Practicality of Statistical Inference

Anandu R

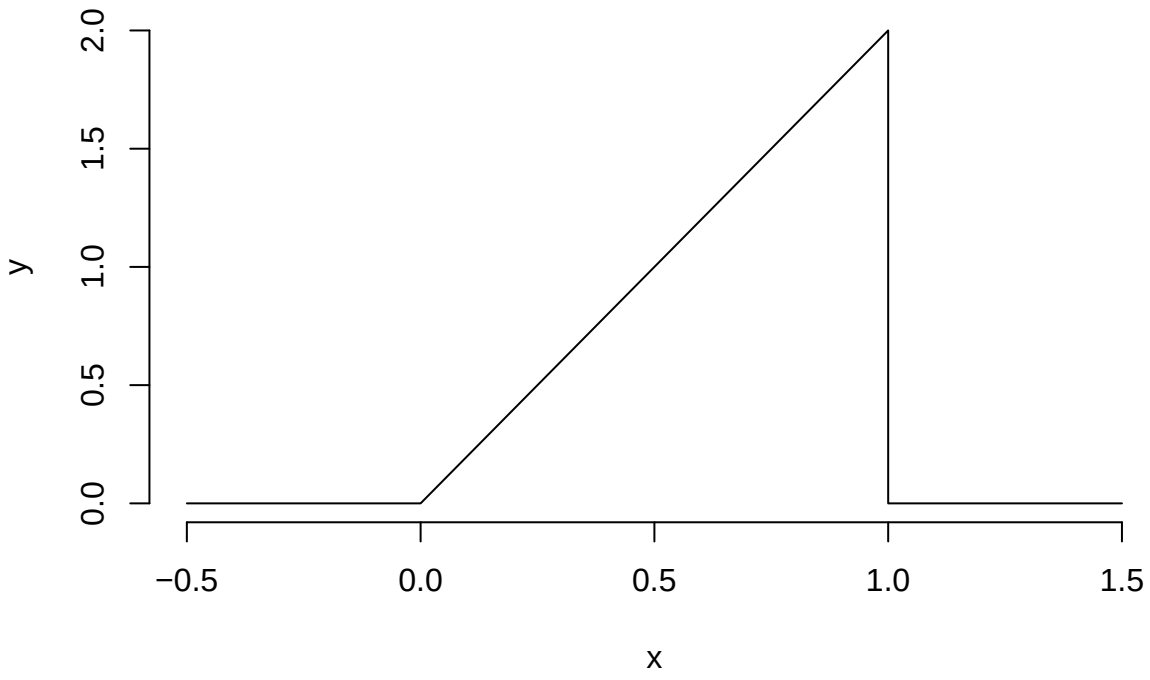
8/25/2020

examples of PDFs

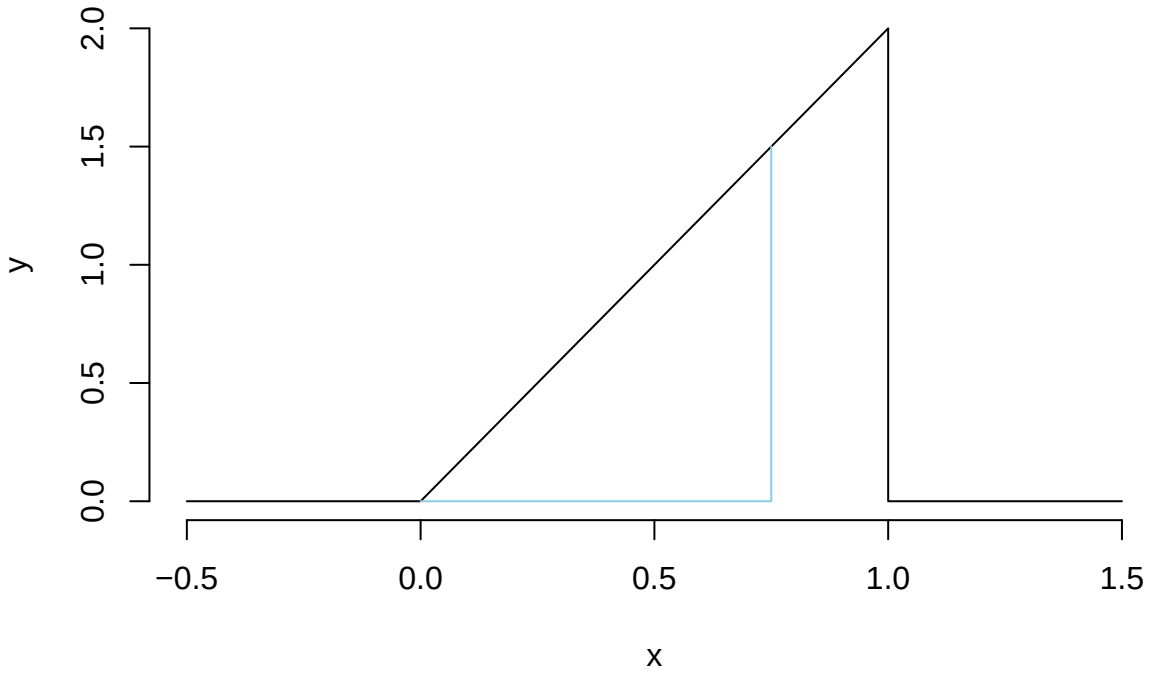
Example 1

Consider the function to denote the probability of help calls addressed by a helpline be

```
f(x) = {2x for 0<x<1; 0 otherwise}
```



