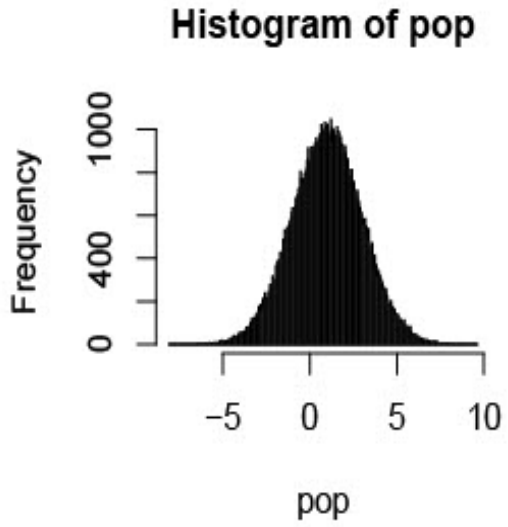
4 Sampling Distributions of the Sample Mean and Median

4.1 Sampling Distribution of the Mean

4.1.1 Normal Population

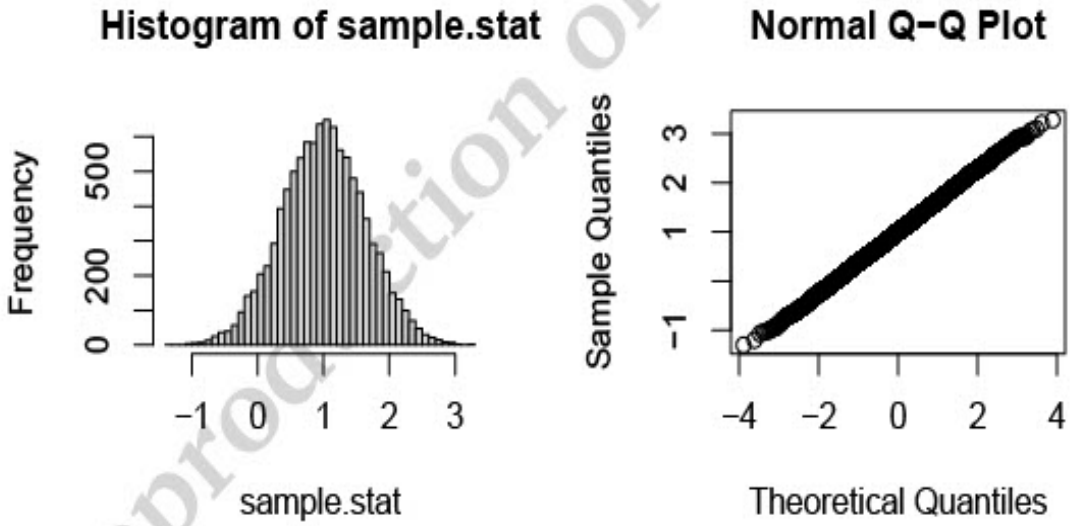
Instead of taking samples from a normal population, using rnorm(), we are going to take ONE huge sample from a normal population, using rnorm(), and then just treat it as our population. The main reason for this is mostly to set the stage for something called "bootstrapping," which we will study later.

```
N <- 100000  # Let N be the population size.
pop <- rnorm(N, 1, 2)  # Take a random sample and treat it as pop.
pop.mean <- mean(pop)  # This is mu, the population mean.
pop.sd <- sd(pop)  # This is sigma, the pop standard deviation.
pop.median <- median(pop)  # Get the population median, for later.
c(pop.mean, pop.sd, pop.median)  # Print them for comparison, below.

[1] 0.9958 2.0070 1.0022

hist(pop, breaks = 400)  # This shows that the population is pretty normal.

# Experiment underlying the sampling distribution.
n.trial <- 10000  # Take 10000 samples of
sample.size <- 10  # size 10 (i.e., small) from the population.
sample.stat <- numeric(n.trial)  # Create space to store the 10000 sample means.

for (i in 1:n.trial) {
  samp <- sample(pop, sample.size, replace = T)  # Take a sample (with replacement).
  sample.stat[i] <- mean(samp)  # Compute each sample's mean.
}
mean(sample.stat)  # Compare mean of sample means

[1] 1.002

pop.mean  # with the population mean.

[1] 0.9958

sd(sample.stat)  # Compare the standard deviation of sample means

[1] 0.6333

pop.sd  # with the pop standard deviation.

[1] 2.007

pop.sd / sqrt(sample.size)  # But compare with (pop std dev)/sqrt(n)

[1] 0.6347
```



