For Loops, Functions and Botstrappingin R

Michael Anderson and Nick Kortessis

18 September, 2024

```r
library(knitr)
opts_knit$set(root.dir = '/Users/nicholaskortessis/Library/CloudStorage/GoogleDrive-kortessn@wfu.edu/.s
```

In this tutorial, we will:

- Review standard deviation, standard error, why the term "Confidence Interval" is confusing
- introduce 'for()' loops'
- build a bootstrap function to calculate the mean

Let's start by reviewing the standard deviation, standard error, and 95% confidence intervals

Before going too far, it's good to think about the basic function of statistics: to describe variation in the world and to ascribe meaning to that variation. But a first step is to set up a way to think about variability.

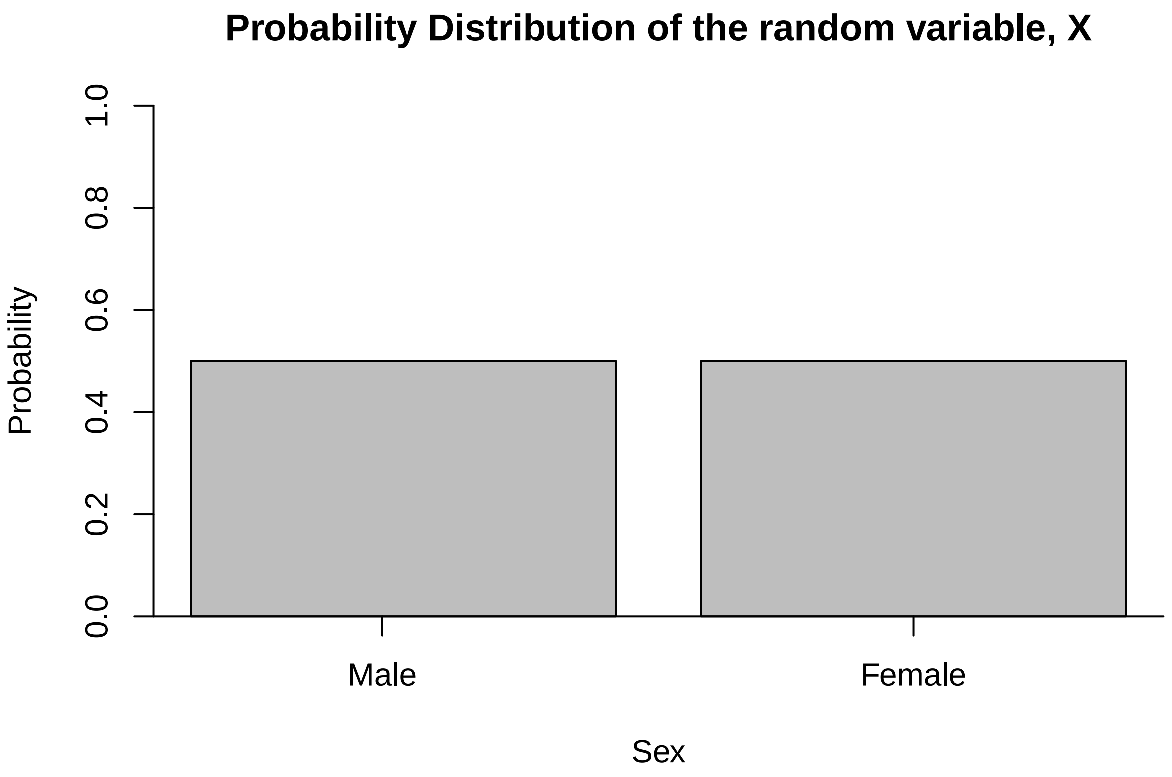
In statistics and probability, we use the concept of a **random variable,** which is an as yet unmeasured individual from a population, to describe and work with variability. It is unmeasured and so can take one of many potential values. The potential values, and their likelihood of being observed from an individual is called a **probability distribution**. This is the essence of variabiliity. If there were no variability, we could be certain of the characteristic of an individual before even looking (and so there would be no need for statistics).

For example, imagine that you were to randomly select a person from the human population and measure whether they were male or female. We would say that X is a random variable representing the sex of this person. To represent the possibility that there might be multiple possible values, we would list all possibilities and give a probability for each one. This random variable is simple enough that we could write something like