To find probability that 75% or fewer of calls get addressed



Find area of given portion

```
(1/2)*(0.75)*(1.5)
= 0.5625
```

The same can be achieved by using the beta function for probability - pbeta()

```
pbeta(0.75,2,1)
```

The quantile for the above population distribution
Median from the distribution shows the datapoint below which 50% of the data is present and above it is the other 50%

```
0.5 = F(X) = P(X<=x) = 50% of the area
Integrating the function F(X)=2x, we get x^2
=> x = sqrt(2) = 0.707
=> The required datapoint
```

The same can be found out using the qbeta() function which gives the quantile of the beta distribution

```
qbeta(0.5,2,1)
```

The inference is that during 50% of the day ~70% of the calls are addressed.

Real World Example of Statistical Inference

Using the Son's Height attribute from the father.son data

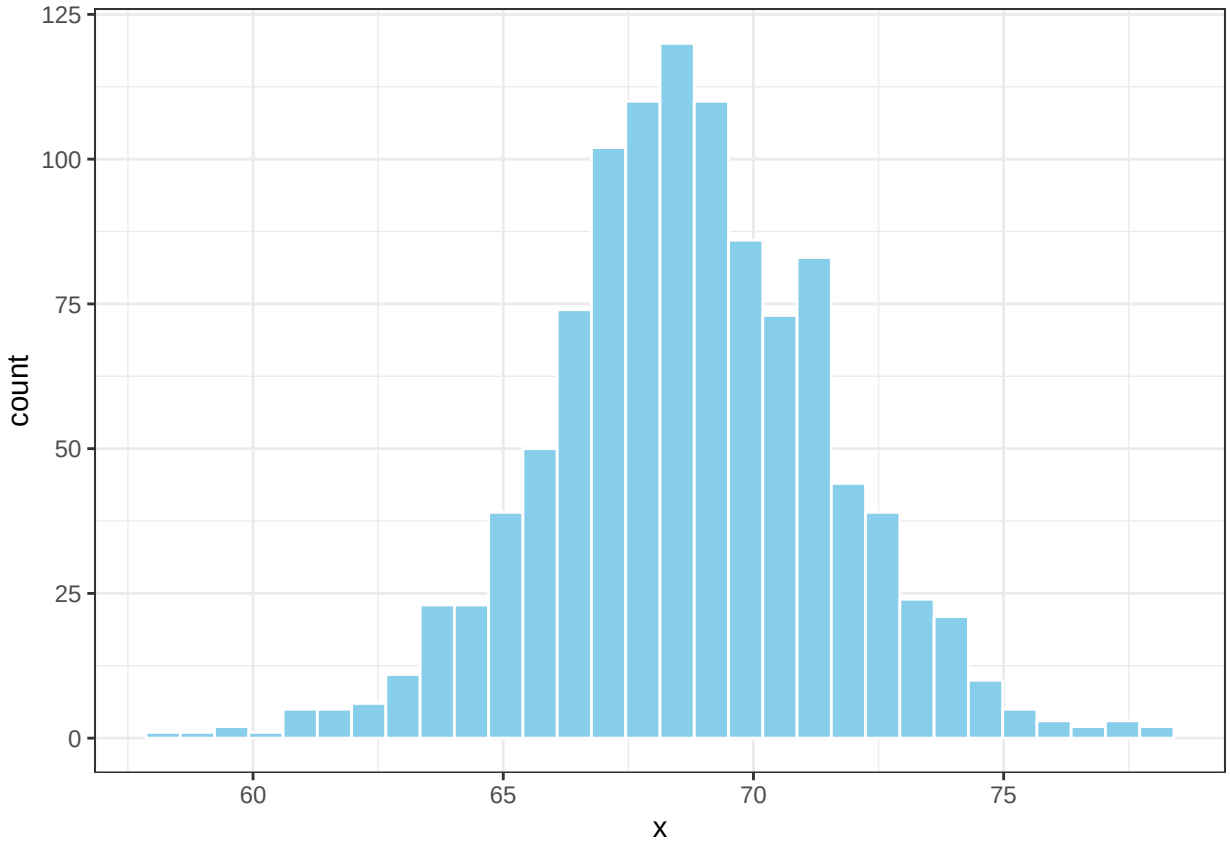
Loading the data

```
## Loading the data
suppressMessages(
  {
    if(!require('UsingR')){
      install.packages('UsingR')
      library(UsingR); data(father.son)
    }
    if(!require('dplyr')){
      install.packages("dplyr")
      library(dplyr)
    }
  }
)
x = father.son$sheight
n = length(x)
```

Plot of heights

```
ggplot() + geom_histogram(aes(x), col = "white", fill = "skyblue") + theme_bw()
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



The height is represented in feet instead of inches, divide the height by 12 to convert to feet unit

