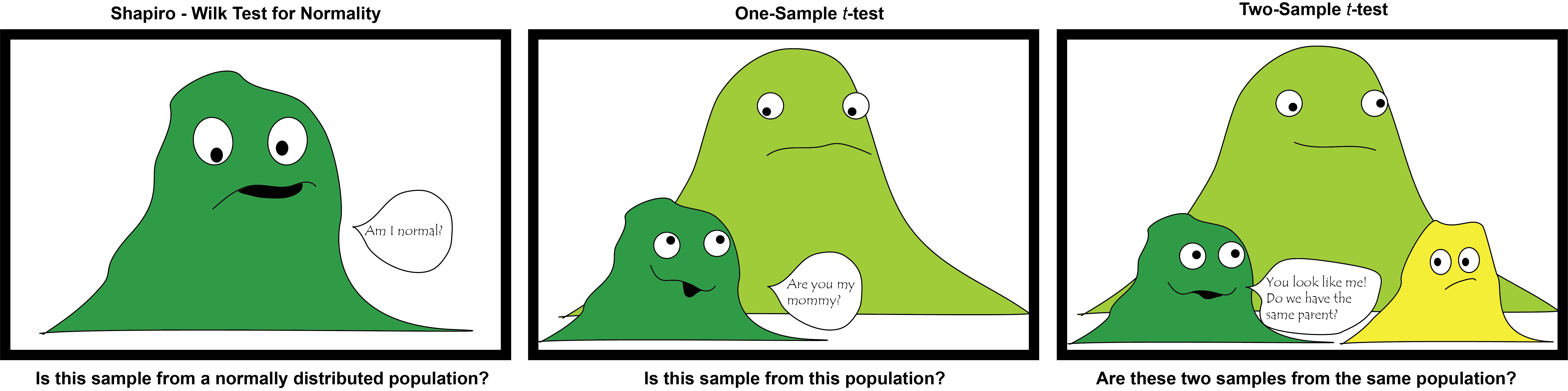
Lab 4 - Basic *t*-Tests

# Introduction

This lab is all about basic *t*-tests, which look to see if samples are different from expected populations (or each other). First you will use fake data to help you practice with the code, then you will use a real dataset on Kickstarter funds to see how the *t*-tests work in real life.

If you are still struggling with *p* values after this lab, make sure to reach out to your section leader or instructor for help! They are a big part of statistical tests.



Last week, we ran the tests of the first panel, today we will do the middle and third.

# Learning Outcomes

By the end of today’s lab you should be able to:

* Use the t.test() function to run a one-sample *t*-test with a specified mean
* Use the t.test() function to run a two-sample *t*-test
* Use the par(new=TRUE) function to overlap two plots
* Use the par(mfrow=c()) function to stack two plots
* Interpret *p*-values from a one or two-sample *t*-test
* Use the subset() command to filter numeric and character data
* Identify sources of uncertainty in financial data

# Part 1: Fake Data Practice

The idea of the *t*-test is that you can use it to tell if a sample belongs to a certain population, or if two samples came from the same population. So to start with, it is helpful to make fake data that meets certain criteria:

1. a sample that comes from a known population
2. a second sample that comes from a different population

You have already learned how to make fake data using the rnorm() function, and today you will do the same thing. In last week’s lab, it didn’t much matter what the average was because your question was *is this normally distributed.* Today, the means are important because they are what tells us if your fake sample comes from a particular population with a certain mean.

## Part 1.1 Make the Data

We want these samples to be different from each other, so we will make two samples from populations with different means - in this case, a mean of 15 (**dist.15**) and a mean of 25 (**dist.25**). I’m keeping the standard deviations the same and the samples the same; you can change them if you want to.

dist.15 <- rnorm(n = 50, mean = 15, sd = 2)  
dist.25 <- rnorm(n = 50, mean = 25, sd = 2)

