# CH5: Inference for a single mean/median

## 1 One-sample t-test and CI

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

```
exercise <- read.csv("C:/hess/STAT511_FA11/RData/CH5_Exercise.csv")
str(exercise)
```

```
## 'data.frame':    35 obs. of  1 variable:
##  $ y: int  23 19 36 12 41 43 19 28 14 44 ...
```

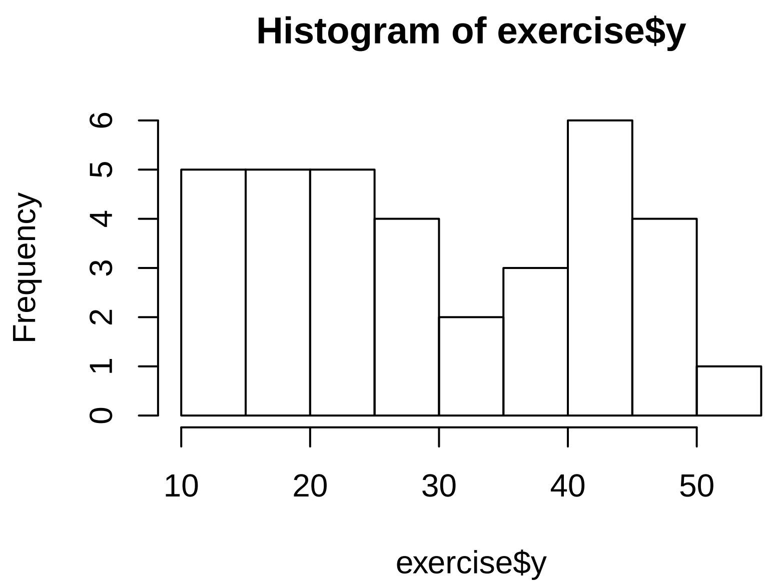
```
mean(exercise$y)
```

```
## [1] 30.51429
```

```
sd(exercise$y)
```

```
## [1] 12.35831
```

```
hist(exercise$y)
```




### 1.1 Confidence Interval

By default, a 95% CI is returned. Use the conf.level option to change from the default.

```
t.test(exercise$y)
```

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

```
## data:  exercise$y
## t = 14.608, df = 34, p-value = 3.244e-16
## alternative hypothesis: true mean is not equal to 0
## 95 percent confidence interval:
##  26.26906 34.75951
## sample estimates:
## mean of x
##  30.51429
```

## 1.2 Two-sided Test

Here we do a two-sided test of H0: $\mu = 25$ vs HA: $\mu \neq 25$. Remember that the hypotheses should be motivated by the research question and can (should!) be specified before looking at the data.

```
t.test(exercise$y, mu = 25)
```

```
##
##  One Sample t-test
##
## data:  exercise$y
## t = 2.6398, df = 34, p-value = 0.01243
## alternative hypothesis: true mean is not equal to 25
## 95 percent confidence interval:
##  26.26906 34.75951
## sample estimates:
## mean of x
##  30.51429
```

## 1.3 One-sided Test

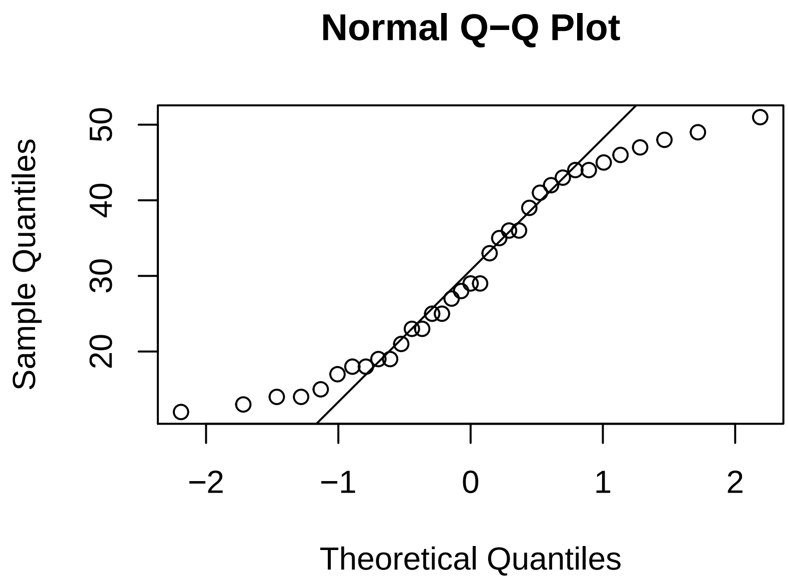
Now we do a one-sided test of H0: $\mu \leq 25$ vs HA: $\mu > 25$.

```
t.test(exercise$y, mu = 25, alternative =  "greater")
```

```
##
##  One Sample t-test
##
## data:  exercise$y
## t = 2.6398, df = 34, p-value = 0.006217
## alternative hypothesis: true mean is greater than 25
## 95 percent confidence interval:
##  26.98205      Inf
## sample estimates:
## mean of x
##  30.51429
```

