Advanced Statistics

author: Bernhard Angele date: Class 2, October 9, 2014

Some general advice on working with R

- Write all important steps into a script file or into your Rmd file so that you have a record.
- That is the number 1 secret for working with R.
- It generalises to SPSS: always save the code you used so you can replicate your analyses.
- This will save you an incredible amount of work if you have to go back and double-check what you did.
- You can send any line in the script file to the console by pressing Ctrl + Enter.
- Use the console only for things you don't want to save, like printing a variable to see what it is, trying out a new command, etc.

How to print your reports

- Make sure the file name ends in `.Rmd`
- Click on the little arrow next to the Knit button
- Select "Knit Word"
- Open the resulting file in Word and print it. Easy!

Commenting

- In your scripts, make use of the `# comment symbol` to write little notes to yourself about what you were thinking.
- Invaluable when you go back years later
- Also, if you put these in your code for the assignments, it will help me understand what you were thinking, too!
- XKCD on comments

Recap

- Last week, we talked a lot about sampling from different probability distribution.
- We also talked about the properties of the distribution of sample means (hint: it's always roughly normal).
- Now what can we do with this knowledge? Remember the convenience function we wrote last time:

```
run_simulation <- function(sample_size = 100,
                           number_of_simulations = 1000,
                           population_mean = 0,
                           population_sd = 1)
  {

sample_means <- replicate(number_of_simulations,
                          mean(rnorm(n = sample_size,
                                     mean = population_mean,
```

```
                                        sd = population_sd)))
}
```

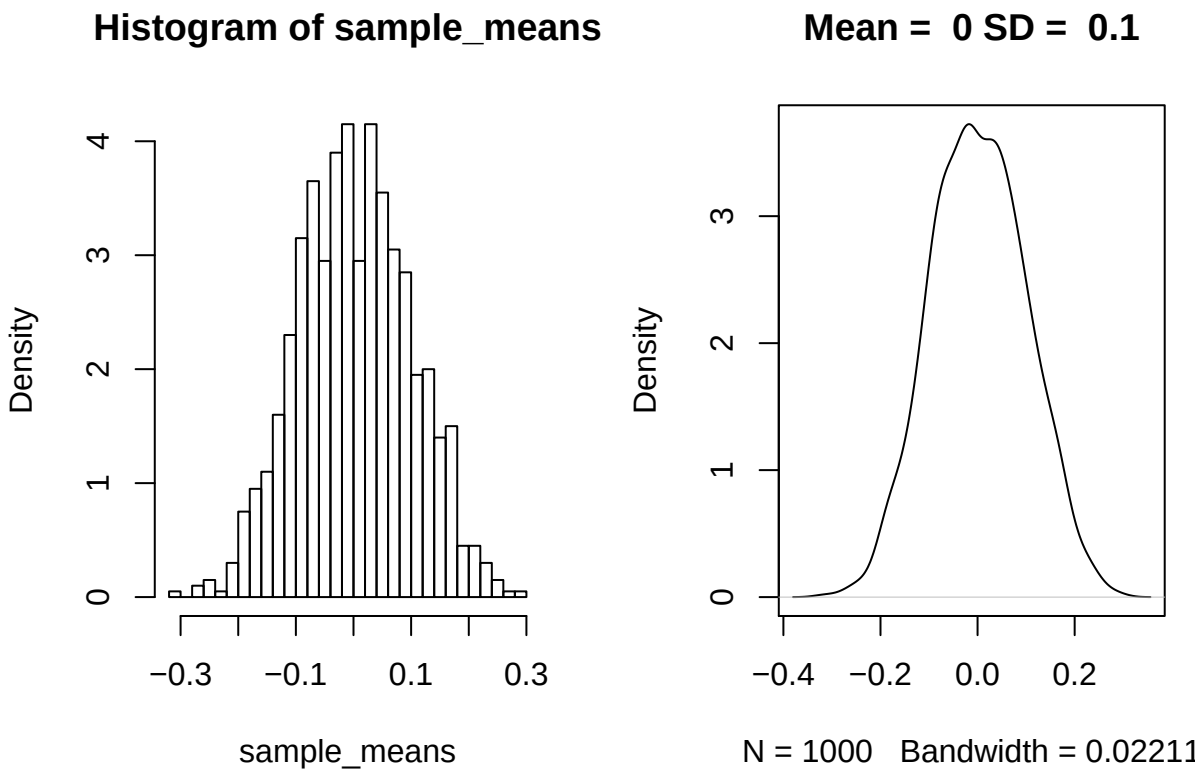
Recap (2)

- Remember our function for plotting these distributions:

```
make_hist_and_plot <- function(sample_means){
  par(mfrow=c(1,2)) #
  hist(sample_means,freq=F, breaks = 30)
  plot(density(sample_means),
       main = paste("Mean = ", round(mean(sample_means),2) ,
                    "SD = ", round(sd(sample_means),2)))}
```

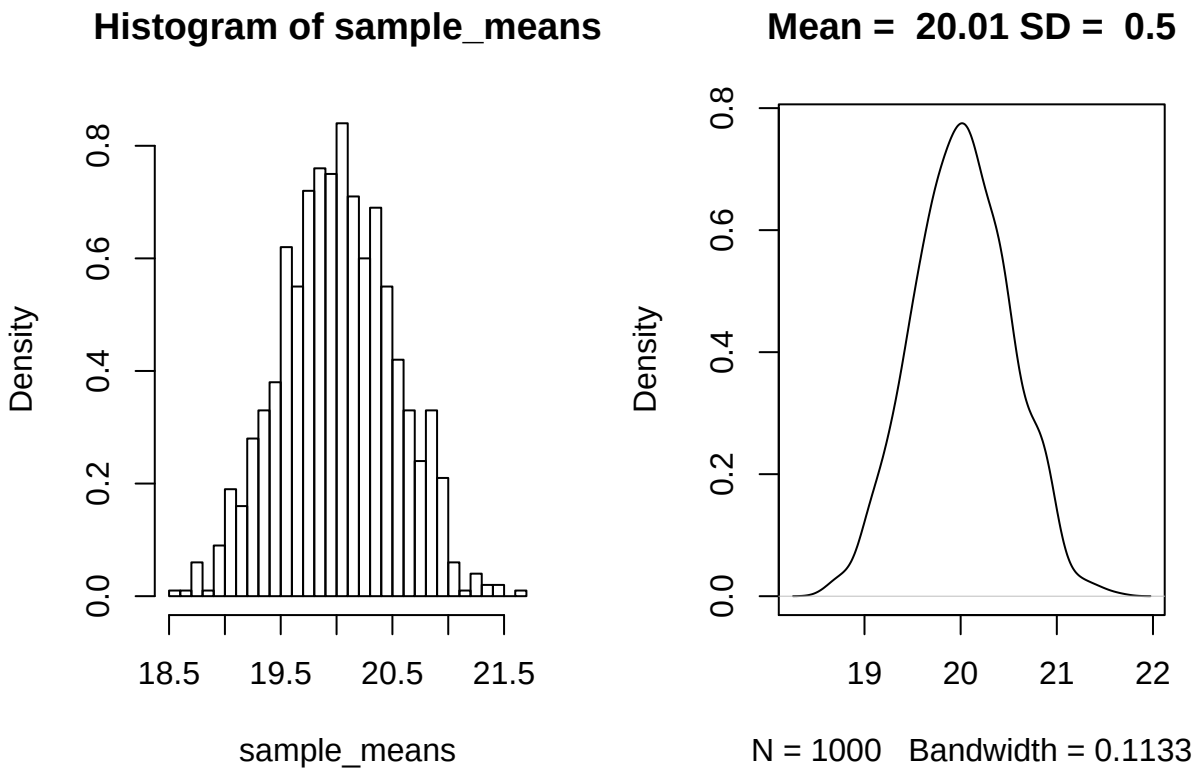
Recap (3)

```
make_hist_and_plot(
  run_simulation(100,1000,0,1))
```