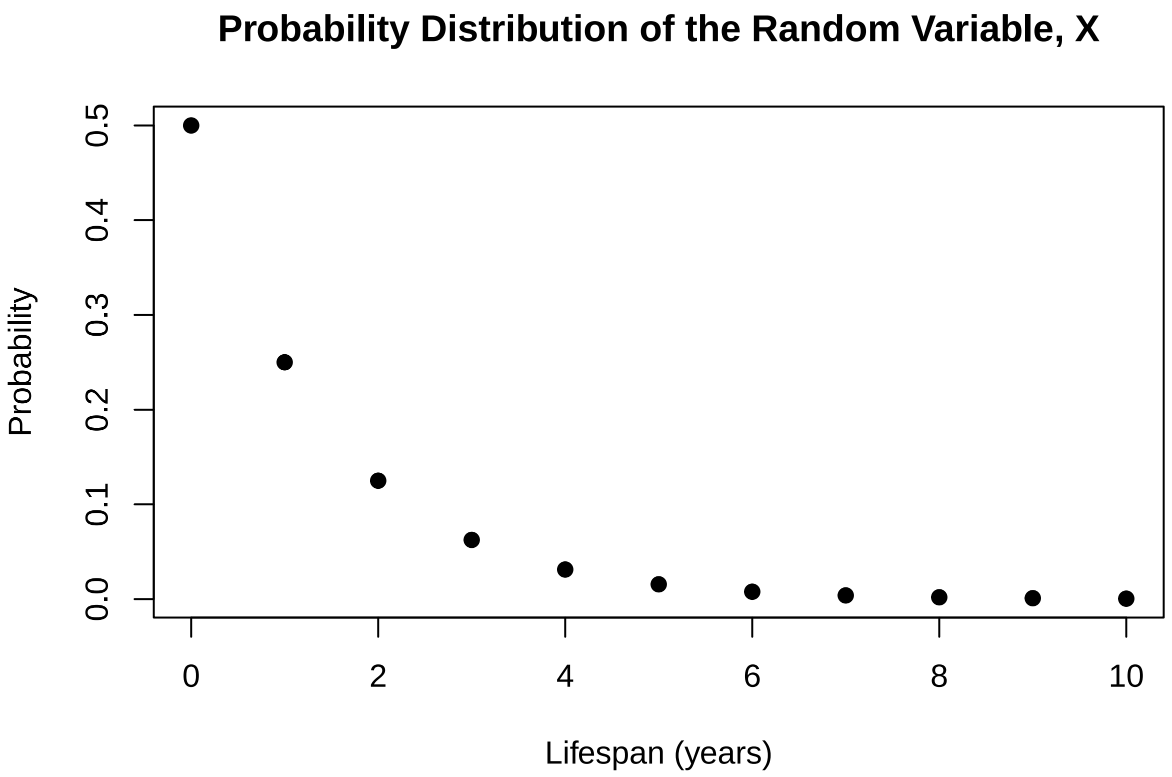
$$X = \begin{cases} \text{female, with probability } p \\ \text{male, with probability } 1-p \end{cases},$$

which describes all possible outcomes of X and the probability of each (we simply define p as the probability that an individual is male).

Here is what this looks like in the case where the sex ratio in the population is even (i.e., $p = 0.5$).



Here is another example. Let X be the random variable describing the lifespan of an individual, i.e., how long it lives. Imagine that we can't know exactly how long someone lives to the exact second, but we just measure how many years the individual lives. The values of X could be $0, 1, 2, 3, \ldots$, and so on. It's harder to write this down concisely, so we might show a graph of the probabilities. Here is an example



This distribution shows that you are most likely to find an individual and measure that it doesn't live a full year and that there is a very low chance of finding any individuals that make it more than 5 years.

In statistics, we like to think that populations can be described by random variables in this way and our job is, in part, to find the properties of these random variables. One of the most important properties of random variables to describe is the variability in outcomes. One measure of variability of a probability distribution is called the **variance**, and typically denoted with the symbol σ^2. The variance has some nice properties but has odd units, so we also typically use the standard deviation, σ. The standard deviation has the same units as the units of measurement, which make it quite interpretable. For example, if we had the following collection of windspeeds, measured in meters per second (m/s)

Day	Sensor	Windspeed (m/s)
01/01/2024	1	1.9
01/01/2024	2	7.4
01/01/2024	3	5.5
01/02/2024	1	10.2
01/02/2024	2	2.1
01/02/2024	3	3.4

The mean windspeed is 5.1 m/s, the variance is 8.9 m^2/s^2, and the standard deviation is 3 m/s.

The standard deviation is a measure of the average difference of any possible outcome from the mean. The standard deviation of a population (or distribution) is often estimated with the *sample standard deviation, s*, which is the square root of the sample variance, s^2. It is calculated as

$$s = \sqrt{\frac{\sum_i^n (x_i - \bar{x})^2}{n-1}}$$

where $\bar{x} = \frac{1}{n}\sum_i^n x_i$ is the mean of the your data, x_i, and n is the sample size. The nice feature of the sample standard deviation is that it, on average, properly estimates the actual standard deviation of a population. This is part of the justification for dividing by $n-1$. If you were to instead divide by n, the estimate would on average *underestimate* the actual variation in the population.

But sometimes there are higher order properties of samples that we would like to describe. For example, we might construct a random variable for the **mean of a sample**, i.e., the mean we would calculate from sampling a population. Or we might want to know estimate what the median of a population is by looking at the sample median. These random variables each have a probability distribution, with all the possible means (or medians) that might arise from random sampling, and the probabilities associated with each. It would be really nice to know what this distribution looks like, right?