## 1.4 Evaluating Normality

```
#QQplot
qqnorm(exercise$y)
qqline(exercise$y)
```




```r
#Tests of normality
shapiro.test(exercise$y)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  exercise$y
## W = 0.92897, p-value = 0.02608
```

```r
ks.test(exercise$y, "pnorm", mean(exercise$y), sd(exercise$y) )
```

```
## Warning in ks.test(exercise$y, "pnorm", mean(exercise$y), sd(exercise$y)):
## ties should not be present for the Kolmogorov-Smirnov test
```

```
##
##  One-sample Kolmogorov-Smirnov test
##
## data:  exercise$y
## D = 0.1162, p-value = 0.732
## alternative hypothesis: two-sided
```

### 1.5 tidyverse

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.

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

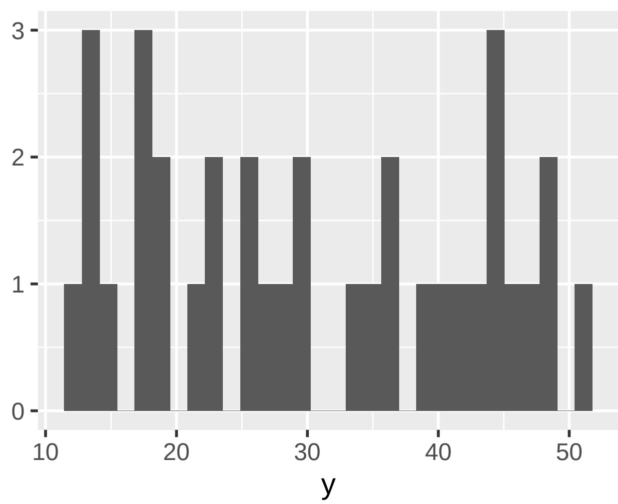
```r
SumStats
```

```
##    n     mean       sd       SE
## 1 35 30.51429 12.35831 2.088935
```

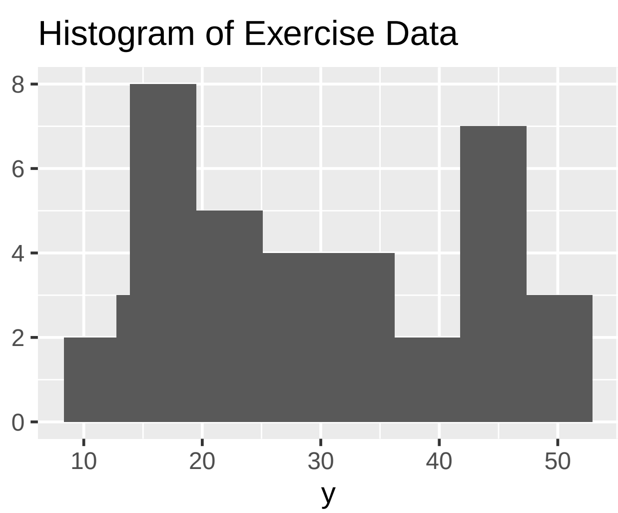
```r
tidy(t.test(exercise$y))
```

```
## # A tibble: 1 x 8
##   estimate statistic  p.value parameter conf.low conf.high method
##      <dbl>     <dbl>    <dbl>     <dbl>    <dbl>     <dbl> <chr>
## 1     30.5      14.6 3.24e-16        34     26.3      34.8 One S~
## # ... with 1 more variable: alternative <chr>
```

```r
qplot(y, data = exercise)
```




```r
qplot(y, data = exercise) +
  stat_bin(bins = 8) +
  ggtitle("Histogram of Exercise Data")
```



# 2 Confidence Interval Simulation

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 some handy functions from dplyr and broom.