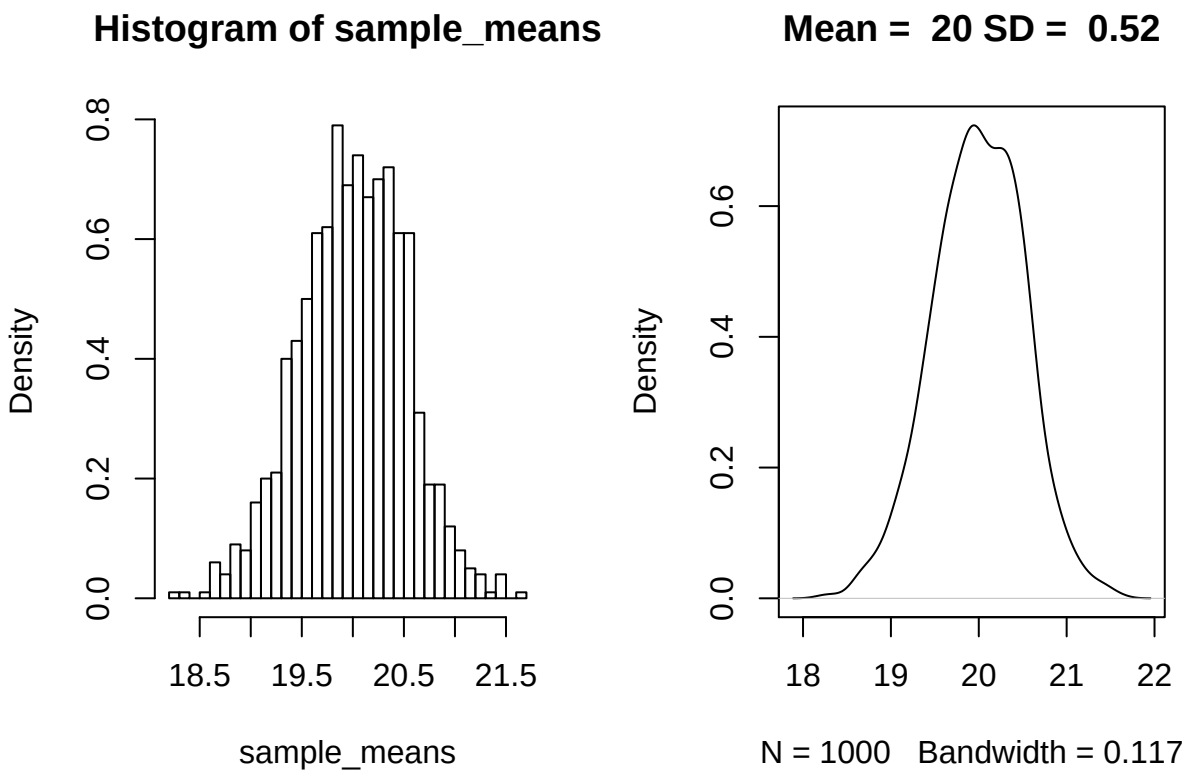
What changes when we re-run the simulation?

```
make_hist_and_plot(
  run_simulation(100,1000,20,5))
```



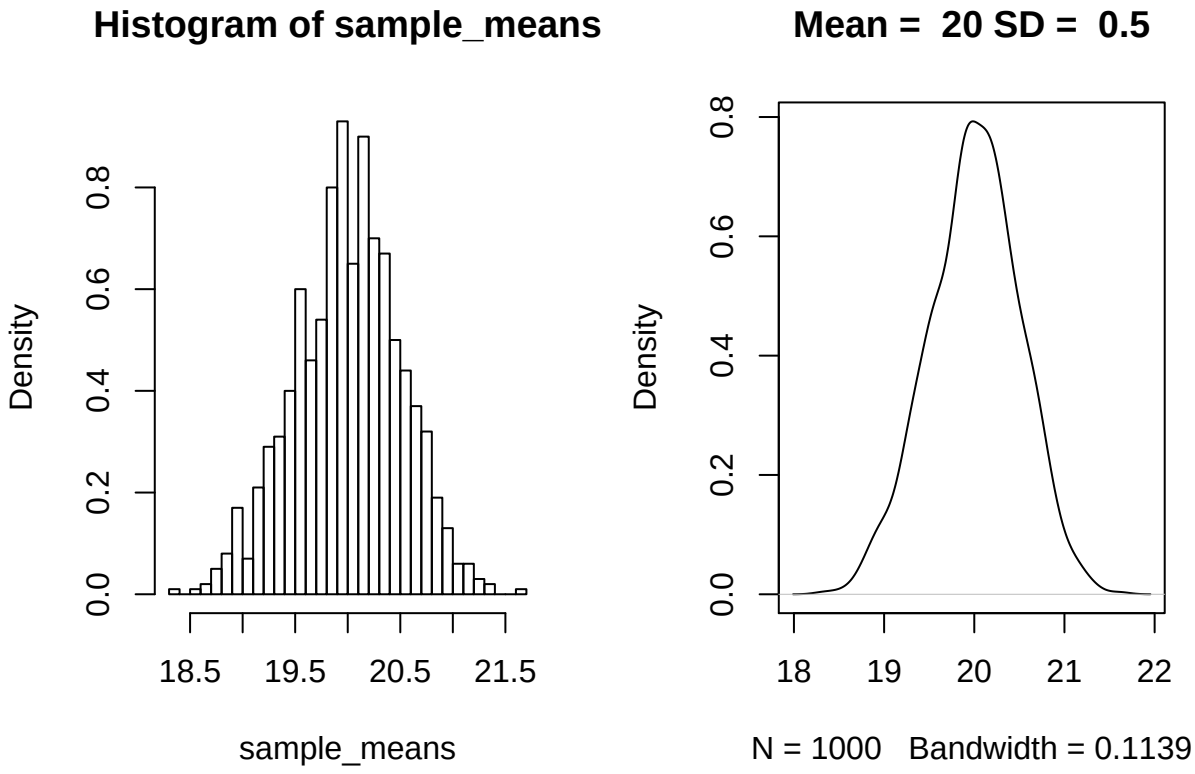
What changes when we re-run the simulation?

```
make_hist_and_plot(
  run_simulation(100,1000,20,5))
```