```
round(c(var(x),var(x)/n,sd(x),sd(x)/sqrt(n)),2)
```

```
## [1] 7.92 0.01 2.81 0.09
```

Explaining confidence intervals

```
(mean(x)+c(-1,1)*qnorm(0.975)*sd(x)/sqrt(length(x)))/12
```

```
## [1] 5.709670 5.737674
```

This tells us that if we were to iid draw the sons from this population the CI for drawing an average height to the sons would in the interval 5.71 to 5.74

Sample Proportions: In an event that each Xi is 0 or 1(binary outcome), with common success probability p, then variance(sigma^2) = p(1-p)
Then the interval takes the form
Wald confidence interval for p - For 95% intervals is:

```
  ^p (+/-) 1/sqrt(n)
```

which is a quick CI estimate for p

Confidence interval example

Your campaign advisor told you that in a random sample of 100 likely voters, 56 intend to vote for you. - Can you relax? Do you have this race in the bag? - without access to a computer or calculator, how precise is the estimate?

Using Wald confidence, 1/(sqrt(100)) gives 0.1 confidence interval of [0.46,0.66] - Not enoug for you to relax, we can't rule out possibilities below 0.5 with 95% confidence, better go do more campaigning.

The above calculation of CI can be done using the binom.test() function in R

```
binom.test(56,100)$conf.int
```

```
## [1] 0.4571875 0.6591640
## attr(,"conf.level")
## [1] 0.95
```

Yielding a similar result as before. Mathematically which is

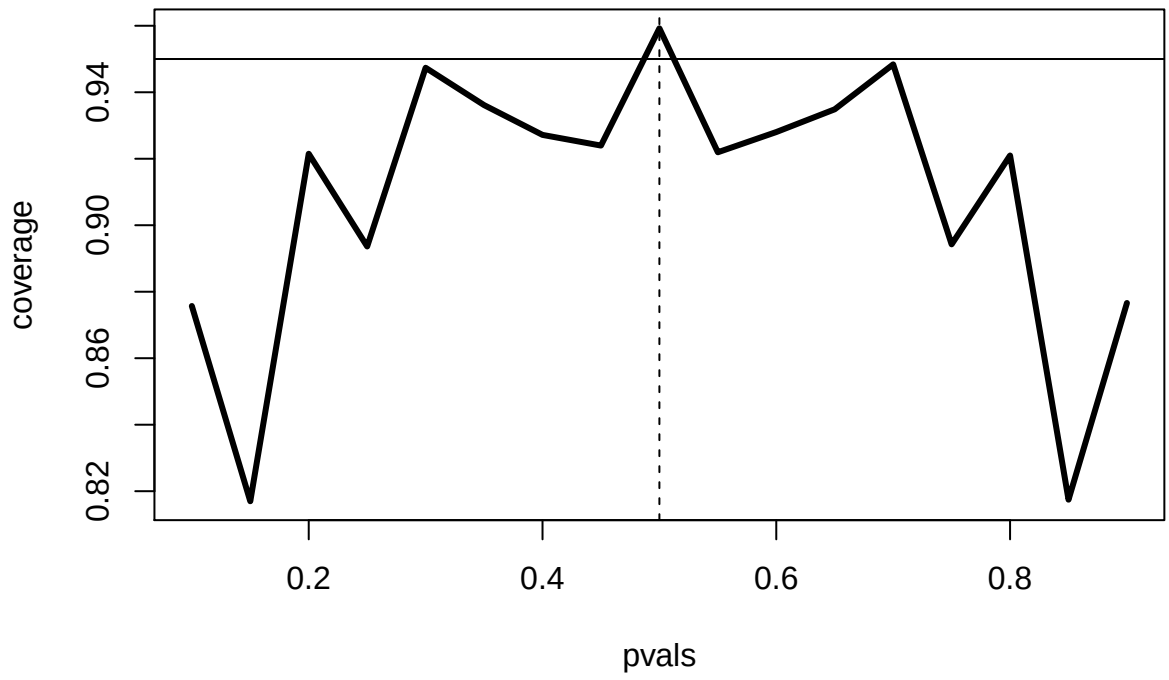
```
0.56 + c(-1,1)*qnorm(0.975)*sqrt(0.56*0.44/100)
```