In many cases, we don't know. But a good descriptor is how different each sample estimate might be from one sample to the next. We can use the concept of a standard deviation to do this. We give the standard deviation of the sampling distribution a special name, the standard error.

Standard error ($\sigma_{\bar{x}}$) is the standard deviation divided by (standardized by) the square root of the sample size:

$$\sigma_{\bar{x}} = \frac{s}{\sqrt{n}}$$

where s is the sample standard deviation and n is the sample size. These standard errors are used to create confidence intervals, such as a 95% confidence interval.

The phrase "95% confidence intervals" can be confusing because it can mean two different things. Whenever you use this term, be specific (NOTE: this is my own terminology). Confidence intervals are simply ranges where you think entities lie. A 95% confidence interval of the mean is a region where you think the mean lies. A 95% confidence interval of the data says where you think 95% of values from a random variable lie.

- **"95% confidence intervals of the data"** is the mean ± 1.96 sample standard deviations:

  $$\bar{x} \pm 1.96s$$

  There is a 95% chance than an individual that is sampled from the population has a value of x within this window. (Note of caution: this equation only applies when you can assume that the data from a population have a bell-shaped curve. It will lead you astray if the data are asymmetric, bimodal, or have many values multiple standard deviations away from the mean.)

- **"95% confidence intervals on the mean"** is a function of the *standard error*:

  $$\bar{x} \pm 1.96\frac{s}{\sqrt{n}}$$

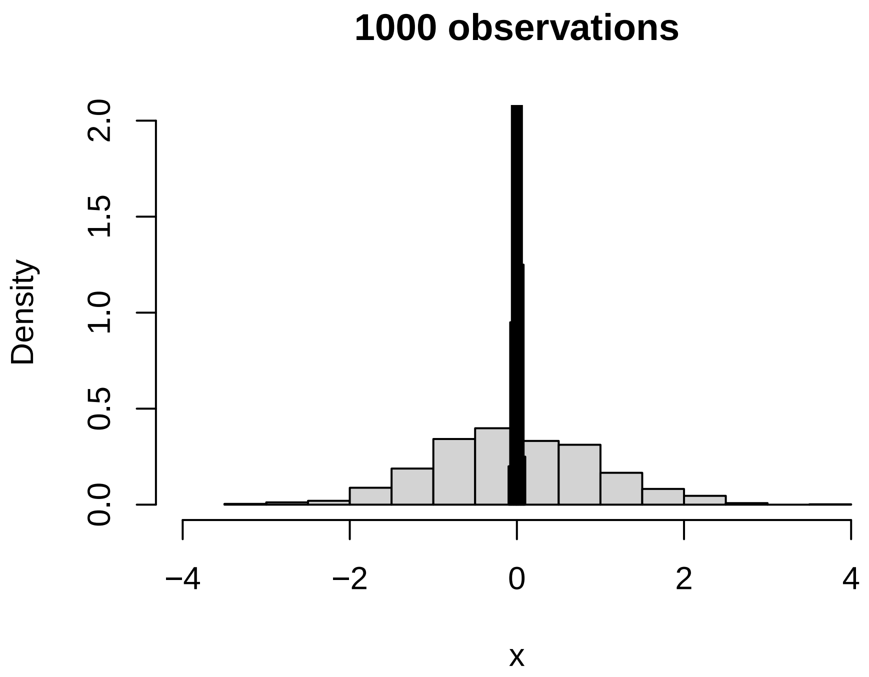
  The mean of a population has a 95% chance of being observed within this range. Unlike the 95% confidence interval of the data, this equation works pretty well, even for data that don't have a bell shaped curve. The only limitation is that your sample size has to be big enough. How big? That's tough to answer. Nonetheless, we will give you tools to get around the issue, even if you don't have many samples.

Usually when people use "95% confidence intervals," they are usually calculating 95% confidence intervals on the mean. In STATISITCAL TESTS, testing whether *a mean is different from zero* or *two means are different from each other* (etc.) usually depends on the 95% confidence intervals on the mean.

However, in MODELING AND BOOTSTRAP, there are many reasons to calculate 95% confidence intervals for many kinds of distributions beyond just the sampling distribution of the mean. Since there are multiple kinds, what's the best practice? When you say "95% CI" for the first time in a paper, define it with equation that you are using or without sufficient explanation that another scientist knows what you mean.