What changes when we re-run the simulation?

```
make_hist_and_plot(
  run_simulation(100,1000,20,5))
```



What changes when we re-run the simulation?

It turns out the mean of the distribution of sample means varies around the population mean. The sd also varies, but a lot less. It varies around

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

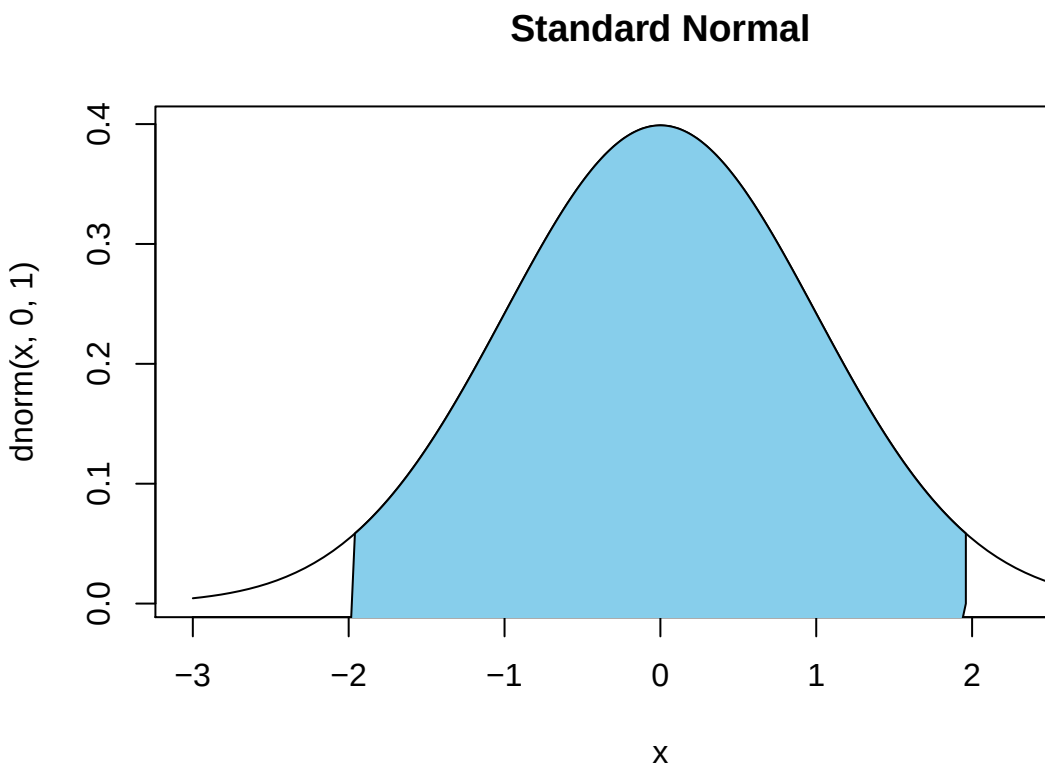
So, to sum up: The distribution of sample means is (roughly) normal, with $\mu_{\bar{x}} = \mu$ and $\sigma_{\bar{x}} = \frac{\sigma}{\sqrt{n}}$. This means we can apply some of our knowledge about the normal distribution!

Confidence intervals

- If the distribution of sample means is normal, that means we can say something about the relationship between sample mean and population mean.
- Let's say the population mean μ is 0 and the population sd σ is 1.
- What is the sample mean going to be?
- Think: what is the answer to this going to look like?
- $\mu_{\bar{x}}$ is a random variable, so it doesn't make sense to give a point estimate
- Instead, we can give an interval.
- Do you remember the function for that?
  - That's right, it's `qnorm`.

Confidence intervals (2)

- So, let's get the interval that $\mu_{\bar{x}}$ is going to be in 95% of the time.



- We want something like this:

Confidence intervals (3)

- Let's start with the standard normal distribution (**z-scores**)
- We want to get an interval that includes 95% of the area under the curve
- That means we need to take off 2.5% on every side
- For the left interval boundary, we want the x value that is greater than or equal to 2.5% of x values

```
qnorm(.025)
```

```
## [1] -1.96
```

Confidence intervals (4)

- For the right interval boundary, we want the x value that is greater than or equal to 97.5% of x values.
- Get the corresponding **z-score**:

```
qnorm(.975)
```

```
## [1] 1.96
```

- If you've done statistics before, these numbers should be pretty familiar to you.
- Generalising this to other normal distributions is easy: $\bar{x} = \mu \pm 1.96 \times \sigma_{\bar{x}}$
- Replacing $\sigma_{\bar{x}}$ with the expression based on the population SD: $\bar{x} = \mu \pm 1.96 \times \frac{\sigma}{\sqrt{n}}$

Exercise

- It is (for some reason) well-known that the amount of cat food a cat needs per day is normally distributed with a mean of 2 cans per day and an sd of .5. I'm planning to adopt two (completely random) cats and need to plan this move financially.
- What's the maximum and the minimum amount of cat food cans I must expect to buy every day?
- This estimate should be fairly accurate and should only have a 10% chance of being wrong.
- Suppose I don't care about the minimum amount, I just want to know the maximum – does that change anything?
- Suppose I'm adopting 3 cats instead of 2 – does that change anything about my estimate?
- Hint: The function you want to use starts with a `q`.