## 2.1 Simulate data from Standard Normal

Hence $\mu$ = 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
```

```
#Apply t.test function 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
##      <dbl>     <dbl>   <dbl>     <dbl>    <dbl>     <dbl> <chr>
## 1   -0.406     -1.84  0.0780        24   -0.860    0.0491 One S~
```

```
## # ... with 1 more variable: alternative <chr>
```

```r
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
```

```r
temp$statistic
```

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

```r
temp$p.value
```

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

```r
rm(temp)
```

## 2.2 Now t.test for each SampleID

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

```r
OutData <- SimData %>%
           group_by(SampleID) %>%
           do(ttest = with(t.test(Y), data = .)) %>%
              tidy(ttest) %>%
              select(SampleID, statistic, p.value, conf.low, conf.high)

head(OutData)
```

```
## # A tibble: 6 x 5
## # Groups:   SampleID [6]
##   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
```

## 2.3 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 occurences.

As expected, we find that about 5% of tests (48/1000) return "false positives", corresonding 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()
```

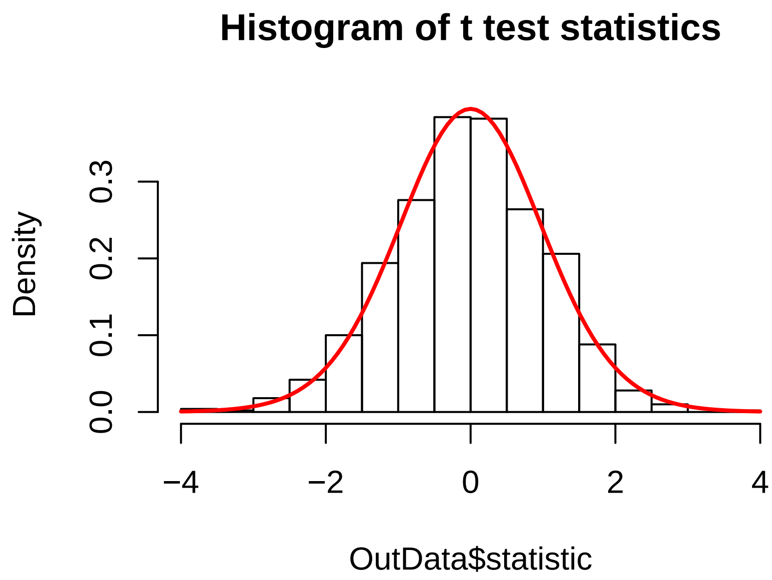
```
## # A tibble: 6 x 7
## # Groups:   SampleID [6]
##   SampleID statistic p.value conf.low conf.high CIFlag PvalFlag
##      <int>     <dbl>   <dbl>    <dbl>     <dbl>  <dbl>    <dbl>
## 1       17     -2.67  0.0133   -0.826   -0.106       1        1
## 2       30     -2.44  0.0225   -0.842   -0.0700      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)
```

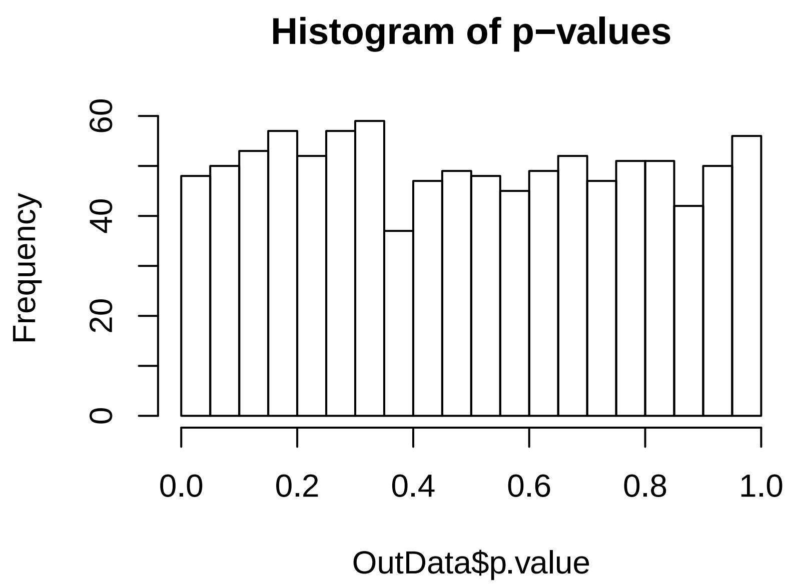
```
##
##       0   1
##   0 952   0
##   1   0  48
```

## 2.4 TS and p-value distributions

```
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)
```




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



# 3 Power for a One-sample t-test

## 3.1 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)
```

## 3.2 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
```