According to the Central Limit Theorem (CLT), the sampling distribution of the sample mean should be normal. To confirm:

```
hist(sample.stat,breaks = 40)
qqnorm(sample.stat)  # Pretty normal.
```



As the sample size increases, the mean of the sample means gets pretty close to the population mean, and the standard deviation of the sample means gets pretty close to the $\frac{sd(pop)}{\sqrt{n}}$. So, the CLT is confirmed.

4.1.2 Non-normal Population

```
N <- 100000
pop <- rgamma(N, 1, 1)
```

```r
pop.mean <- mean(pop)
pop.sd <- sd(pop)
pop.median <- median(pop)
c(pop.mean, pop.sd, pop.median)
```

```
[1] 0.9968 0.9952 0.6955
```

```r
hist(pop, breaks = 400)  # The distribution of sample means looks non-normal.

n.trial <- 10000  # Take 10000 samples of
sample.size <- 10  # size 10 (i.e., small) from the population.
sample.stat <- numeric(n.trial)  # Space for storing the 10000 sample means.

for (i in 1:n.trial) {
  samp <- sample(pop, sample.size, replace=T)  # Take a sample (with replacement).
  sample.stat[i] <- mean(samp)  # and compute each sample's mean.
}

mean(sample.stat)  # Compare mean of sample means with population mean.
```

```
[1] 0.9985
```

```r
pop.mean
```

```
[1] 0.9968
```

```r
sd(sample.stat)  # Compare the sd of sample means with population sd.
```

```
[1] 0.3143
```