Solution

- I'm drawing a random sample of hungry cats (sample size 2) from the population of hungry cats
- How hungry? Mean = 2 cans/day, sd = .05 cans/day
- I want a 90% CI for the mean of that sample
- Get the z-scores for the lower and the upper bound:

```
qnorm(.05)
```

```
## [1] -1.645
```

```
qnorm(.95)
```

```
## [1] 1.645
```

Solution (2)

- Calculate the CI:

```
2 + qnorm(.05) * .5/sqrt(2) # lower limit
```

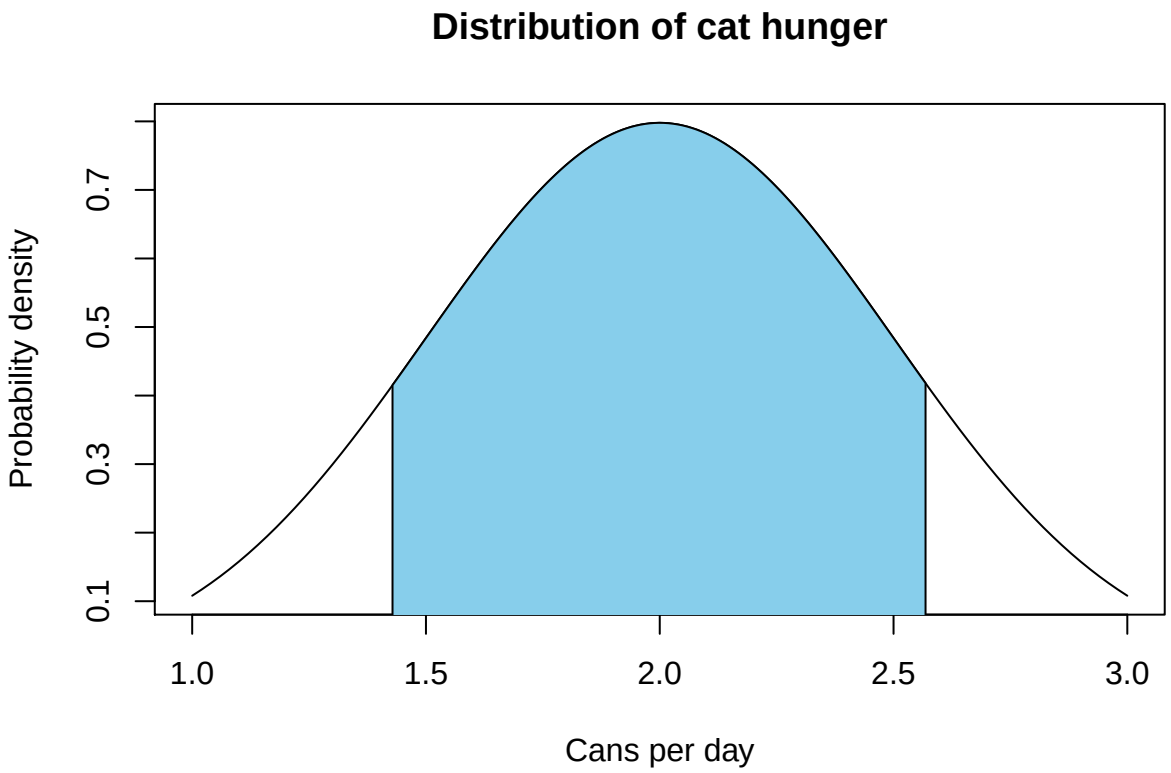
```
## [1] 1.418
```

```
2 + qnorm(.95) * .5/sqrt(2) # upper limit
```

```
## [1] 2.582
```

- Those are some hungry cats!
- I need to plan on buying between 1.4185 and 2.5815 cans of cat food per day (per cat).

Plot it!



Solution (4)

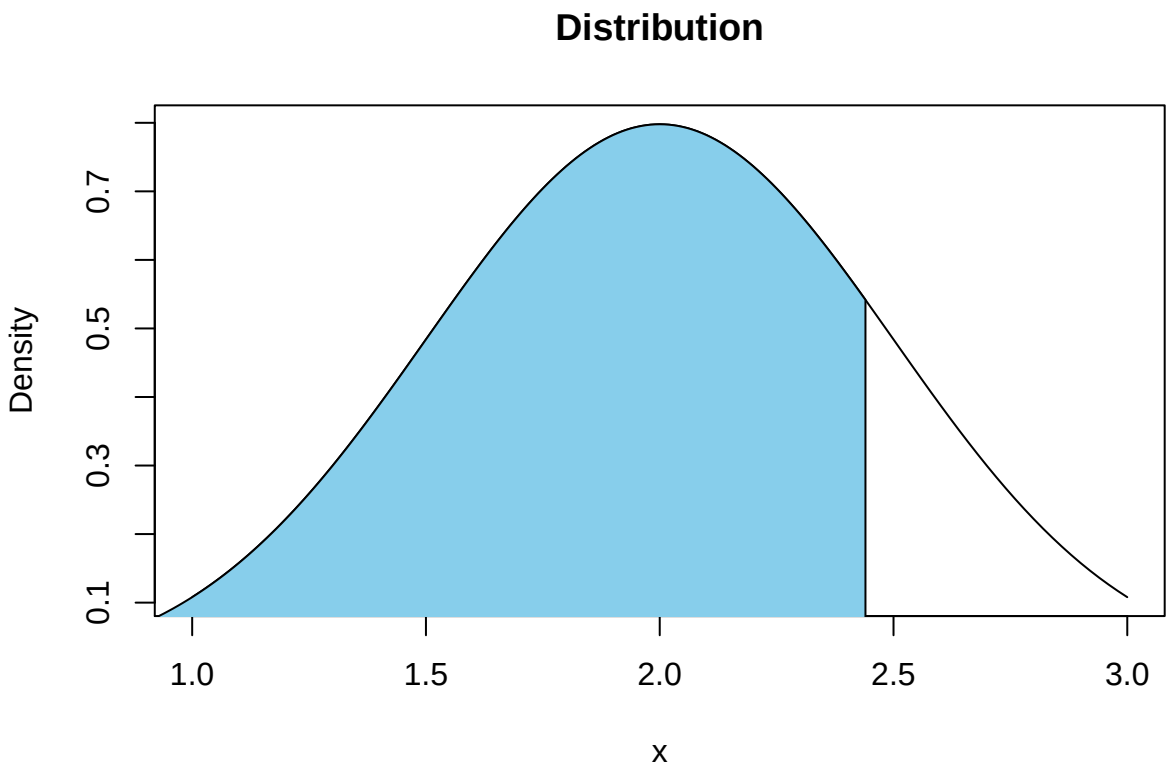
- If I only care about the maximum, I don't need the lower limit.
- I can use a different upper limit to get an interval that delimits 90% of the area under the curve.

```
2 + qnorm(.90) * .5/sqrt(2) # upper limit
```

```
## [1] 2.453
```

- Those are still some hungry cats!
- I need to plan on buying at most 2.4531 cans of cat food per day (per cat).

Plot it again!