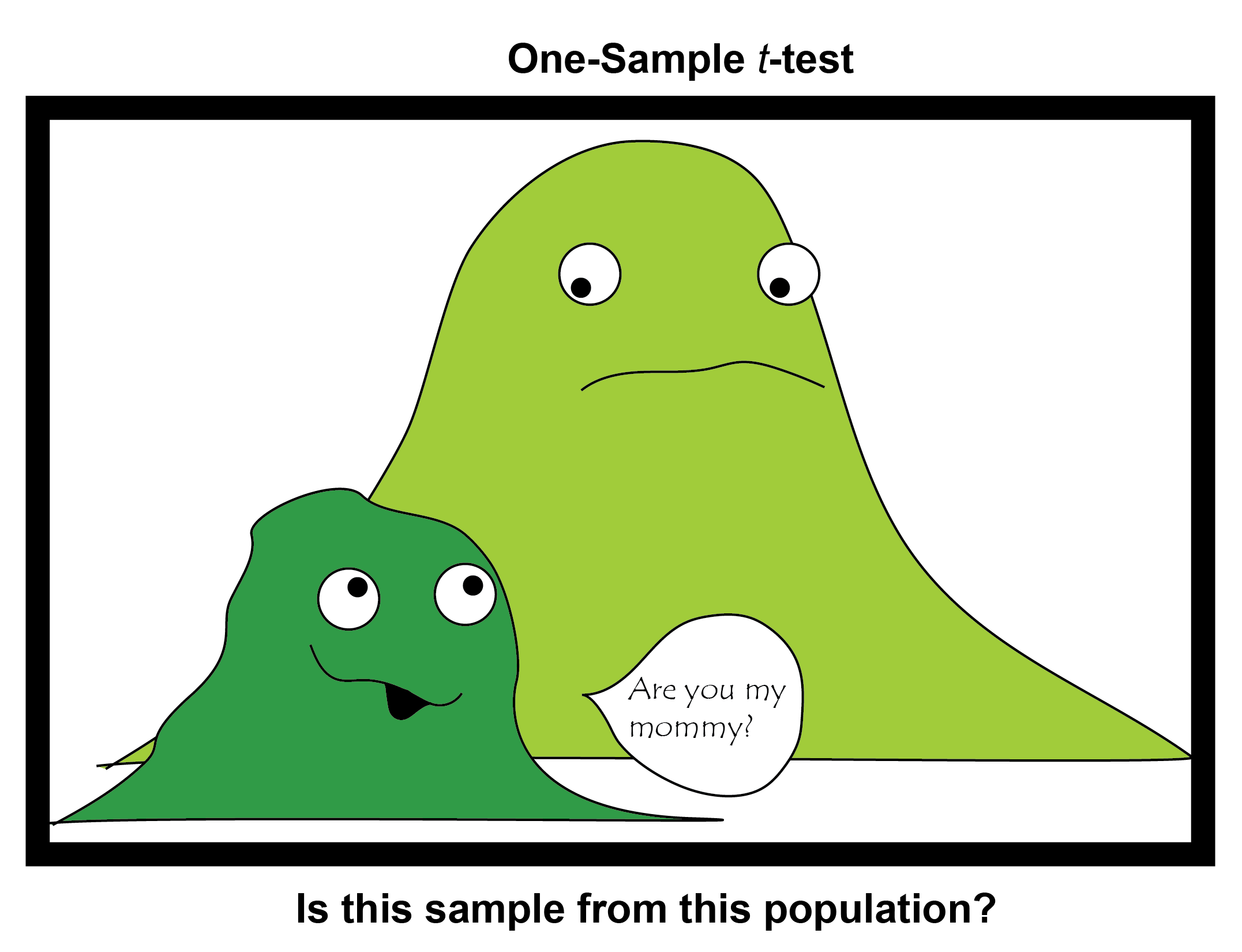
It is always a good idea to look at the data, but to look at two different histograms requires some new code. To overlap two plots on top of each other, use the par(new=TRUE) code in between them and run all three lines together, like so:

hist(dist.15, xlim=c(10,35), ylim=c(0, 15), col="red")  
par(new=TRUE)  
hist(dist.25, xlim=c(10,35), ylim=c(0, 15))

### QUESTION 1: Try removing the xlim and the ylim sections of code from the above plot. What happens with the two histograms?

1. They explode
2. Only one shows up
3. Their axes don’t line up
4. Their titles and labels overlap

## Part 1.2 One-Sample *t*-Test



There are lots, and lots, and LOTs of different types of *t*-tests. To start with, we’re going to look at a one-sample *t*-test. That means you have one sample of data, and you’re trying to find out if it came from a population with a particular mean.

To start with, let’s try the *alternate hypothesis* - that is, let’s run a *t*-test on a sample where we know the answer should be that the sample absolutely did not come from that population. We know the means of our two samples are 15 and 25, so let’s test the idea that one of them came from a population with a mean of 250 (unlikely!).