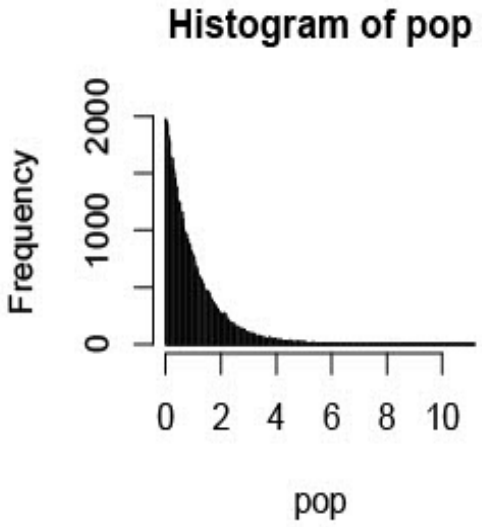
```r
pop.sd
```

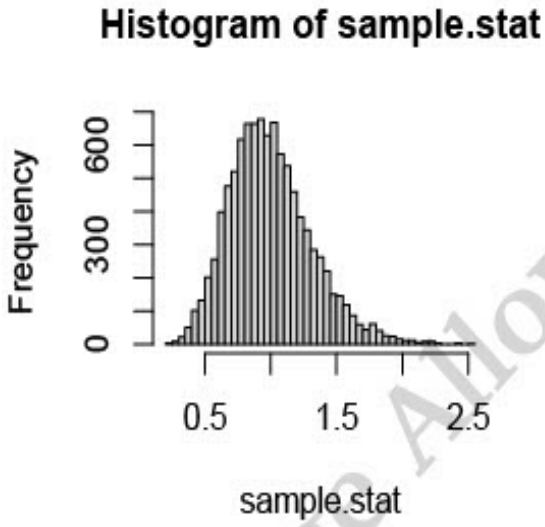
```
[1] 0.9952
```

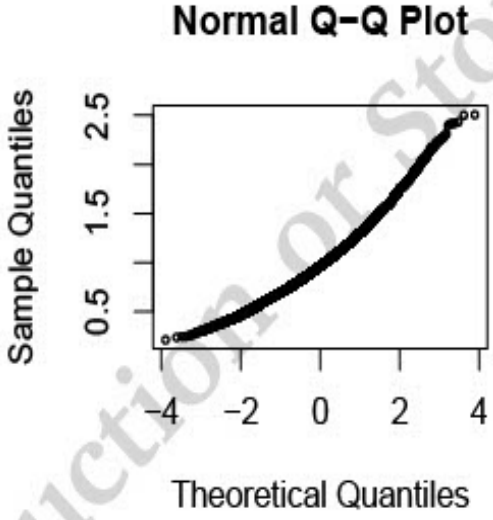
```r
pop.sd / sqrt(sample.size) # Compare with (pop sd)/root(n).
```

```
[1] 0.3147
```

```r
hist(sample.stat, breaks = 40)
qqnorm(sample.stat, cex = 0.5)
```







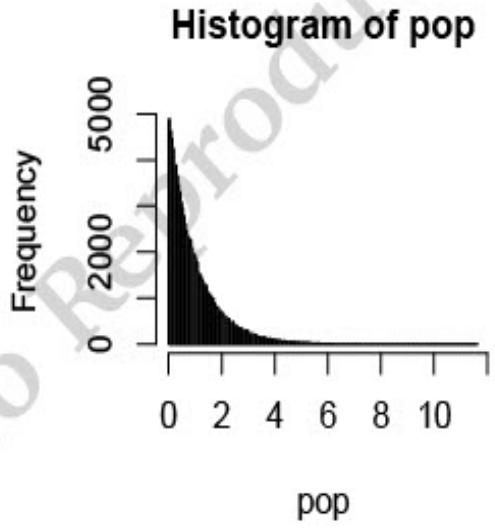
When the population is NOT normal, for small samples (10) the sampling distribution of the sample mean resists looking normal; but with larger samples (100), it is normal even though the population is not normal.

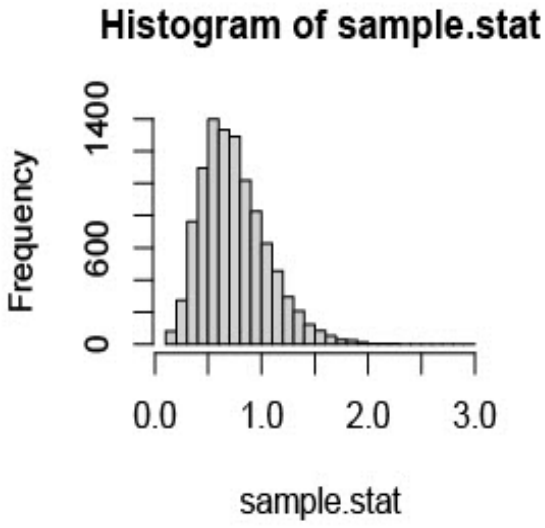
4.2 Sampling Distribution Population Median of sample median