## 3.3 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)
```

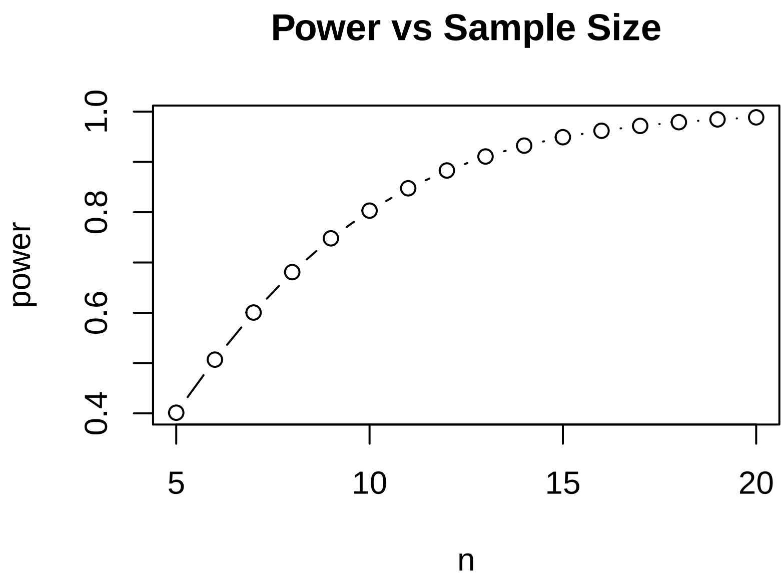
```
## [1] 0.802582
```

## 3.4 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
```

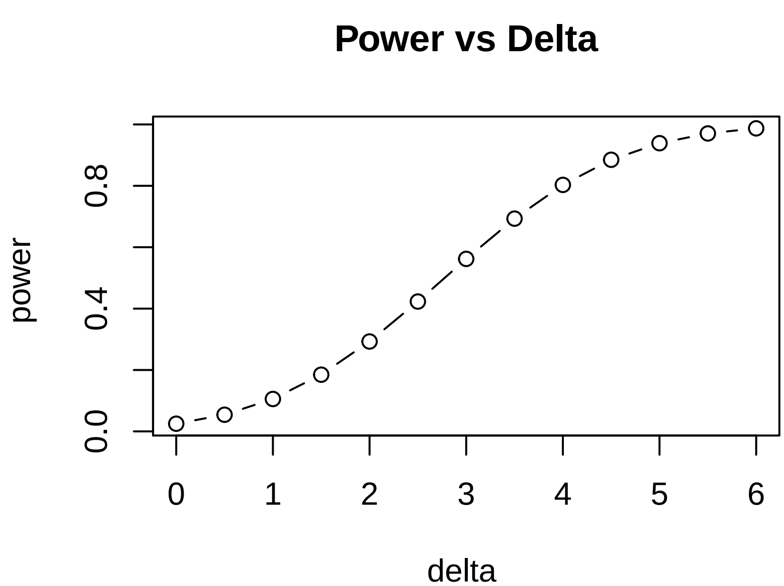
```
##
##      One-sample t test power calculation
##
##               n = 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20
##           delta = 4
##              sd = 4
##       sig.level = 0.05
##           power = 0.4013203, 0.5068167, 0.6004875, 0.6808301, 0.7480155, 0.8030962, 0.8475297, 0.8828
##     alternative = two.sided
```

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



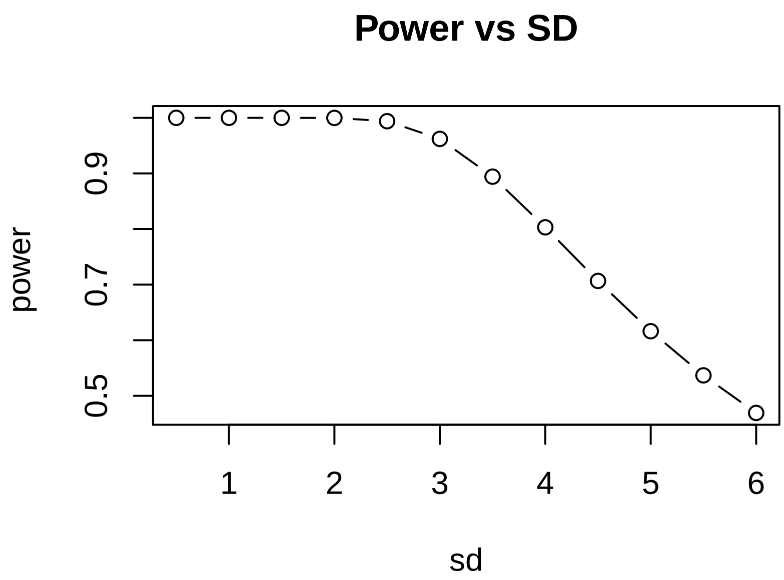

### 3.5 Graph of Power vs Delta (Difference between means)

```
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")
```



