 'identity') +
  scale_y_continuous(breaks = round(cumsum(rev(df$Percent)), 1)) +
  coord_polar("y", start = 0) +
  geom_text(aes(label = col1feat), col = 'black', size=3, position = position_stack(vjust = 0.5)) +
  scale_fill_brewer(palette="Blues") +
  labs(title = "Pie chart of Plants' counts") +
  theme_fivethirtyeight() +
  theme_void()
```



Plot 4: Pie chart for AG

```
df %>%
  select(col2feat) %>%
  group_by(col2feat) %>%
  summarise(Percent = n()/nrow(.) * 100) %>%
  arrange(desc(Percent)) %>%
  ggplot(aes(x = '', y = Percent, fill = col2feat)) +
  geom_bar(width = 1, stat = 'identity') +
  scale_y_continuous(breaks = round(cumsum(rev(df$Percent)), 1)) +
  coord_polar("y", start = 0) +
  geom_text(aes(label = col2feat), col = 'black', size=3, position = position_stack(vjust = 0.5)) +
  scale_fill_brewer(palette="Blues") +
  labs(title = "Pie chart of AGs' counts") +
  theme_fivethirtyeight() +
  theme_void()
```



D. Hypothesis test

$$H_0 : \text{No association} \iff \text{The occurance of AGs is independent of the plants}$$
$$H_1 : \text{There is association} \iff \text{The occurance of AGs is not independent of the plants}$$

To test for independency, we are going to use the Chi-squared test. Below are 2 alternatives to implementing the test: (1) applying the `chisq.test()` function directly, and (2) calculating the test-statistic and the p-value from the tables from part A.

1. Directly apply the function `chisq.test()`

```r
chisq.test(table(df))
```

```
##
##  Pearson's Chi-squared test
##
## data:  table(df)
## X-squared = 153.06, df = 48, p-value = 6.772e-13
```

2. Calculate the test-statistic, and p-value

Under the Chi-squared test, the test-statistic is calculated as:

$$D = \sum_i \sum_j \frac{(O_{ij} - E_{ij})^2}{E_{ij}}$$

where O_{ij} is the observed counts, and E_{ij} is the expected counts for the i^{th} row, j^{th} column.

```r
invisible(capture.output(tables <- tableObsExp(df)))
obsCounts <- tables[1:63]
expCounts <- tables[64:length(tables)]
D <- sum((obsCounts - expCounts)^2/expCounts)
D
```

```
## [1] 153.5797
```

Given 7 rows and 9 columns, the degree of freedom is: $df = (r-1)(c-1) = 48$

```r
pchisq(D, 48, lower = F)
```

```
## [1] 5.641814e-13
```

Comment:

1. The differences in the numerical results from 2 ways are due to rounding errors: the function `tableObsExp()` when calculating the expected counts rounded up the numbers to 2 decimal places.
2. Given such very small p-value, for any reasonable Confidence level of α, the test would reject the Null hypothesis: it appears that the occurance of AGs is not independent of the plants, and that there is an association between plants and AGs.