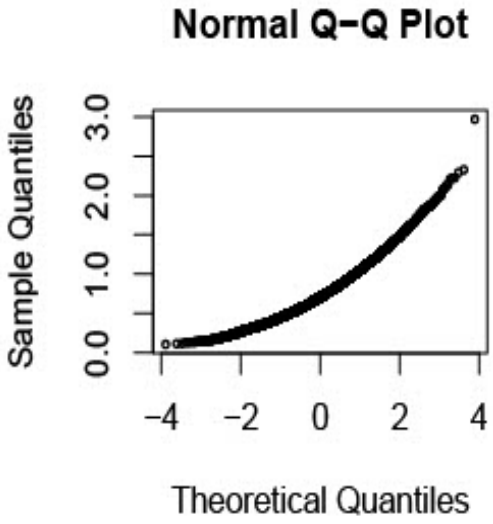
4.2.1 Non-normal Population

```
N <- 100000
pop <- rgamma(N,1,1)
pop.mean <- mean(pop)
pop.sd <- sd(pop)
pop.median <- median(pop)
c(pop.mean, pop.sd, pop.median)
```

```
[1] 1.0014 1.0009 0.6937

hist(pop, breaks = 400)

n.trial <- 10000  # Take 10000 samples of
sample.size <- 10  # size 10 (i.e., small) from the population.
sample.stat <- numeric(n.trial)  # Space for storing the 10000 sample medians.

for (i in 1:n.trial) {
  samp <- sample(pop, sample.size, replace = T)  # Take a sample (with replacement).
  sample.stat[i] <- median(samp)  # Compute each sample's MEDIAN.
}

mean(sample.stat)  # Compare the MEAN of sample MEDIANS with pop MEDIAN.

[1] 0.7469

pop.median

[1] 0.6937

sd(sample.stat)  # Compare the sd of sample MEDIANS with population sd.

[1] 0.3094

pop.sd

[1] 1.001

# Note that the formula sigma/root(n) applies only to the sample MEAN.

hist(sample.stat, breaks = 40)
qqnorm(sample.stat, cex = 0.5)
```







The sampling distribution doesn't look too normal. But if the sample size is relatively large, the distribution of a bunch of sample medians, taken from even a non-normal population, is still normal. Most statistics (e.g., sample mean, sample median, sample standard deviation, ...) ultimately end up having a normal distribution, but some require a larger sample size.

5 Confidence Interval

Now, we're going to move on from the sampling distribution, and develop the notion of a Confidence Intervals (CI). First, we will show that the formula for the CI for the population mean actually does what it is designed to do. Recall that the formula for the confidence interval (CI) for the population mean is given by:

$$\bar{x} \pm z^* \cdot \frac{\sigma}{\sqrt{n}} \tag{4}$$

and it is designed to cover the population mean in 95% of samples taken from the population. One nontrivial part of this formula is the $\frac{\sigma}{\sqrt{n}}$, also called the standard error (std err) of the sample mean. It's nontrivial because, first we have to approximate the population std (σ) with sample std, but more importantly, we have to use math to derive it. For many statistics (other than the sample mean), the std error is difficult to derive mathematically. The other nontrivial part of the CI formula is the z^*, because it is based on the fact that the sample mean has a normal distribution. Some statistics do not.

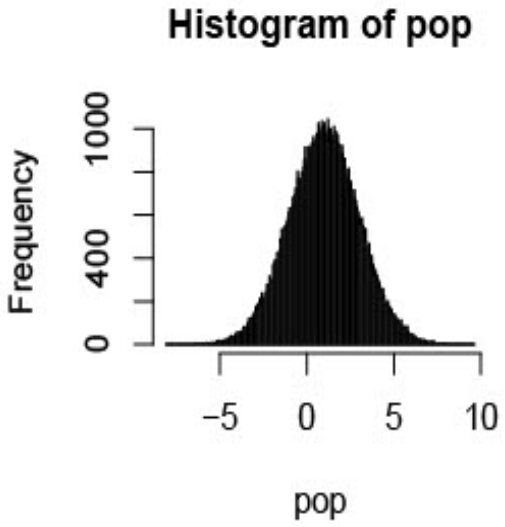
The second task is to show that we can actually get similar answers, WITHOUT using the formula for the std err of the mean, nor the assumption of normality. This is important when simple formulas for the std err do not exist, e.g., for sample median. The main idea is called The Bootstrap: We basically treat the single sample that we have in a realistic situation as if it were the population! So, instead of sampling from the population (i.e., what we did above), bootstrap re-samples from the *sample.* It's like magic, but you'll see how it works below.