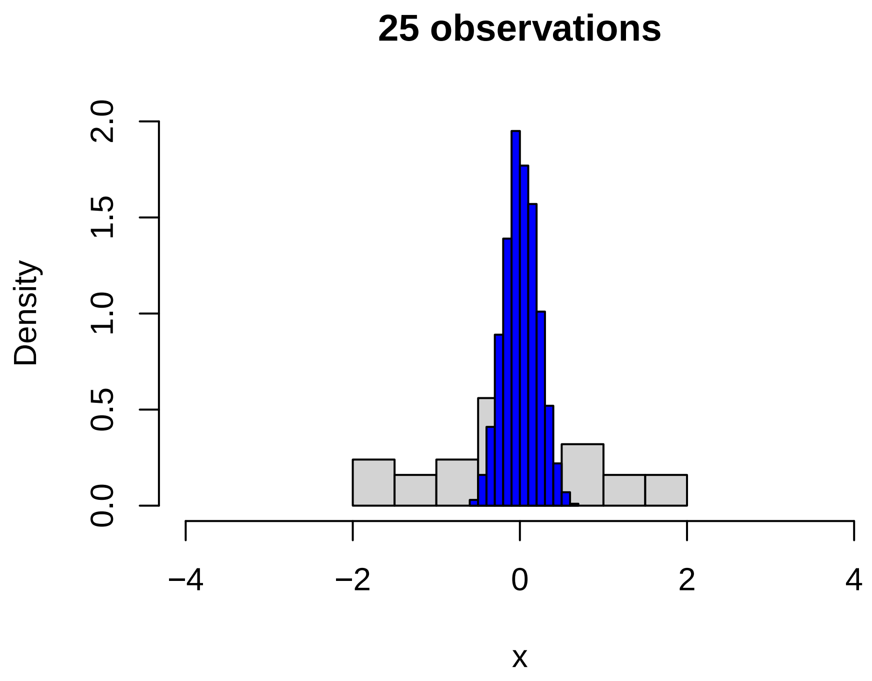
Below we will explore the difference between a 95% confidence intervals of the data vs. a 95% confidence intervals on the mean

```
### First, let's model some data with a normal distribution with 1000 observations
  x <- rnorm(1000)
  hist(x, freq = FALSE, ylim = c(0,2), xlim = c(-4,4), main = "1000 observations")

### Now, let's calculate the mean of that distribution 1000 times
  mean.x <- replicate(1000, mean(rnorm(1000)))
  hist(mean.x, add = TRUE, col = "blue", freq = FALSE)
```



```
### Now, let's do the same thing with only 25 observations
  x <- rnorm(25)
  hist(x, freq = FALSE, ylim = c(0,2), xlim = c(-4,4), main = "25 observations")

### Now, let's calculate the mean of that distribution 1000 times
  mean.x <- replicate(1000, mean(rnorm(25)))
  hist(mean.x, add = TRUE, col = "blue", freq = FALSE)
```

Checkpoint 1: Using the equations above, what is the region where 95% of the data points lie, and what is the region where there is a 95% chance that the mean of the population exists within it? How do these regions differ?



Checkpoint 2: Using the equations above, what is the region where 95% of the data points lie, and what is the region where there is a 95% chance that the mean of the population exists within it? How do these regions compare to the example in which we drew 1000 samples from the distribution?

Let's create a function for a 95% CI that can be stored in our R session and reused /newline Copy (or type out) this function called "My.CI" that takes a vector of observed (normally-distributed) values as input, and outputs a list with the following values:

- standard deviation (OK to use sd())
- standard error
- 95% confidence intervals of the data (upper and lower)
- 95% confidence intervals on the mean (upper and lower)

```r
# create the My.CI function:
  My.CI <- function(x.vect){
    # x.vect should be normally distributed
    sd.x <- sd(x.vect)
    n <- length(x.vect)
    se.x <- sd.x/sqrt(n)
    data.95.CI <- mean(x.vect)+1.96*c(-1,1)*sd.x

    mean.95.CI <- mean(x.vect)+1.96*c(-1,1)*se.x

    return(list(sd=sd.x, se=se.x, mean= mean(x.vect),
                data.95 = data.95.CI,
                mean.95 = mean.95.CI
                )
          )
  }
```

```
# now produce all the output from the functions on x
  My.CI(x)
```

```
## $sd
## [1] 1.013363
##
## $se
## [1] 0.2026725
##
## $mean
## [1] -0.06438108
##
## $data.95
## [1] -2.050572  1.921809
##
## $mean.95
## [1] -0.4616192  0.3328570
```

Note that the above function assumes your data fit a normal distribution. What do you do when your data does not fit a normal distribution?

Bootstrap

What if you want to do a statistical test, such as test if your data from 2 groups are different (equivalent of a t-test), or test if the slope of two variables is different from 0—but your data does not fit any known distribution.

You test these hypotheses by doing a *bootstrap*. A bootstrap comes from the old adage "pulling oneself up by one's bootstraps". The idea is to use what you have — your data — to come up with a best guess of the distribution without relying on other assumptions (like the assumption of normality).

Bootstrapping works by resampling your data many times. In doing this, you are making the assumption that the general features of your data match most of the general features of the distribution that the random variable that generated them. And by resampling your data many times, you can create sampling distributions. These sampling distributions can then be used to estimate quantities (such as medians, interquartile ranges, maxima, etc.) and to test statistical hypotheses.