### 3.6 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)
```

# 4 Bootstrap CI for Single Mean

```
mercury <- c(1.23, 1.33, 0.04, 0.44, 1.2, 0.27, 0.48, 0.19, 0.83, 0.81, 0.71, 0.5,
             0.49, 1.16, 0.05, 0.15, 0.19, 0.77, 1.08, 0.98, 0.63, 0.56, 0.41, 0.73,
             0.34, 0.59, 0.34, 0.84, 0.5, 0.34, 0.28, 0.34, 0.87, 0.56, 0.17, 0.18,
             0.19, 0.04, 0.49, 1.1, 0.16, 0.1, 0.48, 0.21, 0.86, 0.52, 0.65, 0.27,
             0.94, 0.4, 0.43, 0.25, 0.27)
sort(mercury)
```

```
##  [1] 0.04 0.04 0.05 0.10 0.15 0.16 0.17 0.18 0.19 0.19 0.19 0.21 0.25 0.27
## [15] 0.27 0.27 0.28 0.34 0.34 0.34 0.34 0.40 0.41 0.43 0.44 0.48 0.48 0.49
## [29] 0.49 0.50 0.50 0.52 0.56 0.56 0.59 0.63 0.65 0.71 0.73 0.77 0.81 0.83
## [43] 0.84 0.86 0.87 0.94 0.98 1.08 1.10 1.16 1.20 1.23 1.33
```

```
length(mercury)
```

```
## [1] 53
```

```
mean(mercury); sd(mercury)
```

```
## [1] 0.5271698
```

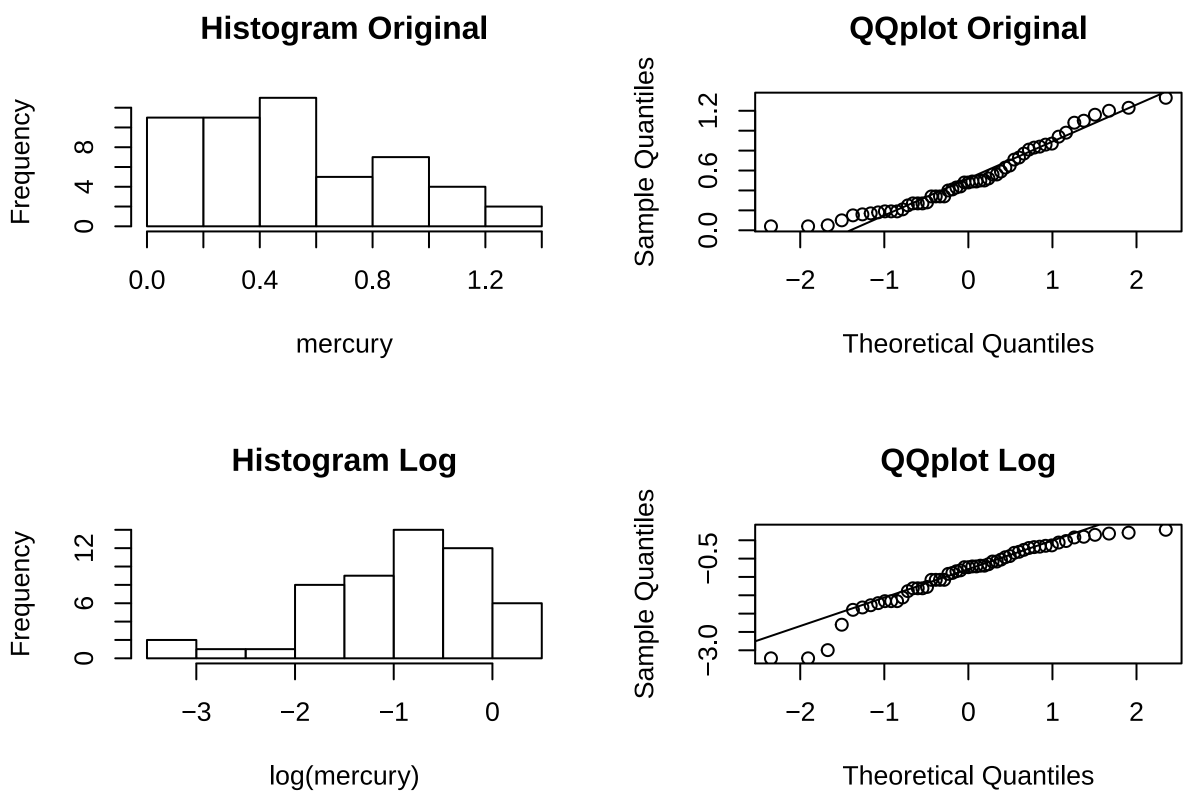
```
## [1] 0.3410356
```