There are different kinds of CI, e.g., 1-sample, 2-sample, 1-sided, 2-sided, large-sample, small-sample, etc. And yet other kinds of CI will be covered as we proceed forward into chapters 8, 9, 10, and 11. In the following, you will also come across words like "p-value," or "hypothesis." For now, you may simply ignore them. Ch8 will introduce that method, which is equivalent to the CI method.

5.1 Confidence Interval for Population Mean

As explained at the start of the section on Sampling Distribution, instead of taking samples from a normal population, using rnorm(), we are going to take ONE huge sample from a normal population, using rnorm(), treat it as our population, and then use the R function sample() to take samples from it. The main reason for this is to set the stage for something called "bootstrapping," which we will study later.

```
# Create a normal population.
rm(list = ls(all = TRUE))
set.seed(1)
N <- 100000  # Sample size = 100000.
pop <- rnorm(N, 1, 2)  # Draw N samples from a normal distribution with
                       # mu = 1 and sigma = 2.
pop.mean <- mean(pop)
pop.sd <- sd(pop)
pop.median <- median(pop)
c(pop.mean, pop.sd, pop.median)

[1] 0.9955 2.0070 1.0016

hist(pop, breaks = 400)

sample.size <- 200  # Sample size.
sample.trial <- sample(pop, sample.size, replace = T)  # Here is a sample.
```



5.1.1 Calculating CI Using Formula

```
sample.stat <-  mean(sample.trial)  # Sample mean.
std.err <- sd(sample.trial) / sqrt(sample.size)  # Calculate the standard error.
sample.stat - abs(qnorm(.05 / 2)) * std.err  # Note z_star .
```

```
[1] 0.7777
```

```
sample.stat + abs(qnorm(.05 / 2)) * std.err  # Sign is correct!
```

```
[1] 1.368
```

```
# sample.stat + qt(0.05 / 2, sample.size - 1) * std.err  # For use later
# sample.stat + qt(1 - 0.05 / 2, sample.size - 1) * std.err
```

5.1.2 Calculating CI Using Built-in Function

```
t.test(sample.trial, alternative = "two.sided", conf.level = 0.95)
```

```

One Sample t-test

data:  sample.trial
t = 7.1, df = 199, p-value = 2e-11
alternative hypothesis: true mean is not equal to 0
95 percent confidence interval:
 0.7759 1.3702
sample estimates:
mean of x
    1.073
```

```
# Sample.trial contains the data/measurements of x.
# "two-sided" specifies a 2-sided CI, and 0.95 is the confidence level.

# To get the confidence interval, use the following command:
t.test(sample.trial)$conf.int[1:2]

[1] 0.7759 1.3702
```

Note that answers from the two methods (by formula and by computer) are very similar. The interpretation of C.I. is that we can be 95% confident that the true mean resides in this interval.