To do a bootstrap, we need a couple more programming tools:

`for()` loops A `for()` loop will loop through your code as many times as you tell it. Note that for() loops can be slow in R if your code is complicated. Most people suggest using the `apply` family of functions built into R. We'll discuss this more throughout the course. Here's a quick example.

```
  for (i in 1:100){
  print(i)
  }
```

This for loop just counts from 1 to 100 and then for each iteration of the loop, it prints the current counting number `i`.

Let's do a calculation on a number at each step of the loop, and store it in an output vector.

```
# create an empty vector to store your output
  my.output <- c()
  my.input <- seq(53, 128, by=5)

# In this case, we want to loop through each element of my.input,
```

```r
# do a calculation, and save it in my.output
  for (i in 1:length(my.input)){
    my.output[i] <- my.input[i]^2 + my.input[i]
  }

  my.output
```

The `sample()` function This function is a great way to resample your data.

```r
  my.input
```

```
##  [1]  53  58  63  68  73  78  83  88  93  98 103 108 113 118 123 128
```

```r
# Let's resample the vector my.input with replacement
  sample(my.input, replace = TRUE)
```

```
##  [1] 113 108  78  83  68  73  93  58 108 123  68 113 108  93 123 128
```

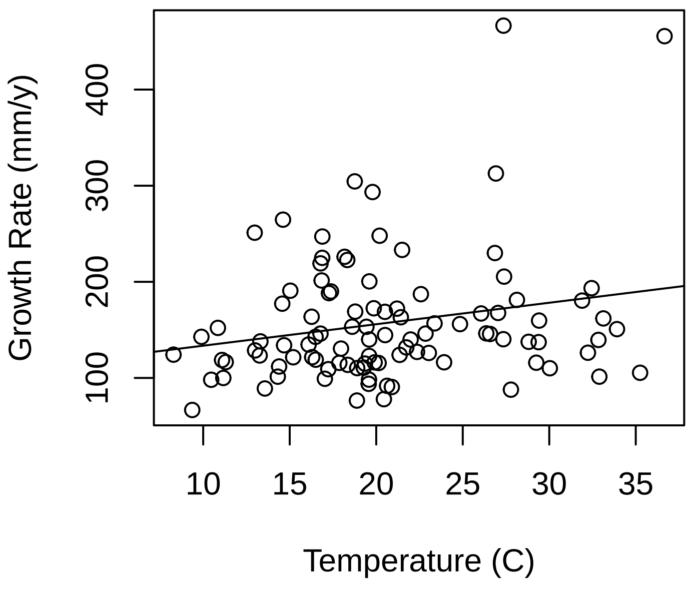
```r
# Is this the same or different from my.input?

# Let's resample the names of two groups 100 times, with different probabilities
  s1 <- sample(c("Group 1", "Group 2"), size = 100, replace = TRUE,prob = c(0.3, 0.7))
  table(s1)
```

```
## s1
## Group 1 Group 2
##      27      73
```

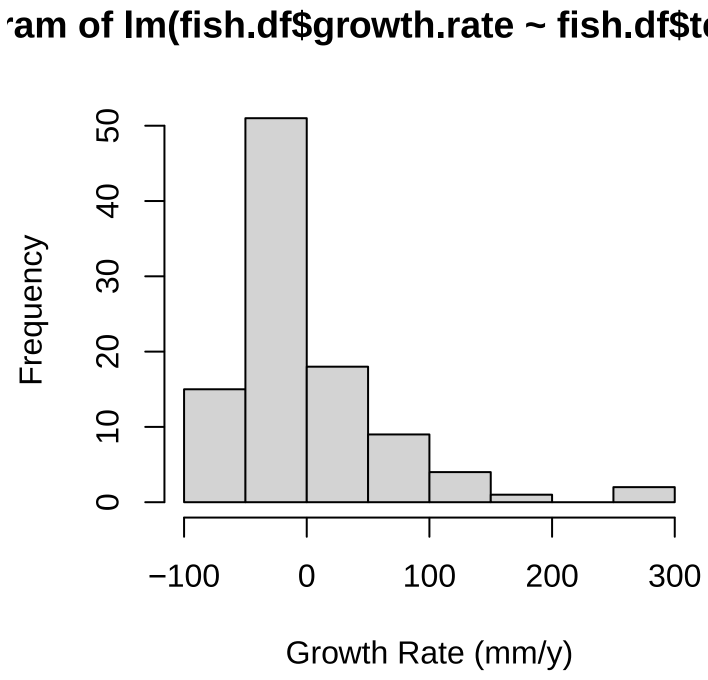
Bootstrap example: test the null hypothesis that the slope between two variables does not equal 0.

Upload the dataset "FishGrowthTemp.txt". This data is from an experiment where we measured fish growth rate (mm) per year as a function of temperature in degrees C.