```
t.test(mercury)
```

```
##
##  One Sample t-test
##
## data:  mercury
## t = 11.254, df = 52, p-value = 1.506e-15
## alternative hypothesis: true mean is not equal to 0
## 95 percent confidence interval:
##  0.4331688 0.6211709
## sample estimates:
## mean of x
## 0.5271698
```

## 4.1 Histograms and QQplots - Original and Log Scales

```
par(mfrow = c(2, 2))
hist(mercury, main = "Histogram Original")
qqnorm(mercury, main = "QQplot Original"); qqline(mercury)
#log=natural log
hist(log(mercury), main = "Histogram Log")
qqnorm(log(mercury), main = "QQplot Log"); qqline(log(mercury))
```




## 4.2 "By Hand" For Illustration

Use lapply and sample to take 10000 bootstrap samples WITH replacement

```
n <- length(mercury)
set.seed(5825)
resamples <- lapply(1:10000, function(i)
    sample(mercury, size = n, replace = T))
dim(resamples); length(resamples)
```

```
## NULL
```

```
## [1] 10000
```

```
sort(resamples[[1]])
```

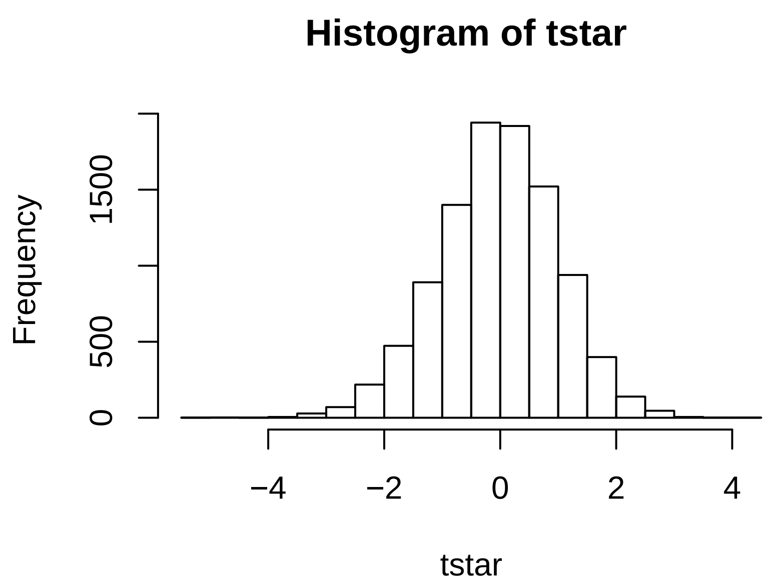
```
##  [1] 0.04 0.04 0.04 0.04 0.05 0.10 0.17 0.18 0.18 0.19 0.19 0.21 0.21 0.21
## [15] 0.25 0.25 0.27 0.27 0.28 0.28 0.28 0.34 0.34 0.34 0.34 0.40 0.40 0.41
## [29] 0.41 0.41 0.41 0.48 0.49 0.49 0.49 0.49 0.50 0.50 0.50 0.50 0.56 0.63
## [43] 0.83 0.84 0.84 0.84 0.87 0.87 0.98 0.98 1.16 1.20 1.33
```

```
resamples <- simplify2array(resamples)
dim(resamples)
```

```
## [1]    53 10000
```

```
colmeans <- apply(resamples, 2, mean)
colsd <- apply(resamples, 2, sd)
```

```r
tstar <- (colmeans - mean(mercury))/(colsd/sqrt(n))
hist(tstar, main = "Histogram of tstar ")
```




```r
#Bootstrap CI
t025 <- quantile(tstar, prob = 0.975)
t975 <- quantile(tstar, prob = 0.025)
t025; t975
```

```
##    97.5%
## 1.878339

##      2.5%
## -2.150141
```

```r
LB <- mean(mercury) - t025*sd(mercury)/sqrt(n)
UB <- mean(mercury) - t975*sd(mercury)/sqrt(n)
CI <- c(LB,UB); names(CI)<-c(); CI
```