```
## [1] 0.4627099 0.6572901
```

Biased coin flip estimation using Wald's Confidence

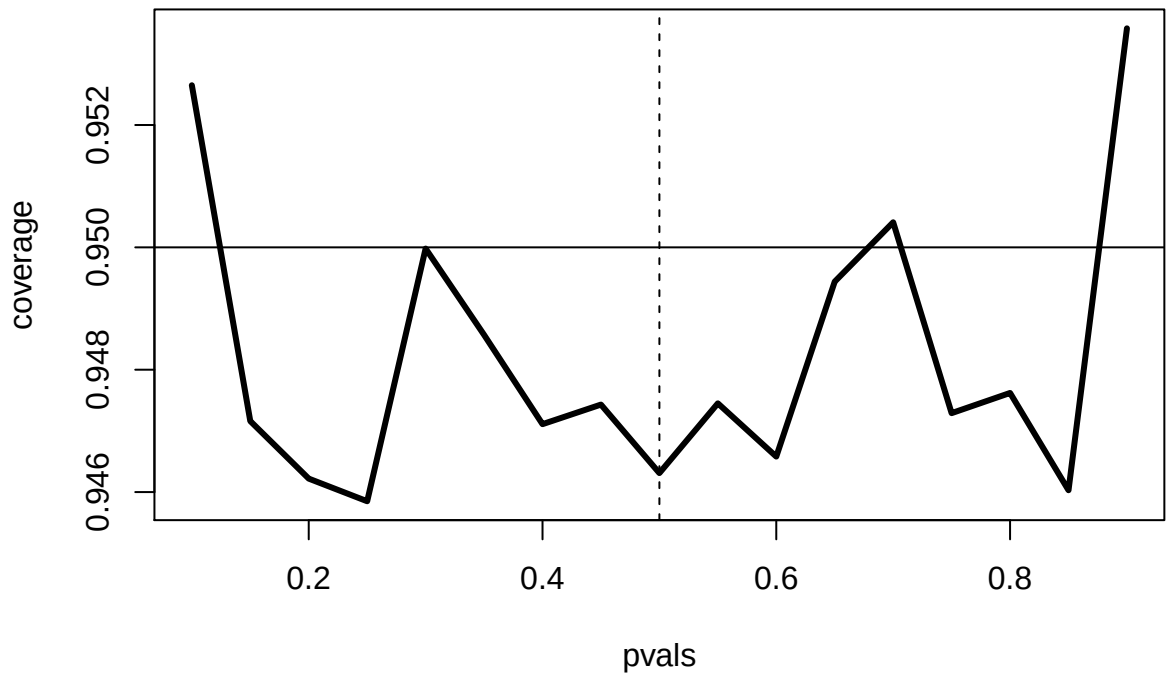
varying p value to find the p val where estimator within confidence interval of the parameter mu,

```
flips_per_sim = 20
pvals = seq(0.1,0.9,by = 0.05)
n_sim = 100000
coverage = sapply(pvals, function(p){
  phats = rbinom(n_sim, prob = p, size = flips_per_sim)/flips_per_sim
  ll = phats - qnorm(0.975)*sqrt(phats*(1-phats)/flips_per_sim)
  ul = phats + qnorm(0.975)*sqrt(phats*(1-phats)/flips_per_sim)
  mean(ll<p & ul>p)
})
plot(pvals, coverage, type = "l", lwd = 3)
abline(h = 0.95)
abline(v = 0.5, lty = 2)
```



this shows that when n, the number of flips, is small (20) the CLT doesn't hold for many values of p, so the Wald interval doesn't work very well.
When we increase n, the number of coin flips in each of our 1000 trials, from 20 to 100 to see if the plot improves. Again, results may vary, but all the probabilities are much closer to the 95% line, so the CLT works better with a bigger value of n