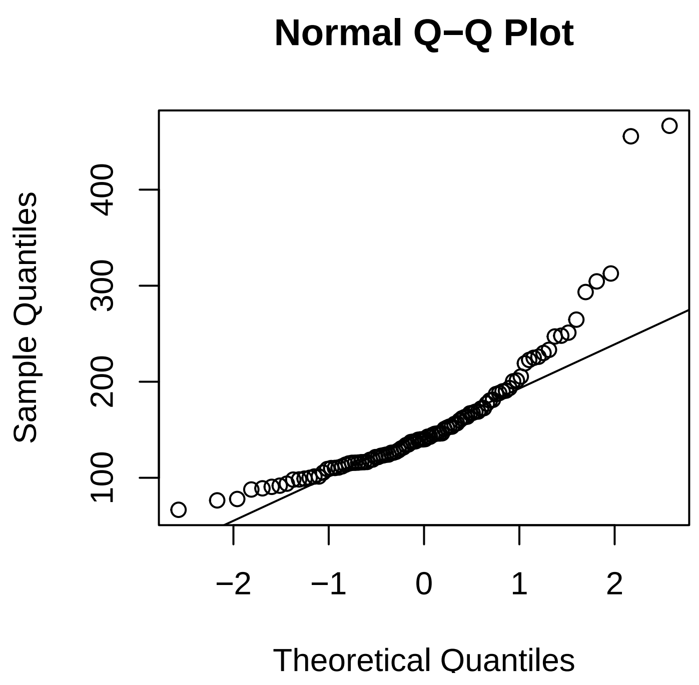
```r
# Upload the data
  fish.df <- read.csv("data/FishGrowthTemp.csv", header=TRUE)

# Plot the data
  par(mfrow=c(1,1))
  plot(fish.df$temp, fish.df$growth.rate,
       xlab = 'Temperature (C)', ylab = 'Growth Rate (mm/y)')
  abline(lm(fish.df$growth.rate~fish.df$temp))
```



```
# This model assumes residuals are normally distributed.
# Let's take a look at them. Are they normally distributed?
  hist(lm(fish.df$growth.rate~fish.df$temp)$residuals,
         xlab = 'Growth Rate (mm/y)')
```



```
# Remember the q-q plot? This shows us the same thing in a different way.
  qqnorm(fish.df$growth.rate)
  qqline(fish.df$growth.rate)
```



```
# This is the slope we observe from a linear model (y = ax + b)
  lm(fish.df$growth.rate~fish.df$temp)$coef
```

```
##  (Intercept) fish.df$temp
##   111.191158     2.234132
```

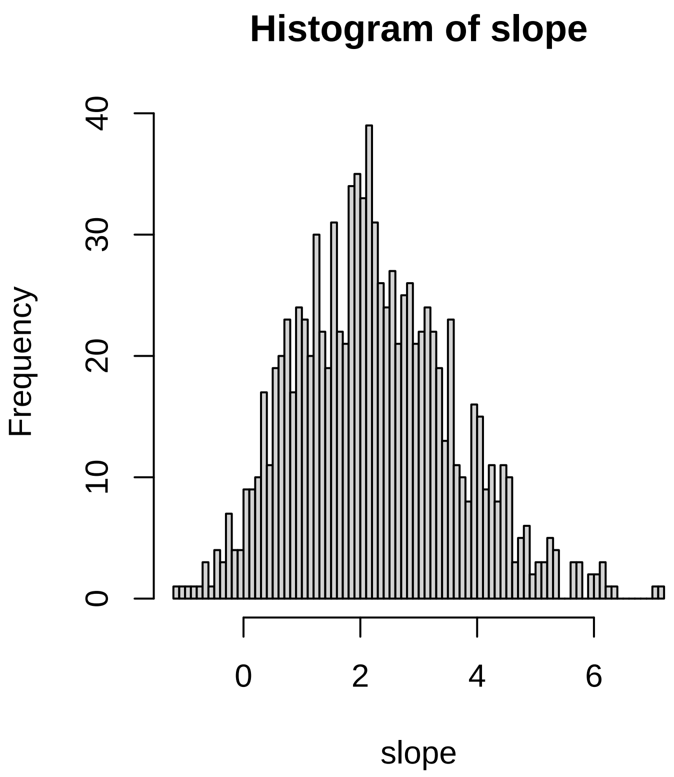
```
# Our slope is positive!  Maybe it will be significant based on a statistical test that assumes the nor

# Let's do a statistical test:
  summary(lm(fish.df$growth.rate~fish.df$temp))
```

```
##
## Call:
## lm(formula = fish.df$growth.rate ~ fish.df$temp)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -85.47 -38.79 -16.66  16.26 294.19
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)   111.191     22.175   5.014 2.37e-06 ***
## fish.df$temp    2.234      1.025   2.180   0.0317 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 65.33 on 98 degrees of freedom
## Multiple R-squared:  0.04625,    Adjusted R-squared:  0.03652
## F-statistic: 4.752 on 1 and 98 DF,  p-value: 0.03166
```

Checkpoint 3: How do we interpret this statistical test given that our residuals are non-normal (and so the test isn't valid)?