```
## [1] 0.4391793 0.6278928
```

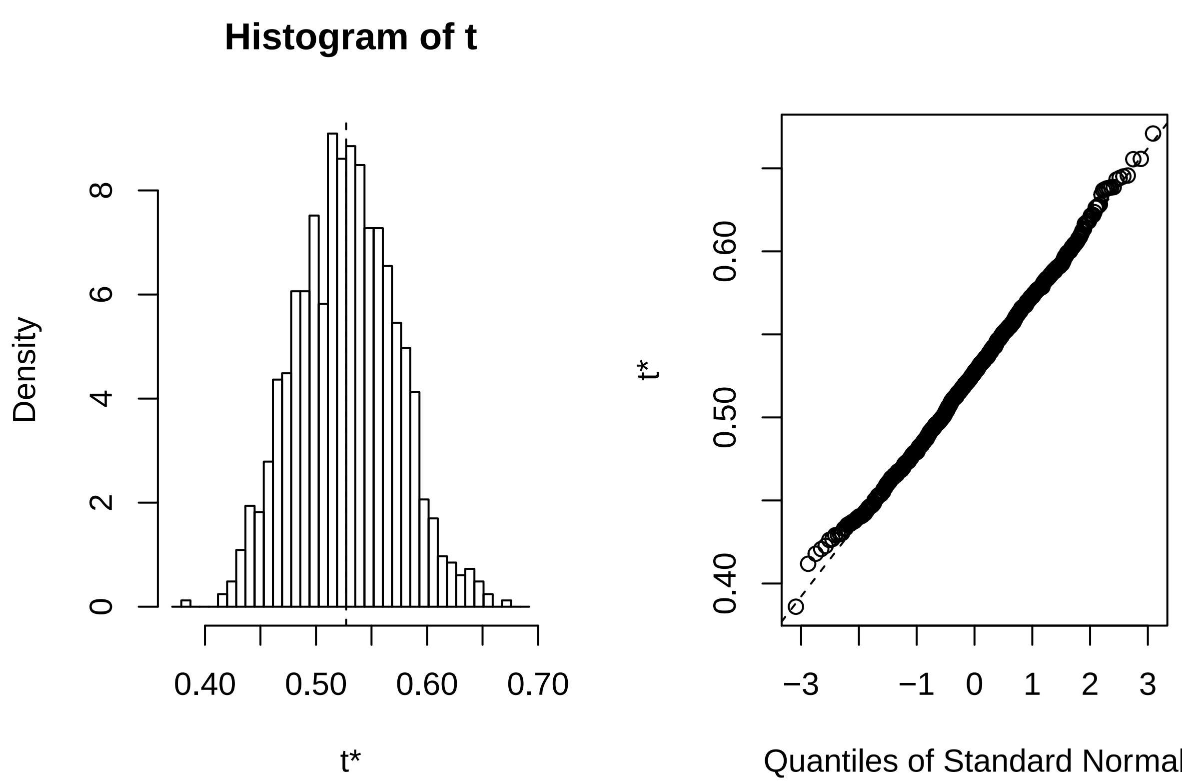
## 4.3 Boot Example 1

We now use the boot package to construct the bootstrap CI.
The boot package will need to be INSTALLED before it can be LOADED! This example does not return type=student because no variance calculated.

```r
library(boot)
set.seed(5841)
results1 <- boot(mercury, function(d,i) mean(d[i]), R = 1000)
results1
```

```
##
## ORDINARY NONPARAMETRIC BOOTSTRAP
##
##
## Call:
## boot(data = mercury, statistic = function(d, i) mean(d[i]), R = 1000)
##
##
## Bootstrap Statistics :
##      original        bias    std. error
## t1* 0.5271698 -6.792453e-06  0.04497613
```

```r
plot(results1)
```




```r
boot.ci(results1, type = "all")
```

```
## Warning in boot.ci(results1, type = "all"): bootstrap variances needed for
```

```
## studentized intervals
```

```
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 1000 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = results1, type = "all")
##
## Intervals :
## Level      Normal              Basic
## 95%   ( 0.4390,  0.6153 )   ( 0.4364,  0.6138 )
##
## Level     Percentile            BCa
## 95%   ( 0.4406,  0.6179 )   ( 0.4420,  0.6206 )
## Calculations and Intervals on Original Scale
```

### 4.4 Boot Example 2

Note that we start by defining the function.