The simplest version of the *t*-test in R is just to use t.test() with your sample, and then specifing that the **mu** argument is the population mean you’re testing it against (250).

t.test(dist.15, mu=250) #mean of 15  
t.test(dist.25, mu=250) #mean of 25

If you run that code, it gives you a lot of data, more than the Shapiro-Wilk test from last week:

##   
## One Sample t-test  
##   
## data: dist.15  
## t = -727.59, df = 49, p-value < 2.2e-16  
## alternative hypothesis: true mean is not equal to 250  
## 95 percent confidence interval:  
## 13.80825 15.10936  
## sample estimates:  
## mean of x   
## 14.4588

The pieces of this output are:

1. Your data, named **dist.15**
2. the **t** value, or test statistic for this test
3. The **degrees of freedom**, which relates to sample size
4. The **p-value**, which is probability

There are some other pieces below that, but don’t worry about them for now. Sometimes R likes to give you way more information than you actually need, and this is one of those cases.

For the following questions, you should read through them before running any code. While the t.test() code will give you the answer, it is good practice to use what you already know to hypothesize about the answers to these questions. If you become good at that, then in the future if you run a test and something goes wrong, you’ll be more likely to spot it!

### QUESTION 2: If you run a *t*-test on dist.25 to determine if it came from a population with a mean of 250, you should get a *p*-value of less than or equal to 0.05 because:

1. There is a good probability it came from a population with a mean of 250
2. There is a very low probability it came from a population with a mean of 250

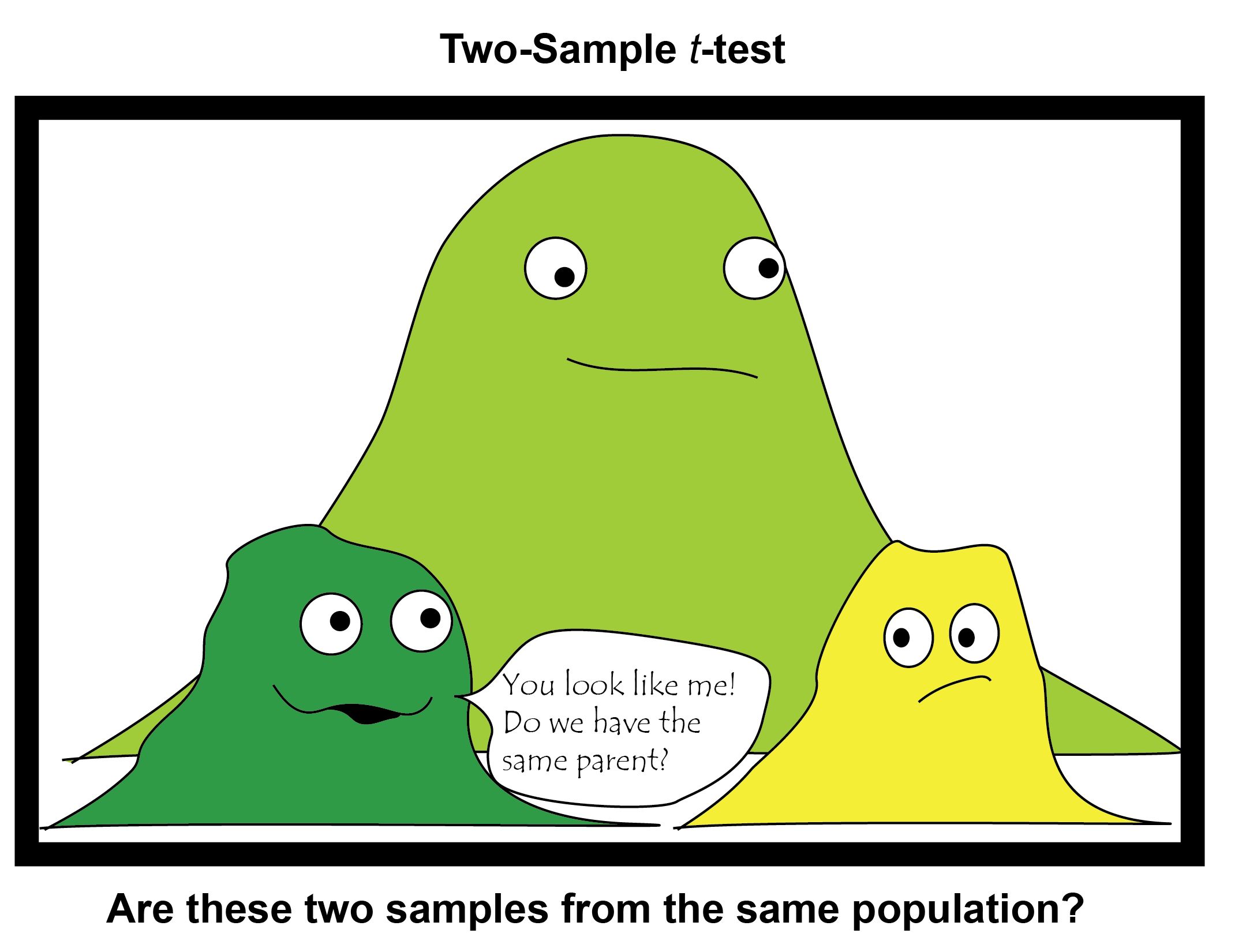
### QUESTION 3: If you run a *t*-test on dist.15 to determine if it came from a population with a mean of 15, you should get a *p*-value of greater than 0.05 because:

1. There is a good probability dist.15 came from a population with a mean of 15
2. There is a very low probability dist.15 came from a population with a mean of 15

### QUESTION 4: If you run a *t*-test on dist.25 to determine if it came from a population with a mean of 25, what *p*-value would you expect?

1. < 0.05 because there is a good probability dist.25 came from a population with a mean of 25
2. > 0.05 because there is a good probability dist.25 came from a population with a mean of 25
3. < 0.05 because it isn’t very likely that dist.25 came from a population with a mean of 25
4. > 0.05 because it isn’t very likely that dist.25 came from a population with a mean of 25

## Part 1.3 Two-Sample t-Tests



Sometimes you aren’t comparing a sample to a single mean value. Sometimes, you’re comparing two samples to each other to see if they came from the same population. This is known as a **two-sample *t*-test.** It is very similar, but instead of mu= you put in the other sample.

For example, let’s try our two made-up samples that we know come from different populations (one from a population with a mean of 15, and the other of 25).

t.test(dist.15, dist.25)

### QUESTION 5: A two sample t test of dist.15 versus dist.25 typically comes up with a *p*-value less than or equal to 0.05. This means:

1. There is a high probability that these two distributions came from the same overall population
2. There is a low probability that these two distributions came from the overall population
3. These two samples are the same

### QUESTION 6: If you run t.test(dist.15, dist.15), would you expect to get a high *p*-value or a low *p*-value?

1. High *p*-value, because these are the same samples so from the same population
2. Low *p*-value, because these samples probably came from different populations

# Part 2: One Sample *t*-tests on Real Data

Now that you’ve played around with two clearly different normal distributions and a variety of means, let’s take a look at some real data. Specifically today we will be looking at Kickstarter campaigns from 2016 and seeing how the monetary goal for these campaigns can impact their success.

## Part 2.1 Load and Clean Data

Use the read.csv() or **import dataset** button to pull in the *2016 Kickstarter Projects.csv* file you downloaded from D2L. I have called my imported dataset ks, you may call it what you like.

This dataset contains a lot of campaigns for essentially beer money - campaigns for a dollar or two. We’re not particularly interested in those, so we are going to remove them using the subset() function to filter our dataset.

Subset works off of a **filtering command**, or a condition that has to be met for the data to be included in the subset. In this case, our filter is that we want the dataset to only include kickstarter goals of more than 10 dollars.

ks1 <- subset(ks, goal > 10)

This function uses a **relational operator**, the > sign, which tells R to only take goals that are greater than 10.

### QUESTION 7: Which of the following filter codes would get you a subset of data with goals only 100 dollars or higher?

A. <- subset(ks, goal == 100)  
B. <- subset(ks, goal >= 100)  
C. <- subset(ks, goal <= 100)  
D. <- subset(ks, goal >> 100)

One common error in subsets is to make a filter that is too specific, where none of the data meets your criteria. For example, if I tried to run a code that said the kickstarter goal was 100 Goats I wouldn’t get any results as pledges are all in US currency, not small hooved mammals.