Solution (5)

- What if I'm getting 3 cats?

```
2 + qnorm(.90) * .5/sqrt(3) # upper limit
```

```
## [1] 2.37
```

- Why is it less?
- The chances of getting 3 out of 3 very hungry cats are lower than the chances of getting 2 out of 2 very hungry cats.

Now reverse the idea

- Usually, we have no other information about a population but the sample we just collected.
- For example, let's say the sample mean is 0 and the sample SD is 1. Apart from this, we know nothing about the population.
- Can we compute a CI for the sample mean?
- Sure enough we can, but it gets a little more complicated.
- (who would have thought?)

What do these numbers mean?

- Anything, really.
- But let's imagine that these numbers are from a survey of student's attitudes towards their Advanced Statistics class.
- Imagine that they could give a rating from -3 ("This is the worst class ever and I want the instructor fired!") to 3 ("This is the best class I've ever taken! I'm going to make so much money with my new R skills!"), with 0 representing a neutral feeling ("It's alright. At least it will be over soon").
- In this case, most students are pretty neutral about the class, but some really love it and some really hate it.

Computing a CI from the sample mean (story time)

- Consider the following scenario:

  I have collected 10 responses to my class evaluation (the other students never turned their forms back in). The mean of the responses is 0 (apathy) and the sd is 1. Given that these 10 responses are just a small sample of the population, and that the population I'm really interested in is the population of all current and future Adv Stats students, is there anything I can say about the true population mean? Can I at least conclude that students didn't absolutely hate this class?

Computing a CI from the sample mean

- We'll have to estimate both the population mean and the population variance.
- We have already estabished that the sample mean is a good estimator for the population mean.
- What about the sample sd (s)? Is it a good estimator for the population sd (σ)?
- Or the equivalent question: is sample variance (s^2) a good estimator of population variance (σ^2)?
- This sounds like really tricky maths problem.
- But we can take it easy and just simulate!

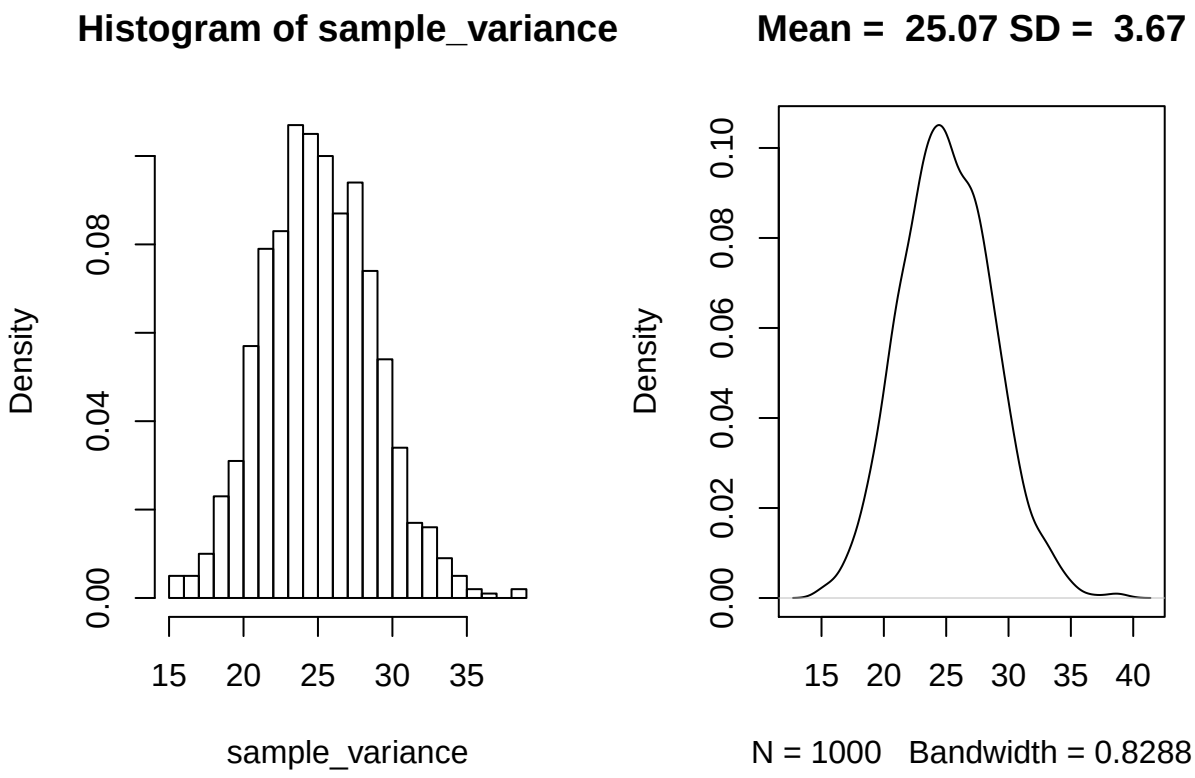
Set up a function to simulate sampling and calculate sample variances

```
run_variance_simulation <- function(sample_size = 100,
                                    number_of_simulations = 1000,
                                    population_mean = 0,
                                    population_sd = 1)
  {

sample_variances <- replicate(number_of_simulations,
                              var(rnorm(n = sample_size,
                                        mean = population_mean,
                                        sd = population_sd)))
}

#Define a new plot function so that the plot titles are correct
```

```r
make_variance_hist_and_plot <- function(sample_variance){
  par(mfrow=c(1,2)) #
  hist(sample_variance,freq=F, breaks = 30)
  plot(density(sample_variance),
       main = paste("Mean = ", round(mean(sample_variance),2) ,
                    "SD = ", round(sd(sample_variance),2)))}
```