Let's do a bootstrap to see if our slope is different from 0. We will resample the dataframe 1000 times and calculate whether our resampled data is different from 0. The code for this process is below:

```
# creat a sample size
  n.boot <- 1000

# fill it with
  slope <- rep(NA, n.boot)
  n.rows <- nrow(fish.df)

# create the slopes
  for (i in 1:n.boot){
    new.rows <- sample(1:n.rows, replace=TRUE)
    new.data <- fish.df[new.rows,]
    slope[i] <- lm(new.data$growth.rate ~ new.data$temp)$coef[2]
  }

# Let's look at our distribution of slope:
  hist(slope, breaks=100)
```



```
  sort(slope)[n.boot*0.025] # Lower 95% CI on distribution
```

```
## [1] -0.1402998
```

```
  sort(slope)[n.boot*0.975] # Upper 95% CI on distribution
```

```
## [1] 5.202787
```

Checkpoint 4: What does the bootstrap analysis suggest about our data? What if we had done more bootstraps? Why is looking at the standard error useless in a bootstrap?

Bootstrap discussion: How to resample your data

How you decide to resample your data will depend on the experimental design.

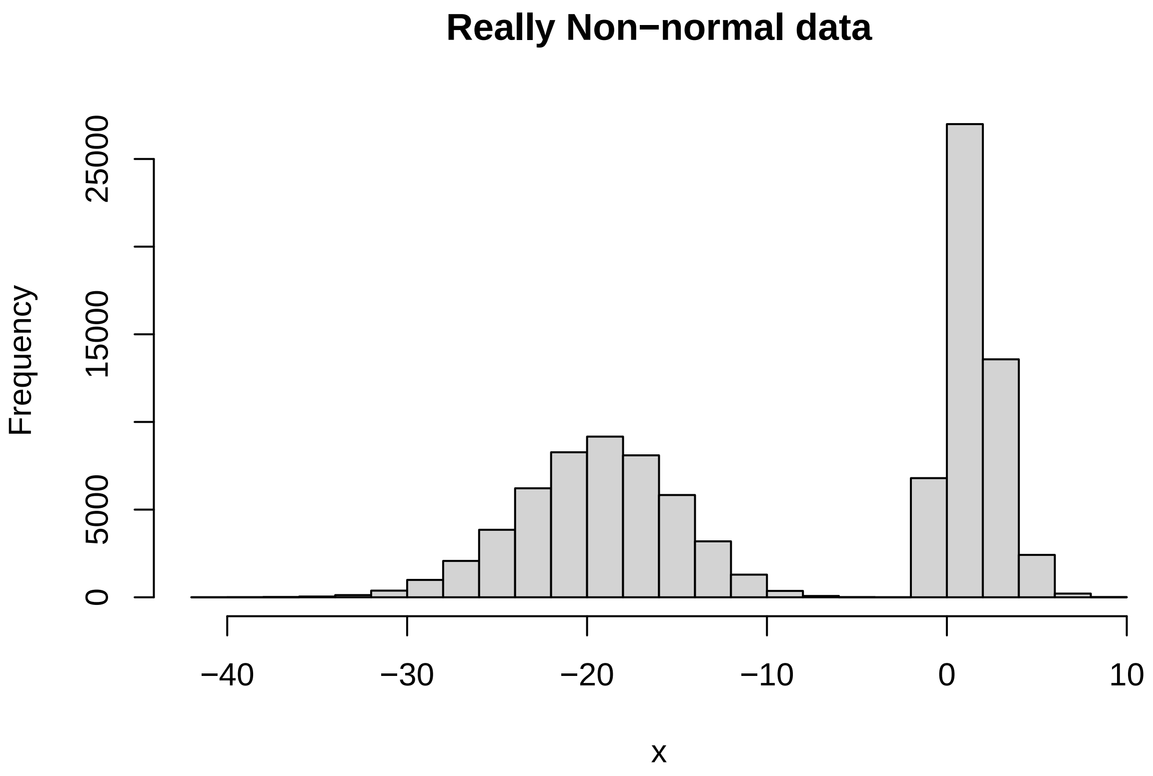
If you have an experiment with factor levels, do you want to resample data within factors, or resample data across all factors?

Bootstrap may be used to (i) estimate uncertainty in a parameter, or (ii) create a null distribution against which to compare your observation. A benefit of using bootstrap is that it generally does not require your data to follow a distribution.

Using bootstrapping to illustrate that the standard error works for the mean.

I told you before that you can generally trust the equation for the 95% confidence interval of the mean, even if your data aren't normally distributed. Maybe you don't trust me. Let's take a look at some VERY non-normal data.

```r
n = 100000
set.seed(1)
x1 <- rpois(n/2, 2)
x2 <- rpois(n/2, 19)

data <- c(x1, -x2)
hist(data, xlab = 'x', main = 'Really Non-normal data')
```



Data doesn't normally look like this, but that's not really the point. This is an extreme example to illustrate a point. The mean of this population is

```r
mean(data)
```

```
## [1] -8.49602
```

Let's sample this with n = 30 individuals and see what we get.

```r
set.seed(1)

my.sample <- sample(data, 30, replace = TRUE)
mean(my.sample)
```

```
## [1] -7.333333
```

Hey, that's not too bad. Let's see what the confidence interval says

```r
se <- sd(my.sample)/sqrt(length(my.sample))
mean.95 <- mean(my.sample) + 1.96*c(-1,1)*se
print(mean.95)
```

```
## [1] -10.818435  -3.848232
```

This says there is a 95% chance that the true mean is in the range [-10.82, -3.85]. That is pretty good. The real mean is -8.5. Let's see what the sampling distribution for the mean looks like

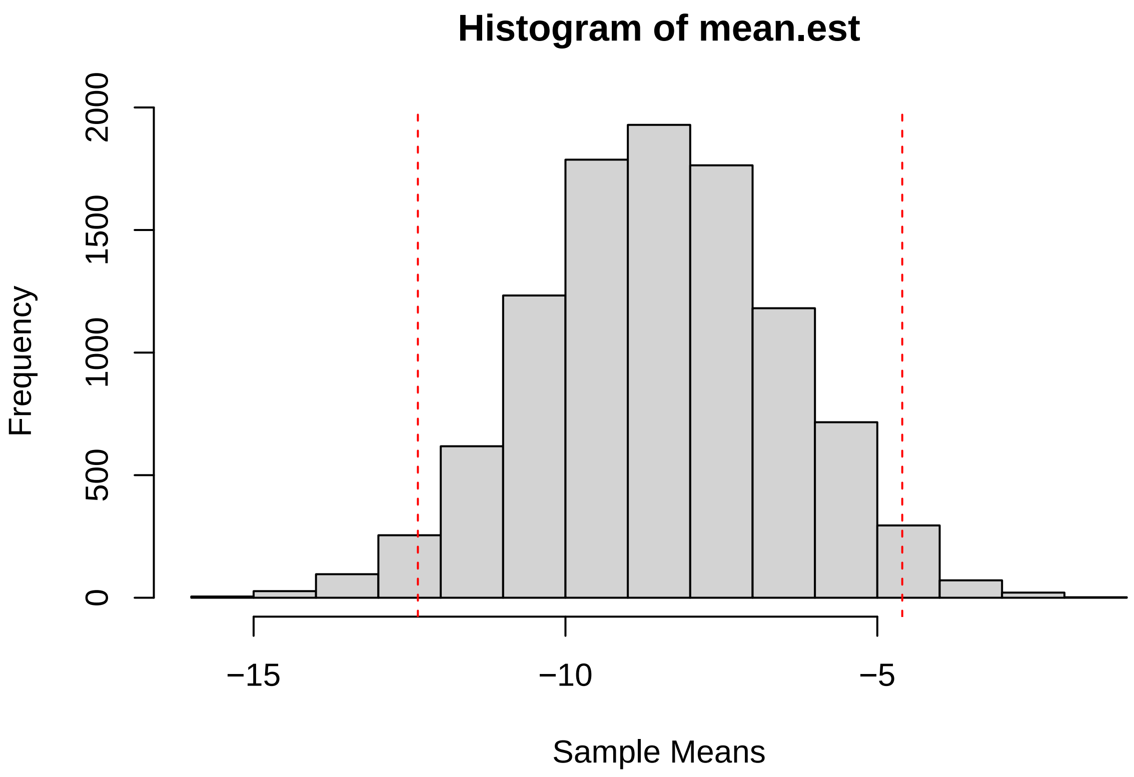
```r
n.boot <- 10000
mean.est <- rep(NA, n.boot)

for (i in 1:n.boot){
  my.sample <- sample(data, 30, replace = TRUE)
  mean.est[i] <- mean(my.sample)
}

# The acutal 95% of sample means are in the range
  hist(mean.est, xlab = 'Sample Means')
  sort(mean.est)[n.boot*0.025] # Lower 95% CI on distribution
```

```
## [1] -12.36667
```

```r
  sort(mean.est)[n.boot*0.975] # Upper 95% CI on distribution
```

```
## [1] -4.6
```

```r
  abline(v = c(sort(mean.est)[n.boot*0.025], sort(mean.est)[n.boot*0.975]),
         lty = 2, col = 'red')
```



The 95% confidence interval relies on the fact that a distribution is normal. But look at this sampling distribution of the mean. It looks normal! Even for highly non-normal **data**, the **sampling distribution of the mean** is normal. This justifies the fact that the equation for the 95% CI of the mean can be trusted, even for non-normal data. This point is based on a well-known, famous proof in mathematics called **the central limit theorem**.

However, if you are ever in doubt about whether equations or tests are appropriate, you can always resort to bootstrapping. It won't let you down (or at least won't let you down any more than any regular statistical test will)!