### QUESTION 8: Which of the following filter codes would get you a subset of data that has no observations?

A. <- subset(ks, goal >> 0)  
B. <- subset(ks, goal >= 0)  
C. <- subset(ks, goal != 0)  
D. <- subset(ks, goal <= 0)

You can also subset your data using text filters. So now that we have ks1 which has goals more appropriate to our questions, we also need to divide it up into *failed* and *successful* data sets using the == operator on the column state. Because these are characters (words), you must put them in quotation marks!

successful <- subset(ks1, state=="successful")  
failed <- subset(ks1, state == "failed")

## Part 2.2 One-Sample Example

Wired magazine (<https://www.wired.com/2012/07/kickstarter/>) suggests “some Kickstarter goals are so high, they’re laughable, and others are too low to be taken seriously. For the best odds of success set your Kickstarter goal near $10,000.”

So testing this assertion could be done with a one-sample *t*-test two different ways:

1. Are *successful* campaign goals significantly different from a population with a goal of $10,000?
2. Are *failed* campaign goals significantly different from a population with a goal of $10,000?

If the $10,000 dollar goal is correct, the answer to the first version should be no, and the answer to the second version of this question should be yes.

To answer the first question, use the successful dataframe and a mu of 10000 ($10,000). Remember that this is using a data frame, so use `$` to indicate the correct column!

### QUESTION 9: Did this sample of successful goals come from a population with a mean of $10,000?

1. No, p < 0.05 meaning there is a less than 5% probability that it came from a population with a $10,000 mean.
2. Yes, p < 0.05 meaning there is a less than 5% probability that it came from a population without a $10,000 mean.
3. No, p > 0.05 meaning there is high probability that it came from a population with a $10,000 mean.

### QUESTION 10: Use the mean() function to find out the mean of the successful sample. Why do you think the Wired campaign suggested a mean of $10,000?

## Part 2.3 One sample *t*-test On Your Own

Now run the second question on your own: does the failed campaign category come from a population that doesn’t have a $10,000 mean?

### QUESTION 11: Is the average failed campaign goal from 2016 significantly different from a goal of $10,000? Support your answer with your *p*-value and the mean of the failed campaign goals.

### QUESTION 12: While Wired suggests that a 10,000 goal is most likely, other webpages suggest that a campaign of 10,000 *or less* is most likely - not a population with a mean of $10,000. Do your results suggest that this is true? Why or why not?

# Part 3: Two-Sample t-Test With Real Data

Another question would be to look at successful versus failed campaigns to see if their goals come from the same overall population (that is, does goal amount differ between successful and failed campaigns). This requires doing a two sample *t* test, but first it is always good to look at your data!

## Part 3.1 Visuals

These two datasets are so very different from each other it can be hard to get the histograms to overlap in a way that makes sense. Use the code below (run all together) to look at the data:

options(scipen=999)  
par(mfrow=c(2,1))  
hist(successful$goal, xlim = c(0, 1000000), ylim= c(0, 5000), breaks=1000)  
hist(failed$goal, xlim = c(0, 1000000), ylim= c(0, 5000), breaks=1000, col = "red")

### QUESTION 13: What do you think the par(mfrow=c(2,1)) code does? To find out, try changing the numbers in the c() function.

1. Makes angry cats appear in your plot
2. Plays golf in the background
3. Turns the background white
4. Places the plots in rows or columns

## Part 3.2 Two-sample *t*-tests

Now that you’ve looked at them, you’ll notice they look pretty different. But it’s always good to actually test that assumption! Use the t.test() formula to compare the successful and the failed goal datasets. Modify the code from part 1.3 with the correct datasets.

### QUESTION 14: Do these two datasets come from the same population?

1. Probably not - there is a less than 1% probability they come from the same population because the failed goals are much larger.
2. Probably not - there is a less than 1% probability they come from the same population because the failed goals are much smaller.
3. Probably - the average goals are in the same ballpark between both datasets.

# Part 4 Analyses

Before you finish, these questions are intended to help you think a little bit more about the data and the analyses.

### QUESTION 15: If someone analyzed this data and provided an alternate average of saying that most successful campaigns were about 5000 dollars, what code would you run to test that?

A. t.test(successful$goal, failed$goal, mu = 5000)  
B. t.test(failed$goal, mu = 5000)  
C. t.test(successful$goal, mu = 5000)

### QUESTION 16: One of the goals was to get rid of Kickstarter campaigns that weren’t actually real, but were jokes. What else could you filter out to get rid of fake campaigns from this dataset? Be specific (and talk about this data, not how you could collect better data).