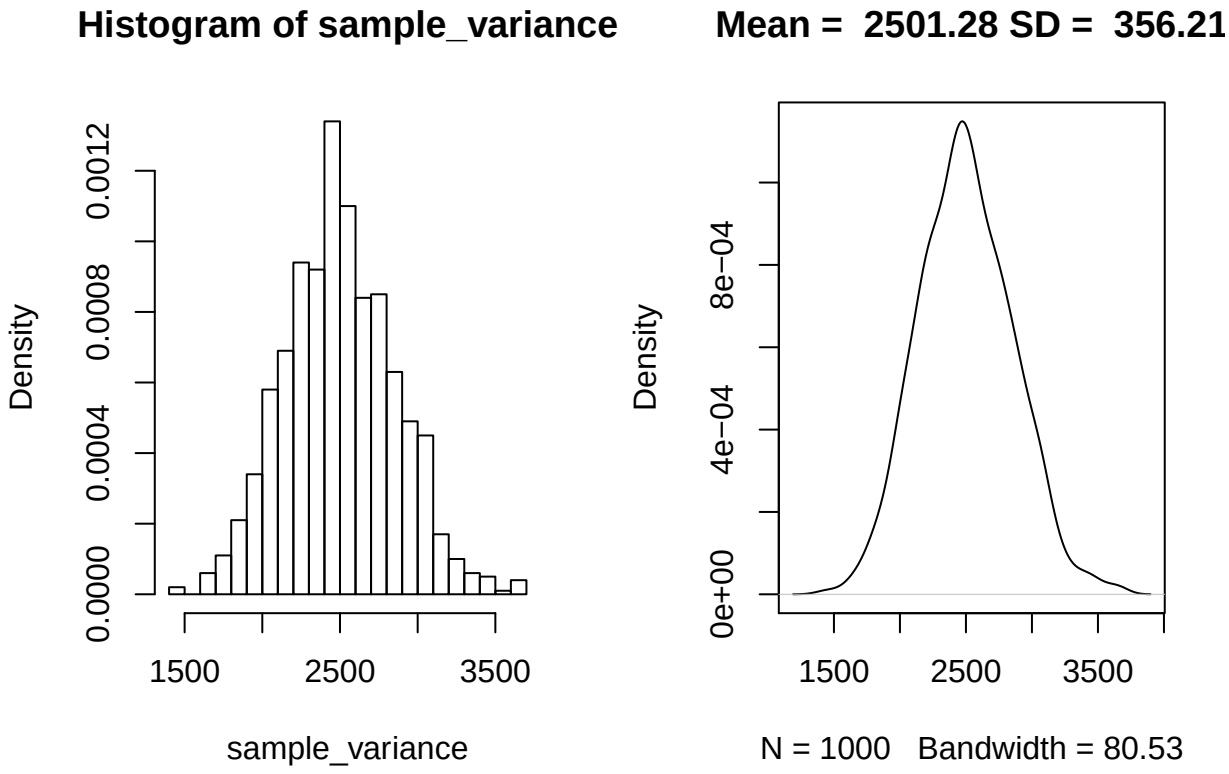
Population variance and sample variance: plots

```r
make_variance_hist_and_plot(
  run_variance_simulation(sample_size = 100,
                          number_of_simulations = 1000,
                          population_mean = 20,
                          population_sd = 5))
```



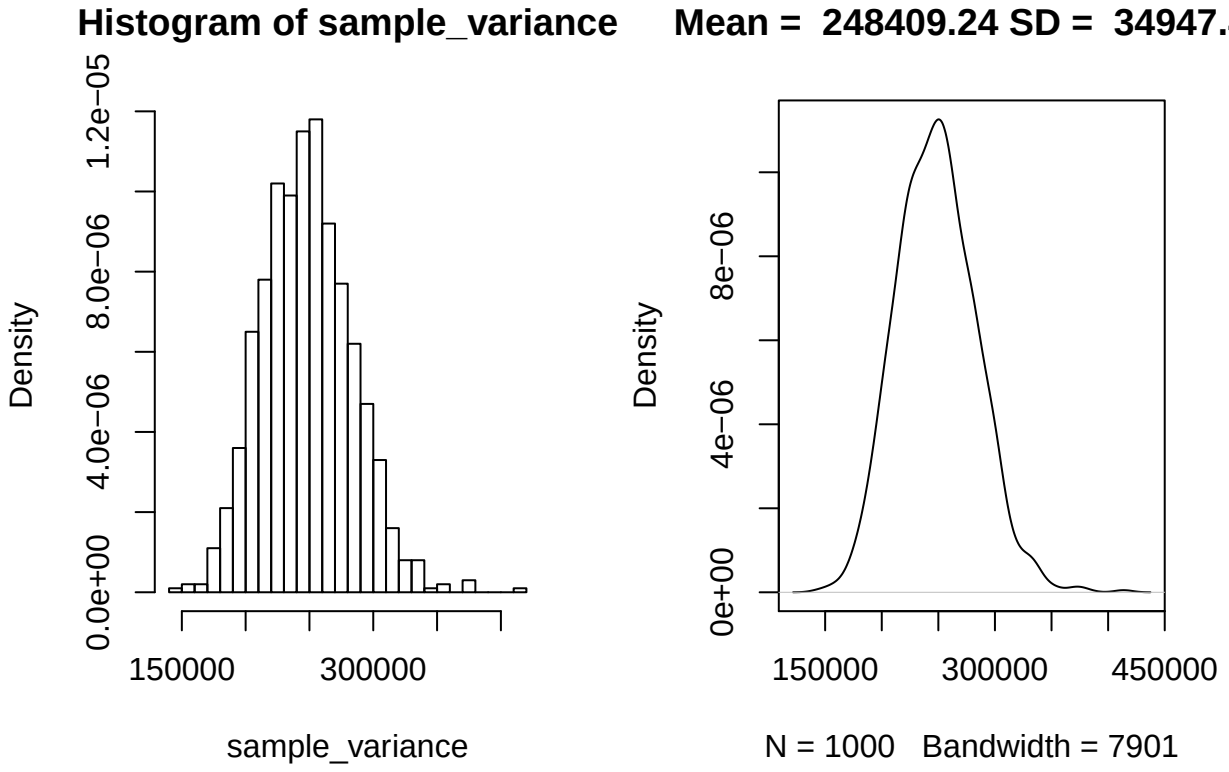
Population variance and sample variance: plots

```r
make_variance_hist_and_plot(
  run_variance_simulation(
    sample_size = 100,
    number_of_simulations = 1000,
    population_mean = 20,
    population_sd = 50))
```



Population variance and sample variance: plots

```
make_variance_hist_and_plot(
  run_variance_simulation(
    sample_size = 100,
    number_of_simulations = 1000,
    population_mean = 20,
    population_sd = 500))
```



Sample variance as an estimator of population variance

- Looks like it's a pretty good estimator (unbiased actually)
- We can plug the sd of the sample into the equation for the SD of the sampling distribution (or rather, the standard error):

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

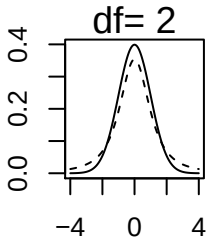
(Note that we are ignoring the question if the relationship between s and s^2 is really the same as the relationship between σ and σ^2. Feel free to simulate that, if you are really curious.)

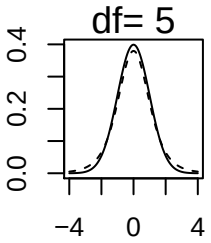
- But this means that our SE is an estimate of an estimate (estimating σ from s, then estimating $SE_{\bar{x}}$).
- This means that our estimate for σ is going to vary. Its accuracy will depend on the sample size.