```
flips_per_sim = 1000
pvals = seq(0.1,0.9,by = 0.05)
n_sim = 100000
coverage = sapply(pvals, function(p){
  phats = rbinom(n_sim, prob = p, size = flips_per_sim)/flips_per_sim
  ll = phats - qnorm(0.975)*sqrt(phats*(1-phats)/flips_per_sim)
  ul = phats + qnorm(0.975)*sqrt(phats*(1-phats)/flips_per_sim)
  mean(ll<p & ul>p)
})
plot(pvals, coverage, type = "l", lwd = 3)
abline(h = 0.95)
abline(v = 0.5, lty = 2)
```



A quick fix to the problem of having a small n is to use the Agresti/Coull interval. This simply means we add 2 successes and 2 failures to the counts when calculating the proportion p'. It is to be noted that although this works, the technique might make the confidence interval too wide.
Why does this work? Adding 2 successes and 2 failures pulls p' closer to .5 which, as we saw, is the value which maximizes the confidence interval.

Understanding T confidence by performing a paired T-Test

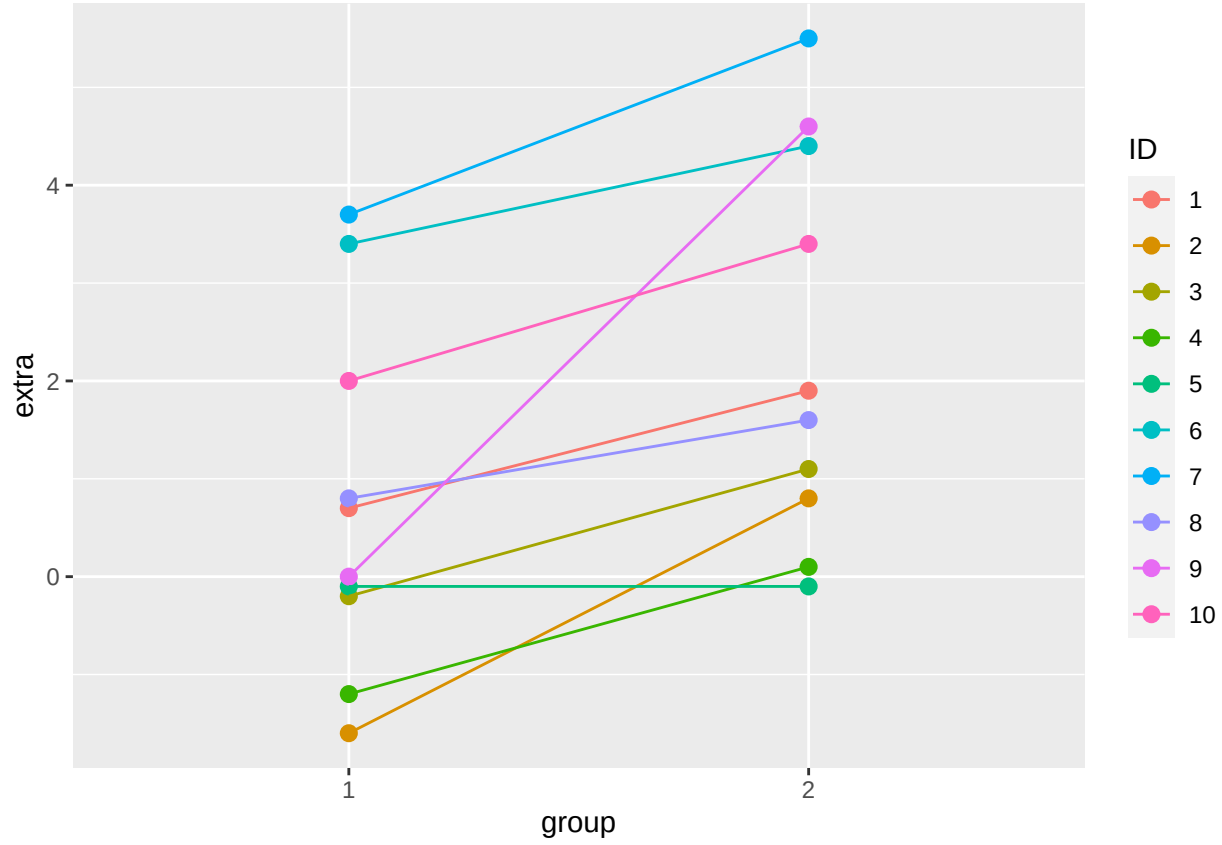
The sleep data, is used to analyse the change in sleeping periods of patients under the influence of two separate drugs hence explaining the groups field of factor datatype containing two levels 1 and 2.
Here the pairing is between the two groups that have the same patient with ID 1 represented by ID 11 in the group 2.
This allows us to study the variations of effects of the drugs on the same patients.

```
data("sleep")
head(sleep)
```

```
##   extra group ID
## 1   0.7     1  1
## 2  -1.6     1  2
## 3  -0.2     1  3
## 4  -1.2     1  4
## 5  -0.1     1  5
## 6   3.4     1  6
```

Plotting the difference in the sleep period

```
ggplot(
  sleep,
  aes(
    x = group,
    y = extra,
    group = ID
  )
) + geom_point(aes(col = ID), pch = 19, cex = 2.5) + geom_path(aes(col = ID))
```



Calculating the mean, variance and standard deviation between the results of the two groups.

```
g1 <- sleep$extra[1 : 10]; g2 <- sleep$extra[11 : 20]
difference <- g2 - g1
mn <- mean(difference); s <- sd(difference); n <- 10
```

Calculating the T confidence

```
mn + c(-1, 1) * qt(.975, n-1) * s / sqrt(n)
```

```
## [1] 0.7001142 2.4598858
```

```r
t.test(difference)
```

```
##
##  One Sample t-test
##
## data:  difference
## t = 4.0621, df = 9, p-value = 0.002833
## alternative hypothesis: true mean is not equal to 0
## 95 percent confidence interval:
##  0.7001142 2.4598858
## sample estimates:
## mean of x
##      1.58
```

```r
t.test(g2, g1, paired = TRUE)
```

```
##
##  Paired t-test
##
## data:  g2 and g1
## t = 4.0621, df = 9, p-value = 0.002833
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  0.7001142 2.4598858
## sample estimates:
## mean of the differences
##                    1.58
```

```r
t.test(extra ~ I(relevel(group, 2)), paired = TRUE, data = sleep)
```

```
##
##  Paired t-test
##
## data:  extra by I(relevel(group, 2))
## t = 4.0621, df = 9, p-value = 0.002833
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  0.7001142 2.4598858
## sample estimates:
## mean of the differences
##                    1.58
```

Which tells us that there is difference in the heights. ## Understanding T intervals Consider a test comparing the SBP(standard blood pressure) for 8 oral contraceptive users versus 21 controls.
$\mathbf{X}$oc = 132.86 mmHg with soc = 15.34 $\mathbf{X}$c = 127.44 mmHg with sc = 18.23 We are required to find out using this sample the T-confidence interval for the average difference in the two independent groups. If the interval >= 0, then we can confidently state that the drug causes increase in blood pressure population.

```r
sp = sqrt((7*15.34^2+20*18.23^2)/(8+21-2))
132.86 - 127.44 + c(-1,1)*qt(0.975,27)*sp*sqrt(1/8+1/21)
```

```
## [1] -9.521097 20.361097
```

Since the above interval contains 0, there is possibility that there is 0 difference between the population of the two groups.

Power explained

Supposed we were to calculate the probability of the sample mean to be 32, given that the population mean is 30 with standard deviation(sigma) 4 and number of samples drawn to be 16, we'd use the pnorm() function with mean = 30 and sd = sigma/sqrt(16)
Then,

```
mu0 = 30
mua = 32
sigma = 4
n = 16
alpha = 0.05
z = qnorm(1-alpha)
pnorm(mu0+z*sigma/sqrt(n), mean = mu0, sd = sigma/sqrt(n), lower.tail = F)
```

```
## [1] 0.05
```

This tells us that we fail to reject the null hypothesis because the t-statistic is equal to alpha.

Now to calculate power, we simply replace the mean m0 with ma,

```
pnorm(mu0+z*sigma/sqrt(n), mean = mua, sd = sigma/sqrt(n), lower.tail = F)
```

```
## [1] 0.63876
```

This gives us power, the percent probability with which we can assure that the null hypothesis is not true, generally you should have an 80% or greater chance of finding a statistically significant difference when there is one.

Illustrating the error correction in multiple comparison test cases

```
set.seed(1010093)
## Creating NULL vector for p-values, we simulate 1000 hypothesis tests
pValues = rep(NULL,1000)
for(i in 1:1000){
  ## Generating two independent normals x and y
  x = rnorm(20)
  y = rnorm(20)
  ## Fitting a linear model relating the two variables
  pValues[i] = summary(lm(y~x))$coeff[2,4]
}
sum(pValues<0.05)
```

```
## [1] 51
```

We get the expected ~50 significant p-values, while performing 1000 tests with confidence 0.05, which gives us 1000*0.05=50 i.e the chance of there being false positives.

Adjusting the p-values - Using bonferroni correction

```
## Controls FWER
sum(p.adjust(pValues, method = "bonferroni") < 0.05)
```

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

- Using Benjamini Hochberg correction

```
## Controls FDR
sum(p.adjust(pValues, method = "BH") < 0.05)
```

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

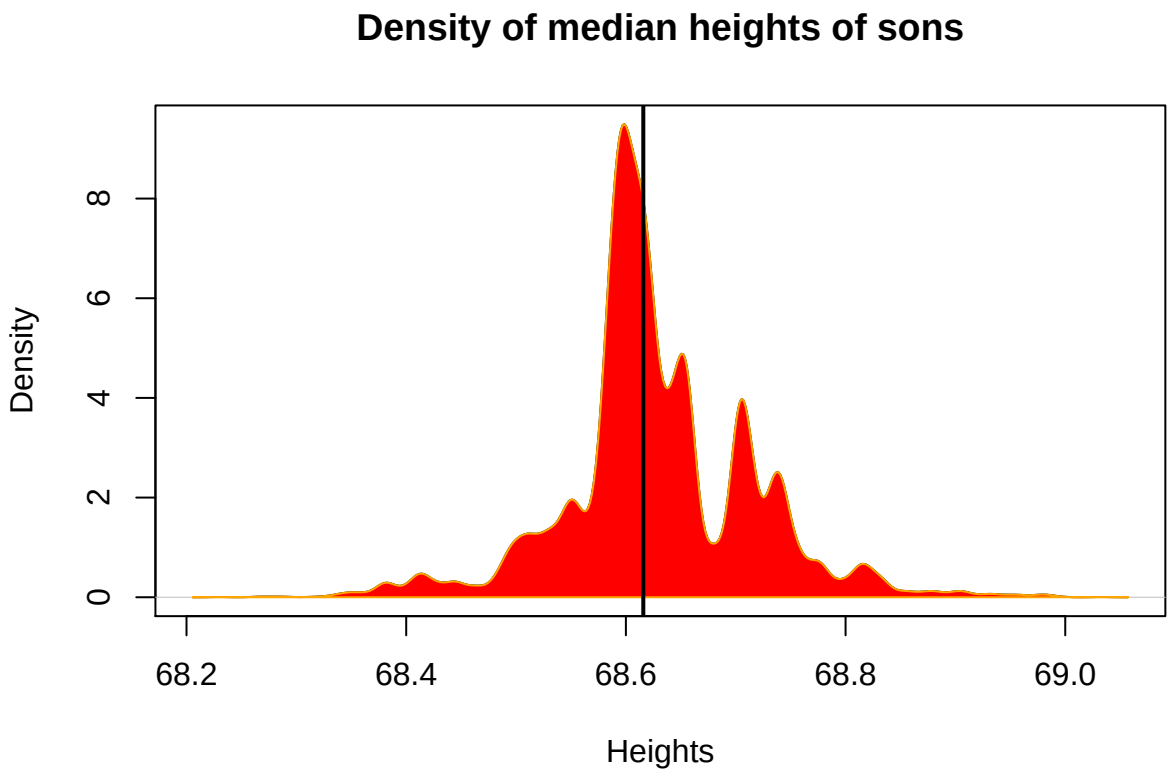
In case there is strong dependence between tests there may be problems when using the "bonferroni" or the "BH" methods, we should consider using "BY" under such circumstances.

Boostrapping Illustrated with father.son dataset

```
suppressMessages(
  {
    library(UsingR)
    data(father.son)
  }
)
x = father.son$sheight
n = length(x)
## We are performing 10000 bootstrap resamples
B = 10000
# We extract n times B samples with replacement, split the entire sample vector
# into multiple vectors of length n, then store each B sample vectors as rows
# of a matrix, here named 'resamples'
resamples = matrix(sample(x, n*B, replace = T),B,n)
## we then calculate the row medians and store in new vector resampledMedians
resampledMedians = apply(resamples, 1, median)
```

Visualising the vector of medians

```
d = density(resampledMedians)
plot(d, main = "Density of median heights of sons", xlab = "Heights")
polygon(d, col="red", border="orange")
abline(v = median(resampledMedians), lwd = 2)
```



The estimated standard deviation can be found

```
sd(resampledMedians)
```

```
## [1] 0.08556286
```

Confidence interval

```
quantile(resampledMedians,c(0.025,0.975))
```

```
##     2.5%    97.5%
## 68.43055 68.81531
```

Illustration of permutation test

Consider the InsectSprays dataset, which contains details of counts of insects killed by different bug-sprays.

```
data("InsectSprays")
head(InsectSprays)
```

```
##   count spray
## 1    10     A
## 2     7     A
```

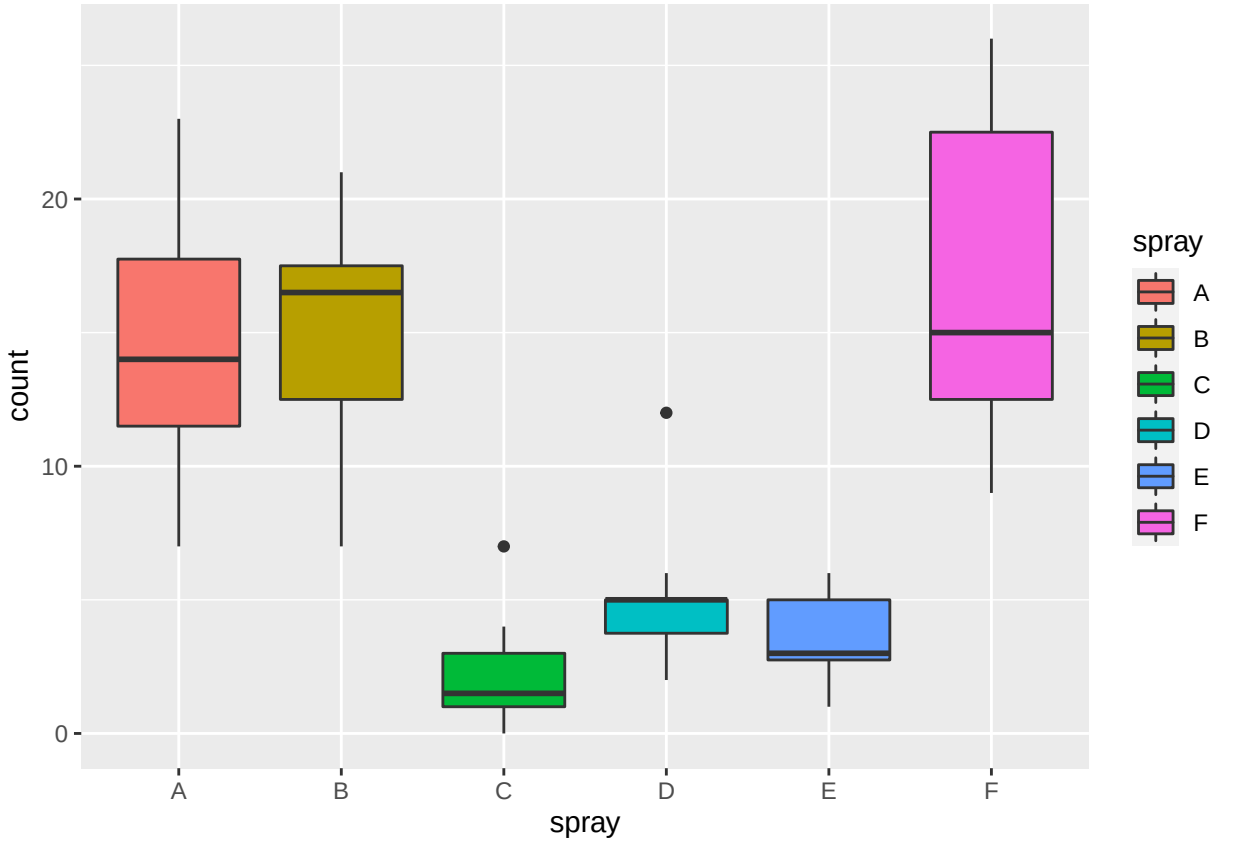
```
## 3    20     A
## 4    14     A
## 5    14     A
## 6    12     A
```

We subset the dataset to only get count of bugs killed by bug-spray 'B' and 'C' and define our variables 'x' and 'group'.

```
subset = subset(InsectSprays, spray %in% c("B","C"))
x = subset$count
group = as.character(subset$spray)
```

If we take mean of count of insects killed by each group of bug-sprays we know that each group of bug-spray has different means by the boxplot below

```
ggplot(
  InsectSprays,
  aes(
    x = spray,
    y = count
  )
) + geom_boxplot(aes(fill = spray))
```



We wish to show that the bug-spray has effect on count on bugs killed, we consider the null hypothesis that the group label is unrelated to the outcome. So if we were to permute the labels and find permutations

of labels that have higher mean difference than what we get compared to current group label then we can disapprove the null hypothesis.

```
## Function to compare the mean of each group given set of group labels
testStat = function(x,g){
  mean(x[g=='B'])-mean(x[g=='C'])
}
## Observing the difference in mean for the observed data
observedStat = testStat(x, group)
## finding various permutations on the group labels and finding the mean
## differences for each
permutations = sapply(1:10000, function(i) testStat(x,sample(group))) # This
# vector would cantain the mean difference in 10000 permutations groups B and C
# labels
print(paste('Observed mean difference:',as.character(observedStat)))
```

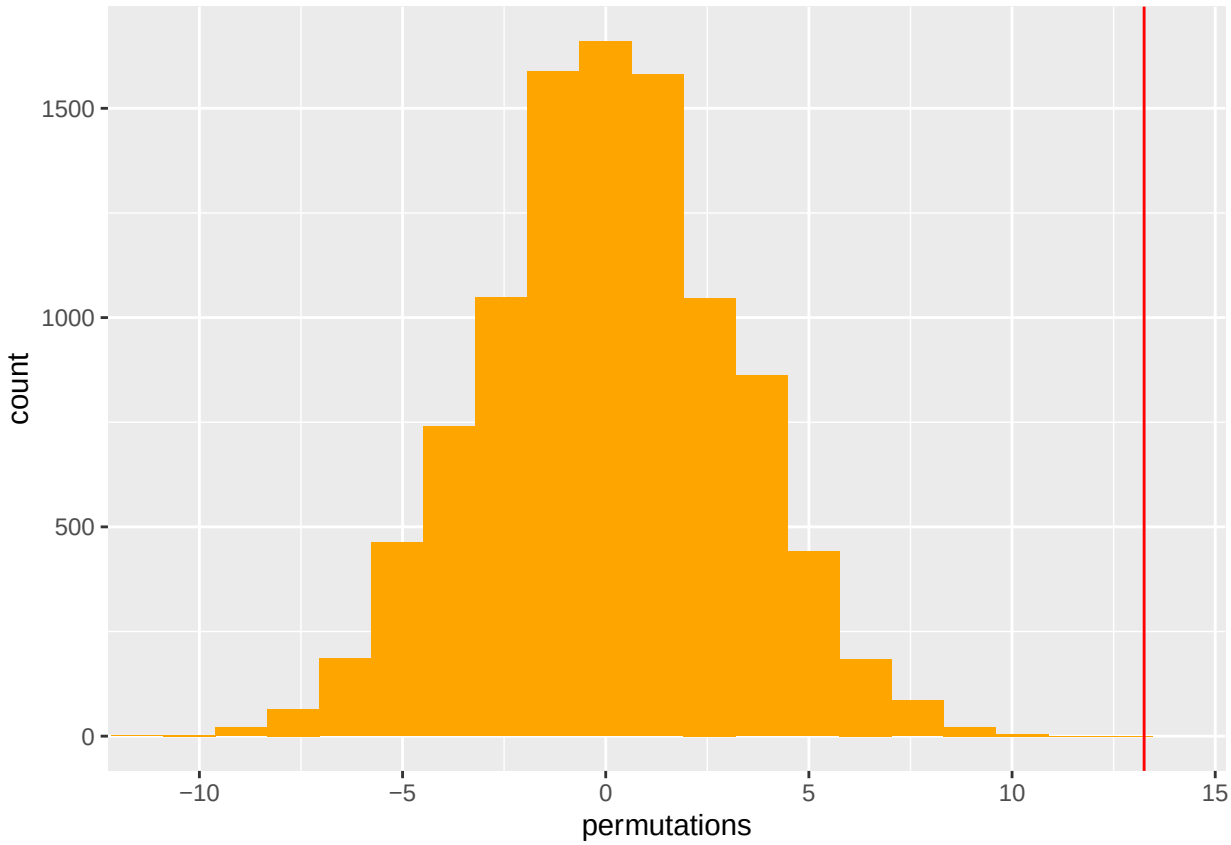
```
## [1] "Observed mean difference: 13.25"
```

```
print(paste0('Percentage of permutation that have higher mean difference: ',as.character(mean(permutati
```

```
## [1] "Percentage of permutation that have higher mean difference: 0%"
```

Hence we are able to reject the null that the group labels have no effect on the outcome.

```
ggplot(
  data.frame(permutations),
  aes(x = permutations),
) + coord_cartesian(xlim = c(-11,14)) +
  geom_histogram(bins = 20,fill="orange") +
  geom_vline(aes(xintercept=observedStat),col = "red")
```



We observe from the plot that the observed statistic is far ahead of the normal distribution of the permutations.