```
#Define the function
mean.fun <- function (d, i)
{ m <- mean(d[i])
  n <- length(i)
  v <- (n-1)*var(d[i])/n^2
  c(m, v)
}
set.seed(7231)
results2 <- boot(data = mercury, mean.fun, R = 1000)
boot.ci(results2, type = "all")
```

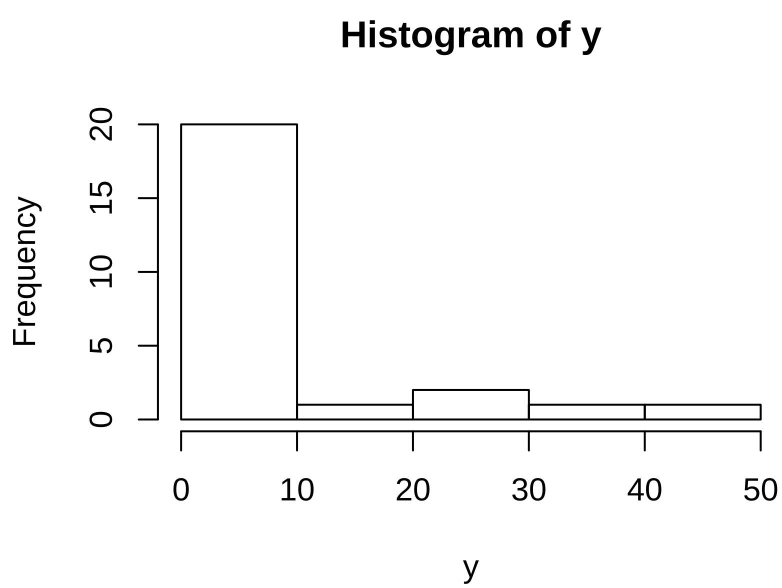
```
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 1000 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = results2, type = "all")
##
## Intervals :
## Level      Normal              Basic             Studentized
## 95%   ( 0.4362,  0.6192 )   ( 0.4339,  0.6192 )   ( 0.4378,  0.6301 )
##
## Level     Percentile            BCa
## 95%   ( 0.4351,  0.6204 )   ( 0.4418,  0.6240 )
## Calculations and Intervals on Original Scale
```

# 5 Sign Test

Sign Test is used to make inference for a single population median. This is a non-parametric test, so no assumption of normality required.

We will use the BSDA package for the Sign Test and CI for median using BSDA. The BSDA package will need to be INSTALLED before it can be LOADED!

```
library(BSDA)
y <- c(14.2, 5.3, 2.9, 4.2, 1.2, 4.3, 1.1, 2.6, 6.7, 7.8,
       25.9, 43.8, 2.7, 5.6, 7.8, 3.9, 4.7, 6.5, 29.5, 2.1,
       34.8, 3.6, 5.8, 4.5, 6.7)
hist(y)
```




```
summary(y)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   1.100   3.600   5.300   9.528   7.800  43.800
```

```
sort(y)
```

```
##  [1]  1.1  1.2  2.1  2.6  2.7  2.9  3.6  3.9  4.2  4.3  4.5  4.7  5.3  5.6
## [15]  5.8  6.5  6.7  6.7  7.8  7.8 14.2 25.9 29.5 34.8 43.8
```

```
SIGN.test(y, md = 5)
```

```
##
##  One-sample Sign-Test
##
## data:  y
## s = 13, p-value = 1
## alternative hypothesis: true median is not equal to 5
## 95 percent confidence interval:
##  3.931247 6.700000
## sample estimates:
```

```
## median of x
##        5.3
##
## Achieved and Interpolated Confidence Intervals:
##
##                   Conf.Level L.E.pt U.E.pt
## Lower Achieved CI     0.8922 4.2000    6.7
## Interpolated CI       0.9500 3.9312    6.7
## Upper Achieved CI     0.9567 3.9000    6.7
```

```r
#One-sided Test
SIGN.test(y, md = 5, alternative = "greater")
```

```
##
##  One-sample Sign-Test
##
## data:  y
## s = 13, p-value = 0.5
## alternative hypothesis: true median is greater than 5
## 95 percent confidence interval:
##  4.163925      Inf
## sample estimates:
## median of x
##        5.3
##
## Achieved and Interpolated Confidence Intervals:
##
##                   Conf.Level L.E.pt U.E.pt
## Lower Achieved CI     0.9461 4.2000    Inf
## Interpolated CI       0.9500 4.1639    Inf
## Upper Achieved CI     0.9784 3.9000    Inf
```