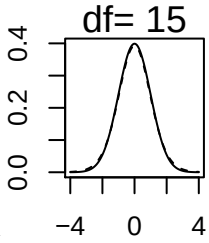
Dealing with the uncertainty in s

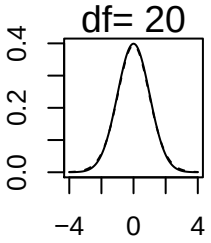
- We need a way to account for s being less accurate at low sample sizes.
- Solution: assume that the sample means aren't normally distributed, but rather t-distributed
- Why t?
- The t-distribution is like the standard normal distribution, but it has an additional parameter that we call df (for degrees of freedom, but don't worry about the name yet).
- The higher df, the closer the t-distribution is to the standard normal distribution
- For lower df, the t-distribution has "heavy tails", meaning that it's wider
  - This reflects greater uncertainty.

See for yourselves









Solid = normal distribution, dashed = t-distribution

Let's try this

- Using the t-distribution can compute CIs from samples as follows: get the lower and upper bounds (depending on sample size, e.g. 10):

```
  n <- 10
sample_means <- rnorm(n, mean = 0, sd = 1)
  qt(.025, df = n - 1)
```

```
## [1] -2.262
```

```
  qt(.975, df = n - 1)
```

```
## [1] 2.262
```

Why $n - 1$? Unless you really love statistics and want to find out more, don't worry about it.

Computing CIs

- Then take the upper and lower bounds and compute the CIs as follows: $\bar{x} = \mu_{\bar{x}} \pm 2.262 \times \frac{s}{\sqrt{n}}$
- Remember we estimated $\mu_{\bar{x}}$ using the sample mean

Computing CIs (2)

```
  (sample_mean <- mean(sample_means))
```

```
## [1] -0.3791
```

```
  (sample_sd <- sd(sample_means))
```

```
## [1] 0.6478
```

```
  (lower_bound <- sample_mean + qt(.025, df = n - 1)*(sample_sd/sqrt(n)))
```

```
## [1] -0.8425
```

```
  (upper_bound <- sample_mean + qt(.975, df = n - 1)*(sample_sd/sqrt(n)))
```

```
## [1] 0.08425
```

Back to our example

- Hey, there's a 95% chance that my current and future students don't hate me (yet)!
- The lowest mean in the CI is -0.8425, which maybe translates to "apathetic but slightly worried."
- But they don't love me either:
- The highest mean in the CI is 0.0843, which maybe translates to "apathetic but slightly hopeful."
- Of course, there is a 5% chance that the true mean is actually outside this interval.

There's a function for that

```
t.test(sample_means)
```

```
##
##  One Sample t-test
##
## data:  sample_means
## t = -1.851, df = 9, p-value = 0.09722
## alternative hypothesis: true mean is not equal to 0
## 95 percent confidence interval:
##  -0.84253  0.08425
## sample estimates:
## mean of x
##   -0.3791
```

How convenient is that?

What does the CI of the sample mean mean? (sorry)

- Remember, we are reversing the idea that the sample mean has a 95% probability to be within the 95% confidence interval around the population mean.
- When we calculate a 95% CI from a *sample* this **DOES NOT MEAN** that there is a 95% probability that the population mean is within this 95% CI.
- The true mean either is or is not in this particular CI.
- Rather, it means that if you take a lot of samples and compute the CI around the sample mean, 95% of those CIs will contain the true population mean.
- In other words, the CI bounds are random variables, but the population mean isn't.
- (In Bayesian statistics, you can actually get something equivalent to the first definition – a 95% credible interval.)

Let's test this

- Let's get 10 samples from a normal distribution, then get CIs from them and see how often they contain the true mean.
- Write a function for that:

```
test_cis <- function(n, mean = 60, sd = 4){
  t_results <- t.test(rnorm(n, mean, sd))
  # test that the mean is within the bounds of the CI. Returns TRUE or FALSE.
  mean > t_results$conf.int[1] & mean < t_results$conf.int[2]
}
```

Let's test this (2)

- Now run the tests:

```
mean_in_ci <- replicate(1000, test_cis(n = 10, mean = 60, sd = 4))
table(mean_in_ci)
```

```
## mean_in_ci
## FALSE  TRUE
##    43   957
```

- It's true! Almost exactly 5%

What happens if we don't use the t distribution?

```
test_cis_norm <- function(n, mean = 60, sd = 4){
  samples <- rnorm(n, mean, sd)
  upper <- mean(samples) + 1.96*(sd(samples)/sqrt(n))
  lower <- mean(samples) - 1.96*(sd(samples)/sqrt(n))
  # test that the mean is within the bounds of the CI. Returns TRUE if it is and FALSE if it isn't.
  mean > lower & mean < upper
}
```

What happens if we don't use the t distribution? (2)

```
mean_in_ci <- replicate(1000, test_cis_norm(10, 60, 4))
table(mean_in_ci)
```

```
## mean_in_ci
## FALSE  TRUE
##    91   909
```

- The proportion of CIs that did not contain the true mean is larger than 5%! This is because the normal distribution is narrower than the *t*-distribution at low dfs.
- Be **very** careful! If you *think* you have a 95% CI, but you actually have a 90% CI or worse, you are prone to making errors in interpreting the results.
- Horrible, money-wasting, science-distorting, extermely expensive errors!

What if we use a larger sample size?

```
mean_in_ci <- replicate(1000, test_cis_norm(100, 60, 4))
table(mean_in_ci)
```

```
## mean_in_ci
## FALSE  TRUE
##    57   943
```

- Back at 5%! For large sample sizes it's fine to use the normal distribution instead of the t-distribution (of course, the t-distribution works anyway).
- Could you have come up with this? You didn't have to thanks to the work William Sealy Gosset did back in 1908.

Random variables

- One of the points that I was trying to make above was that the true population mean and sd are not random. They have an actual point value (at any given point in time), and we could calculate that value if we were to measure every individual in the population.
- From the point of view of any individual, their value on whatever measurement we're taking is of course not random either.
- But, if we grab individuals at random, we will notice that their measure values are not completely arbitrary.
- Instead, depending on the probability density function, we will get some values more often than others.
- That's all we mean by saying that something is a random variable.

Random variables (bear with me)

Can we follow Shravan here? Warning: Some mathematical notation follows.