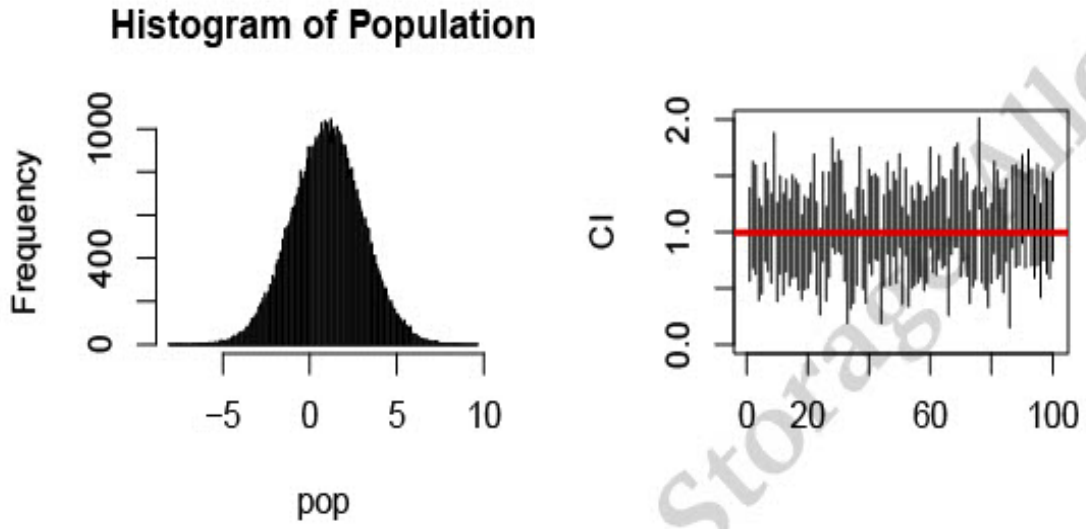
5.2 Coverage of a Confidence Interval

In practice, you will have only one sample (samp) (and not the population (pop)), and so you will build only one CI. But here we want to confirm that the CI, the way we compute it (i.e., with our formulas or with t.test()) covers the population mean the correct percentage of time. **This is what a CI is designed to do: to have the correct coverage.**

To do so, we will draw n.trial = 100 samples of size sample.size = 90 from the normal population, above. For each sample, we will construct the 95% CI and we will make a plot that shows all 100 CIs then count how many of them cover the population mean.

```
rm(list = ls(all = TRUE))
set.seed(1)
N <- 100000
pop <- rnorm(N, 1, 2)
pop.mean <- mean(pop)
pop.sd <- sd(pop)
pop.median <- median(pop)
c(pop.mean, pop.sd, pop.median)

[1] 0.9955 2.0070 1.0016

hist(pop, breaks = 400, main = 'Histogram of Population')

n.trial <- 100  # Number of samples to draw from population.
sample.size <- 90 # Size of each sample = 90.
CI <- matrix(nrow = n.trial, ncol = 2)  # Create space to store n.trial CIs.

for (i in 1:n.trial) {
  sample.trial <- sample(pop, sample.size)  # For each sample/trial,
  CI[i, ] <- t.test(sample.trial)$conf.int[1:2]  # compute (and keep) only CI.
}

count <- 0  # Count number of CIs that cover mu.
for (i in 1:n.trial) {
  if (CI[i, 1] <= pop.mean && CI[i, 2] >= pop.mean) {
    count <- count + 1
  }
}
count

[1] 97
```

```
plot(c(1, 1), CI[1, ], ylim = c(0, 2), xlim = c(0, 101), ylab = "CI", xlab = '',
     type = "l")
for (i in 2:n.trial) {
  lines(c(i, i), CI[i, ])  # Draw CIs (vertically).
}
abline(h = pop.mean, col = "red", lwd = 3)  # The population mean (horizontally).
```



5.3 Two-Sample, Two-Sided Confidence Interval

The following is data from a Statistics class, when students were asked their gender, and what percentage of time they attend class. We will assume percentage is normally distributed, although it is not.

```
dat <- read.table('attend_dat.txt', header = T)
attendance <- dat[, 1]
gender <- dat[, 2]
pa.boy <- attendance[gender == 0]  # Percent of time attending class for boys.
pa.girl <- attendance[gender == 1]  # Percent of time attending class for girls.

n.boys <- length(pa.boy)  # Number of boys. Same as sum(y == 0).
n.girls <- length(pa.girl)  # Number of girls. Same as sum(y == 1).

# The sample mean of these attendance rates is higher for boys than girls:
mean(pa.boy)

[1] 87.57

mean(pa.girl)

[1] 86.4
```

Suppose you wonder if the two true/population means (of attendance rate) are **different**, then, you need to build a 2-sample, 2-sided CI. We will first start by computing 1-sample, 2-sided CIs for each mean: