Lab 5: The Normal Distribution

Your name here

Date here

General information

This lab is due February 24th by 11:59 pm. You must upload your .rmd file and knitted PDF to Canvas. This lab is worth 10 points. You are welcome and encouraged to talk with classmates and ask for help. However, each student should turn in their own lab assignment and all answers, including all code, needs to be solely your own.

Objective

The goal of this lab is to get familiar working with the normal distribution, including calculating area under the curve and probability of observations. You will also learn to run one and two-sample t.tests and interpret their results.

This lab may be much easier to follow if you read through it in PDF format first. There are some symbols and formulae that are rendered in PDF format but not in R markdown.

The normal distribution

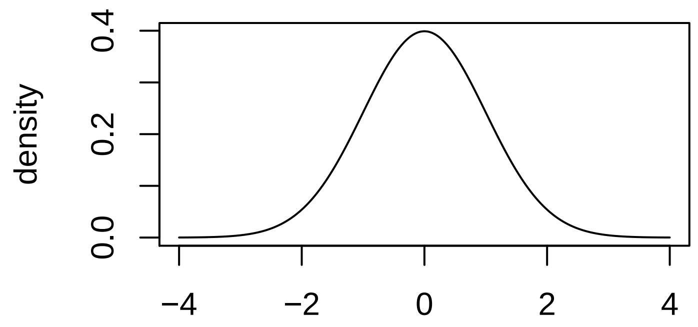
The normal distribution is *the* classic distribution that all of statistics is built on. The default normal distribution has a mean of 0 and a standard deviation of 1.

Let's start by running the below code chunk, which creates a function for visualizing a normal distribution. (You don't have to memorize this code.)

```r
# Here is a custom function for visualizing a normal
# distribution.  You may not have seen a function defined
# this way before. This lets us create our own function that
# can be used in our code, like other R functions. You only
# need to define a function once.
plot.normal.curve <- function(mu, sd, qleft = NULL, qright = NULL) {
    s = sd * 4
    plot(seq(mu - s, mu + s, by = 0.01), dnorm(seq(mu - s, mu +
        s, by = 0.01), mean = mu, sd = sd), type = "l", ylab = "density",
        xlab = "")
    if (is.null(qleft) == FALSE) {
        x = seq(mu - s, qleft, by = 0.01)
        y = dnorm(x, mean = mu, sd = sd)
        polygon(c(mu - s, x, qleft), c(0, y, 0), col = "red")
    }
    if (is.null(qright) == FALSE) {
        x = seq(qright, mu + s, by = 0.01)
        y = dnorm(x, mean = mu, sd = sd)
        polygon(c(qright, x, mu + s), c(0, y, 0), col = "red")
    }
}
```

And finally, call our new function to visualize a standard normal distribution.

```
# note: We are specifying the desired figure dimensions in
# the chunk setup {} with fig.width and fig.height. Units by
# default are inches, and default figure dimensions are 7'
# wide by 5 ' high.
plot.normal.curve(mu = 0, sd = 1)
```



There are a few built-in calculations in R that are commonly used to work with the normal distribution.

1. `rnorm(n, mean, sd)` randomly generates *n* random numbers from a normal distribution with your specified *mean* and *standard deviation*.

```
# generate 15 random numbers from a normal distribution
# normal distribution has a mean of 6.7, sd of 2
rnorm(n = 15, mean = 6.7, sd = 2)
```

```
##  [1]  5.796276  5.260882  7.583356  9.665981  6.022691  5.232150  7.265932
##  [8]  9.330810  8.833077  7.459243 10.811255  1.988614  6.325618  9.577186
## [15]  6.931972
```

When you ran the `rnorm` code (above) I have no idea what you would see. However, when you specify the *seed* to be 20 and then run the `rnorm` code, I know that you will see this:

[1] 9.0253706 5.5281511 10.2709300 4.0348126 5.8068665 7.8392122 0.9205648 [8] 4.9619633 5.7765946 5.5889182 6.6597293 6.3992356 5.4437465 9.3464417 [15] 3.6572989

Try it yourself! Does the output match?

```r
# generate 15 random numbers from a normal distribution with
# mean of 6.7, sd of 2 set seed to 20 so output is
# reproducible every time > note: seed number selected is
# arbitrary (you can pick anything number you like) but 20
# will reproduce output in text narrative above
set.seed(20)
rnorm(n = 15, mean = 6.7, sd = 2)
```

```
##  [1]  9.0253706  5.5281511 10.2709300  4.0348126  5.8068665  7.8392122
##  [7]  0.9205648  4.9619633  5.7765946  5.5889182  6.6597293  6.3992356
## [13]  5.4437465  9.3464417  3.6572989
```

Basically, R randomly picks a 'seed' in each call to its random number generator. You can override this by defining what seed you want R to use. In most cases you will *not* want to use *set.seed()* but I thought it would be helpful for you to know this.

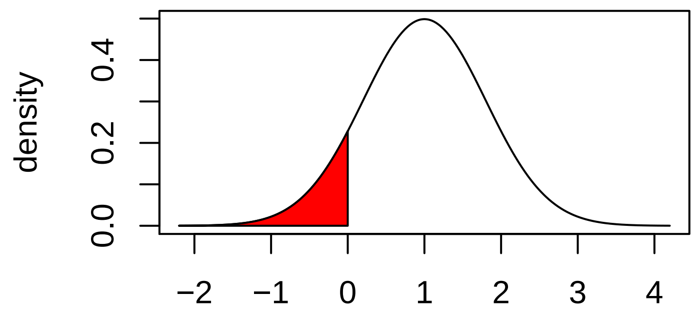
Here endeth the aside!

2. `pnorm(q, mean, sd)` gives the area under the normal curve *to the left* of value *q*. In other words it gives the probability of a value less than or equal to q. To get the area under the curve to the *right* of *q*, you can specify `pnorm(q, mean, sd, lower.tail=FALSE)`. Or since the area to the left and right of this value must sum to one we could also use *1 - the area to the left.*

```r
# probability of a value of 0 or LESS, from a distribution
# with mean of 1 and sd of 0.8
pnorm(q = 0, mean = 1, sd = 0.8)
```

```
## [1] 0.1056498
```

```r
# use the function we created above to plot the probability
# density function and shade the area under the curve to the
# left of q
plot.normal.curve(mu = 1, sd = 0.8, qleft = 0)
```

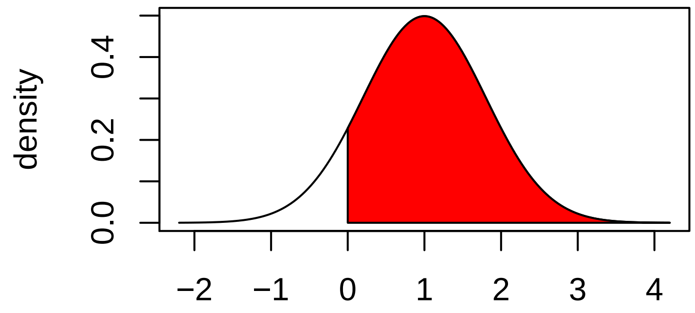


```
# the shaded area here visualizes the probability given in
# the previous pnorm line (about 11% of the area under the
# curve)
```

```
# probability of a value of 0 or MORE, from the same
# distribution
pnorm(q = 0, mean = 1, sd = 0.8, lower.tail = FALSE)
```

```
## [1] 0.8943502
```

```
# use the plotting function created above and shade the area
# under the curve to the right of q
plot.normal.curve(mu = 1, sd = 0.8, qright = 0)
```



```
# the shaded area here visualizes the probability given in
# the previous pnorm line (about 89% of the area under the
# curve)
```

```
# alternative way to get the same value
1 - pnorm(q = 0, mean = 1, sd = 0.8)
```

```
## [1] 0.8943502
```

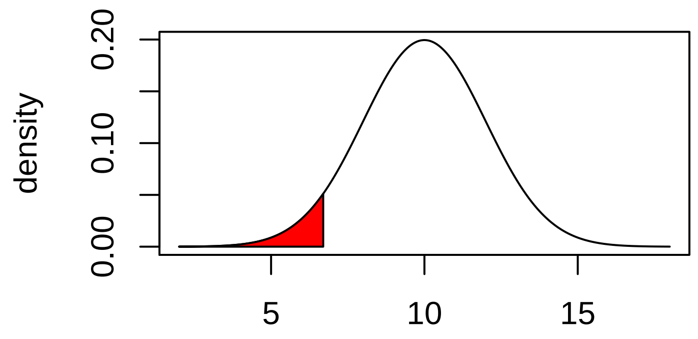
3. `qnorm(p, mean, sd)` gives the inverse of the *pnorm* function. In other words, it gives the value at which the cumulative distribution function is p for your specified mean and standard deviation.

```
qnorm(p = 0.05, mean = 10, sd = 2)
```

```
## [1] 6.710293
```

```r
# there is a 5% chance of observing a value of 6.7 or lower
# from this distribution
```

```r
plot.normal.curve(mu = 10, sd = 2, qleft = 6.7)
```



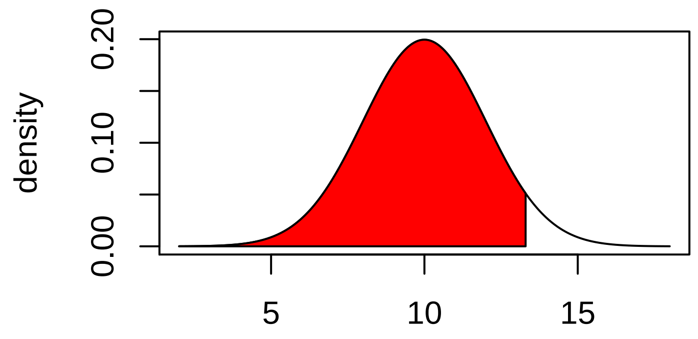
```r
# the shaded area under the curve represents 5% of the
# probability density function, from the lower tail
```

```r
qnorm(p = 0.95, mean = 10, sd = 2)
```

```
## [1] 13.28971
```

```r
# 95% of the distribution lies below 13.3
```

```r
plot.normal.curve(mu = 10, sd = 2, qleft = 13.3)
```



```
# the shaded area under the curve represents 95% of the
# probability density function, from the lower tail
```

Calculating probability from the normal curve

For the following questions, it might be useful to sketch out the standard normal curve, so you can estimate if your answer corresponds to the correct tail of the distribution (e.g. does a given value of X seem very likely or not likely at all?)

Question 1 (1 point): Find the probability of observing a value less than or equal to -2.5, for the standard normal curve. Print out the value.

```
# Your code here
```

Question 2 (1 point): Find the probability of observing a value greater than or equal to 3, for the standard normal curve. Print out the value.

```
# your code here
```

Question 3 (0.5 point): What is the total area under any normal distribution? Why?

your answer here

Question 4 (0.5 point): Using one of the functions described above, sample 10 random numbers from a normal distribution with $\mu = 4$ and $\sigma = 1.5$. Graph a histogram of your sample. Calculate your sample mean ($\bar{X}$). How close is your sample mean ($\bar{X}$) to the true parameter μ? Calculate your sample standard deviation (s). How close is your sample standard deviation (s) to the true parameter σ?

```
# Your code here
```

your answer here

Question 5 (1 point): Repeat what you did above, but this time sample 100 random numbers from the distribution, and then repeat this by sampling with n=1,000. What happens to your estimates of μ and σ as sample size increases, and why?

```
# your code here
```

your answer here

Question 6 (0.5 point). For a normal distribution with $\mu = 50$ and $\sigma = 0.5$, calculate the X value at which only 2.5 percent of the area underneath the curve is to the RIGHT. Confirm this using $pnorm()$. Take a look at the functions I've described at the beginning of the lab and pick the appropriate function. Sketch out the curve first to help you decide how to do this.

```
# Your code here
```

The t-distribution

The t-distribution is also a continuous probability distribution that is very closely related to the normal distribution. We use it to approximate a normal distribution when you have a small sample size and the true population's standard deviation is unknown. The vast majority of scientific studies fall into this camp (small number of observations relative to the population, and an unknown true parameter).

So, instead of comparing a test statistic to a *normal* distribution, we usually compare it to the *t-distribution*, which ends up being a little more conservative.

There are analogous functions for working with the t-distribution, that are very similar to the functions described above for the normal distribution:

1) `rt(n, df)` for drawing a random sample with size=n from a t-distribution with degrees of freedom df

(2) `pt(q, df)` for calculating the area under the curve to the left of q, for a t-distribution with degrees of freedom df

(3) `qt(p, df)` for calculating the value of q for which the area under the curve to the left is equal to p.

Note that when we refer to the t-distribution we are no longer specifying the mean and sd, but now we need to specify the df since this affects the shape of the t-distribution. When the sample size gets very large, the t-distribution essentially becomes the same as the Z-distribution.

T-tests: one-sampled

We use the t-distribution to test whether an observed mean ($\bar{X}$) is different from the mean under the null hypothesis (μ_0). Suppose we are interested in testing whether human body temperature was different than the supposed healthy body temperature of $\mu_0 = 98.6$. Thus their null hypothesis is:

H_0: mean(body temperature) is equal to 98.6

H_A: mean(body temperature) is not equal to 98.6

Researchers collected data from a random sample of people. The data are:

```
heat <- read.csv(url("http://www.zoology.ubc.ca/~schluter/WhitlockSchluter/wp-content/data/chapter11/cha
head(heat)  # print first several rows in dataset
```

```
##   individual temperature
## 1          1        98.4
## 2          2        98.6
## 3          3        97.8
## 4          4        98.8
## 5          5        97.9
## 6          6        99.0
```

```
summary(heat)  # preview values in columns
```

```
##    individual  temperature
##  Min.   : 1   Min.   : 97.40
##  1st Qu.: 7   1st Qu.: 98.00
##  Median :13   Median : 98.60
##  Mean   :13   Mean   : 98.52
##  3rd Qu.:19   3rd Qu.: 99.00
##  Max.   :25   Max.   :100.00
```

Question 7 (0.5 point): Calculate the mean body temperature ($\bar{X}$) of their sample. Print the answer.

```r
# your code here
```

The mean of the sample is not 98.6 degrees, exactly. But is it far off enough that we can reject the null hypothesis? What is the probability of observing our sample mean, or a mean more extreme, if the null is true ($\mu = 98.6$)?

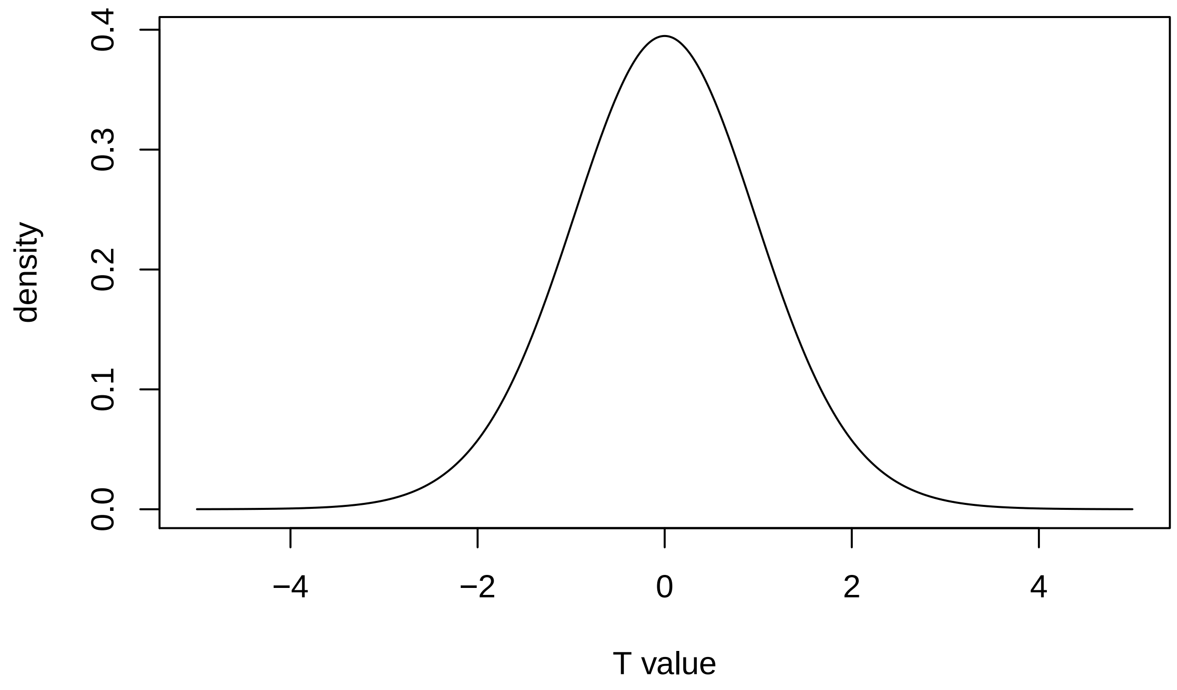
If you've been paying attention to the first part of the lab, you might cleverly think that we could simply create a normal distribution with a mean of 98.6, and then calculate the area underneath the curve to the left of our observed mean. That would give us the probability of observing our sample mean or one even lower, if the true population mean is 98.6 degrees. We could multiply by 2 to get our P-value.

But, there's a big problem. We can't construct this null normal distribution because we don't know what the standard deviation of the population (σ) is! Remember, to draw a normal distribution, we need to know the mean AND the standard deviation. So, instead of comparing our observed mean to a normal distribution, we need to compare our observed mean to a *t-distribution*.

The t-distribution is a null distribution that illustrates the mean of a sample taken from a normally distributed population. Its shape is determined by the degrees of freedom. *The degrees of freedom is the number of samples minus 1.* The t-distribution is *not* described by a population standard deviation, which is great, because we don't know it!

Let's take a look at the t-distribution using the correct degrees of freedom for the temperature dataset:

```r
plot(seq(-5, 5, by = 0.01), dt(seq(-5, 5, by = 0.01), df = length(heat$temperature) -
    1), type = "l", ylab = "density", xlab = "T value")
```



If we want to compare our mean to the t-distribution, we run into some trouble. Our sample mean is nowhere near the curve, because the t-distribution is centered around 0, not 98.6. In order to compare our sample mean to the t-distribution, we need to *standardize* it. The standardized sample mean is also known as the t-statistic, and is the test statistic for the t-test. To get the *test statistic (t)*, we use the formula below (Note that the formula below will display nicely in the knitted PDF but is hard to read in markdown):

$$t = \frac{\bar{Y} - \mu_0}{SE_{\bar{Y}}}$$

In other words, we subtract the null hypothesis mean from our observed sample mean, then divide by the standard error of our sample. Your book uses $\bar{Y}$ to denote a sample mean; this is equivalent to $\bar{X}$ which is also sometimes used to denote a sample mean.

Question 8 (1 point): Calculate the t-statistic for your observed data and report it below. It may be useful to define variables like n, SE first, before doing the actual calculation, to avoid having a terribly messy fraction. *Do not use raw numbers; use functions in R to calculate things like n, df, etc.* If you are not sure where to begin, check out the example in the book, on p. 310. Comment your code so I can understand your approach.

```
# your code here
```

your answer here

Question 9 (0.5 point): Calculate the probability of observing a test statistic as or more extreme as yours, using the function 'pt'. It may be helpful to follow along in the book or lecture slides. Report the probability.

```
# your code here
```

your answer here

Question 10 (1 point): What does the probability you calculated tell you about the null hypothesis that the average body temperature is 98.6?

your answer here

You just ran a one-sample T-test by hand!! You calculated a test statistic from your observed mean (t), and compared it to a null t-distribution to get a P-value. By now, this theme of hypothesis testing should seem familiar:

(1) define null and alternative hypotheses

(2) generate a null distribution for a test statistic

(3) calculate a test statistic from the observed data

(4) compare your test statistic to the null distribution to get a P-value

Another way to think of the t-distribution is that it is essentially a sampling distribution for the t-statistics we might get if we took many different samples with a certain degree of freedom, and if the null hypothesis were true. Thus, some t values might be pretty large or pretty small (when the sample mean is far from the null mean), but if the null hypothesis is true, those events should be rare.

Example 1: Running one and two-sample t-tests using R's functions

There is a function in R that does steps 2-4 of the t-test automatically. This is the `t.test()` function. There are many different ways to customize the function to run the specific test you need, as well as add different types of corrections. Check out the ?t.test help page for information.

One-sample test

The simplest version of a t-test is the one-sample, for comparing the mean of a sample to a null mean. One sample t-tests are used to answer the question: is the true population mean μ equal to some hypothesized mean μ_0? This is what you just ran, by hand. To run this in R, you need to give the function two arguments. The argument `x =` is a vector containing all your sample measurements, and the second argument `mu =` gives your expected mean under the null hypothesis.

For instance: `t.test( x = mydata$height, mu = 155)` would test whether the true population mean is equal to 155, given a sample dataset called mydata$height. R calculates the degrees of freedom automatically, using the number of data points.

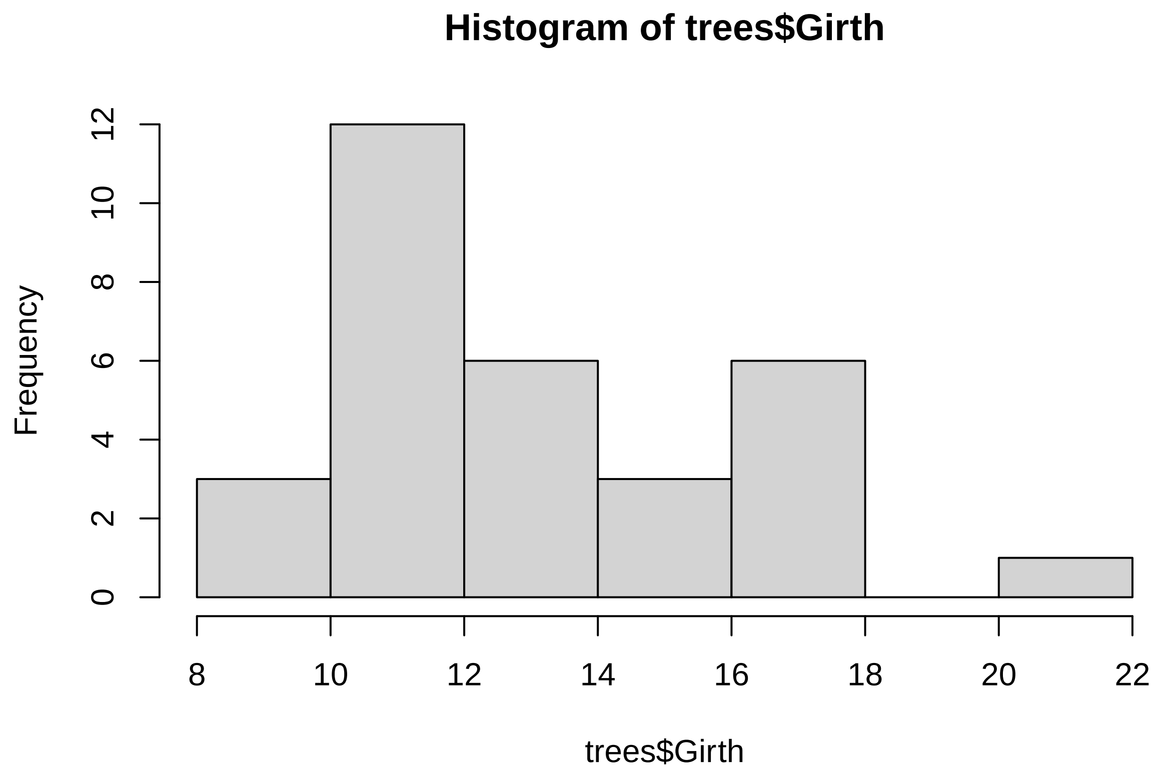
The default test is a two-tailed test. For more modifications, see ?t.test

For the next few problems, let's work with the Black Cherry tree dataset. This gives the measurements of cherry trees that were gathered for timber.

```
head(trees)
```

```
##   Girth Height Volume
## 1   8.3     70   10.3
## 2   8.6     65   10.3
## 3   8.8     63   10.2
## 4  10.5     72   16.4
## 5  10.7     81   18.8
## 6  10.8     83   19.7
```

```
hist(trees$Girth)
```



The forester wants to know if the stand of cherry trees she sampled from (the true population) was healthy before the timber harvest. Mature cherry trees should have a girth of around 12.5 inches. So, she'd like to use the sample of trees that she harvested to test if the mean girth of all the cherry trees in the stand is different than the expected girth.

Question 11 (0.5 point): Use the t.test function to test the null hypothesis that mean cherry tree girth from the stand is 12.5 inches. Interpret the output, making sure to explain what the P-value actually means, and what this indicates about the original question.

```
# your code here
```

Your answer here

On your own, run the `t.test` function to compute a P-value for the body temperature data. Did you get the same result as when you calculated it hand in Question 9? If you've done it correctly, those numbers should match.

Two-sample test

Lastly, to compare the means of two groups (a two-sample t-test), you must give the t.test two vectors to compare. The two sample sets do not need to be of equal length. The two-sample t-test tests the null hypothesis that there is *no* difference in mean between the two groups. Don't get two-sample confused with two-tailed!

```
myvalues <- c(44, 46, 29, 72, 90)
myvalues2 <- c(34, 32, 30, 31, 32, 30, 19, 45)
t.test(x = myvalues, y = myvalues2)
```

```
##
##  Welch Two Sample t-test
##
## data:  myvalues and myvalues2
## t = 2.1938, df = 4.4232, p-value = 0.0868
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -5.387915 54.537915
## sample estimates:
## mean of x mean of y
##    56.200    31.625
```

Some experiments used paired measurements; that is, they apply both treatments to every sampling unit. In these experimental designs, data are *paired.* For instance, a study might measure patients' blood pressure before and after receiving a medication. In this case, each patient is in both the "before" experimental group and the "after" experimental group, and the data are paired. To run a paired t-test, you add the argument `paired = TRUE` to the `t.test()` function.

```
before <- c(150, 144, 128, 159, 141, 132, 129)
after <- c(148, 142, 128, 165, 138, 128, 130)
t.test(x = before, y = after, paired = TRUE)
```

```
##
##  Paired t-test
```

```
##
## data:  before and after
## t = 0.45004, df = 6, p-value = 0.6685
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -2.535518  3.678375
## sample estimates:
## mean of the differences
##               0.5714286
```

Review/conceptual questions

Question 12 (1 point): What is the difference between bias and sampling error? Give an example of each.

Your answer here

Question 13 (1 point):. Alaska is the state with the lowest gender ratio of women to men. 48 percent of Alaskans are female. The freshman class at University of Alaska, Fairbanks has 623 males and 825 females. Is the proportion of males and females different at UAF relative to the rest of the state? Use one of the statistical tests we have learned so far. Report the P-value and test statistic, and interpret the results.

```
# your code here
```

your answer here