- A random variable X is a function $X : S \to \mathbb{R}$ that associates to each outcome $\omega \in S$ exactly one number $X(\omega) = x$.
- Note: $\mathbb{R}$ = real numbers
- S_X is all the x's (all the possible values of X, the support of X). i.e., $x \in S_X$.
- Good example: number of coin tosses till you get Heads (H) for the first time
- $X : \omega \to x$
  - ω: H, TH, TTH,... (infinite)
  - $x = 0, 1, 2, \ldots; x \in S_X$

Random variables (2)

Every discrete random variable X has associated with it a **probability mass/density function (PDF)**, also called *distribution function*.

$$p_X : S_X \to [0, 1]$$

defined by

$$p_X(x) = P(X(\omega) = x), x \in S_X$$

- Back to the example: number of coin tosses till H

- $X : \omega \to x$
- ω: H, TH, TTH,... (infinite)
  - $x = 0, 1, 2, \ldots; x \in S_X$
- $p_X = .5, .25, .125, \ldots$

Hypothesis tests

- In a way, by calculating the CI we already have a way to test hypotheses
- Let's say we got a 95% CI from our sample with a lower bound of 2 and an upper bound of 3.
- Let's use the simplest null hypothesis possible
- Null hypothesis: the mean of the population that the sample came from is 0
- $H_0 : \mu = 0$
- Given the 95% CI above, can we reject the null hypothesis?
  - And if so, what is the chance that we're wrong?
- Answer: Yes, we can, since 0 is not part of the CI.
  - There is the possibility that we are wrong, though, since only 95% of the CIs will contain the true population mean.
  - This is called the type I error, and its probability here (called α)is 5%.

Example

- Remember my survey? The CI did not contain -3, so I can conclude (with an α of 5%), that the average member of the population of current and future Adv Stats students attitude towards me is not intense hatred. Relief!

Hypothesis tests (2)

Let's look at the `t.test` output again.

```
t.test(sample_means)
```

```
##
##  One Sample t-test
##
## data:  sample_means
## t = -1.851, df = 9, p-value = 0.09722
## alternative hypothesis: true mean is not equal to 0
## 95 percent confidence interval:
##  -0.84253  0.08425
## sample estimates:
## mean of x
##   -0.3791
```

Two-tailed t-tests

- Instead of computing the CI from the t-value, we can also just take the t-value itself as a measure of how far the sample mean is away from the mean specified in the null hypothesis.
- We can determine a critical t-value t_{crit} depending on our α criterion and the df. For example, for a df of 9, t_{crit} for the upper bound is

```
qt(.975, df = 9)
```

```
## [1] 2.262
```

and t_{crit} for the lower bound is

```
qt(.025, df = 9) # note that the t-distribution is symmetrical
```

```
## [1] -2.262
```

In short, if $t \geq |t_{crit}|$, we can reject the null hypothesis.

What about one-tailed t-tests?

- If we are absolutely sure of the direction of the effect, then we could use a t-test that only rejects the null hypothesis when the t-value is greater than t_{crit} or if it is smaller than t_{crit} (depending on what direction we want to test for).
- In this case, our t_{crit} can be a little closer to 0, since the entire 5% rejection area is in one tail only.

```
qt(.95, df = 9)
```

```
## [1] 1.833
```

- But be careful, if the effect is in the wrong direction (even if it's ridiculously strong in the wrong direction), we can't reject the null hypothesis with that test.
- This is one of the weird cases in null hypothesis significance testing (NHST) where our intentions can determine the results of the test. Bayesian statisticians are right to complain about this.

Example

> I'm trying a new type of medication to help insomniac patients sleep better. Each of my 5 patients reports how much longer (or shorter) they have been sleeping (in hours) after taking the medication compared to before. The numbers are below. Based on this, can I conclude that the medication has changed my patients' sleep? Or are the variations that the patients observed random and unrelated to the intervention?

```
## [1] 1.11 2.68 0.55 1.81 0.20
```

Your turn. What is the null hypothesis?

Example solution

The H_0 is that the true mean of the population is 0.

```
t.test(sleep_times)
```

```
##
##  One Sample t-test
##
## data:  sleep_times
## t = 2.851, df = 4, p-value = 0.04635
## alternative hypothesis: true mean is not equal to 0
## 95 percent confidence interval:
##  0.03322 2.50678
## sample estimates:
## mean of x
##      1.27
```

If p $\leq$ .05: reject the null hypothesis.

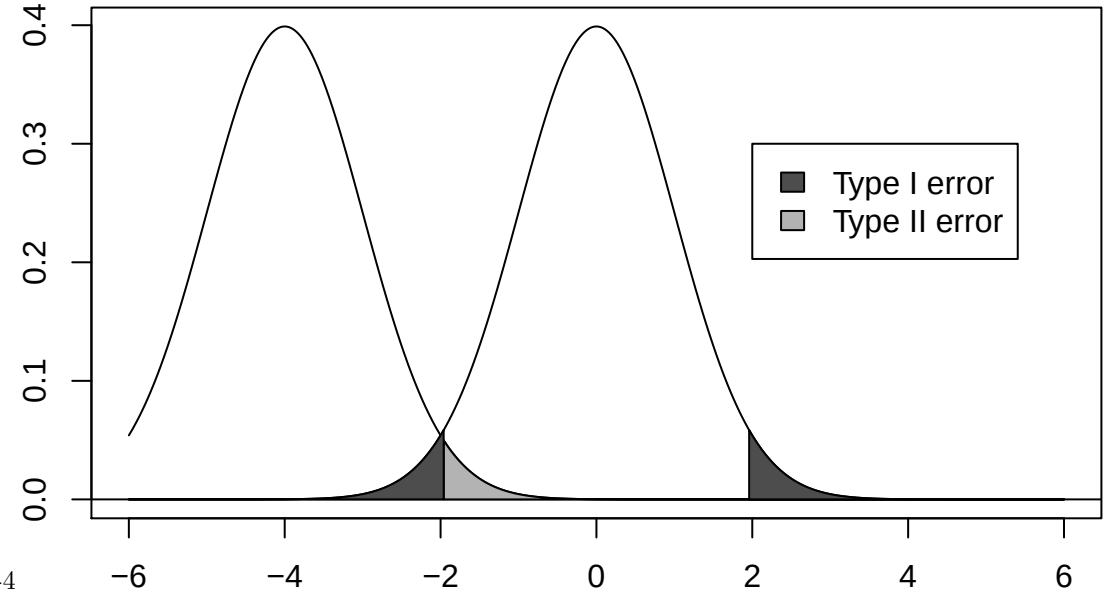
Power

- We've talked about the probability of the type I error (α, which we want to be no greater than 5%).
- What about the opposite error, where there is an actual effect but we fail to reject the null hypothesis?

Test result	No true effect	True effect
H_0 rejected	Type I error (α)	correct (Power)
H_0 not rejected	correct ($1-\alpha$)	Type II error (β)

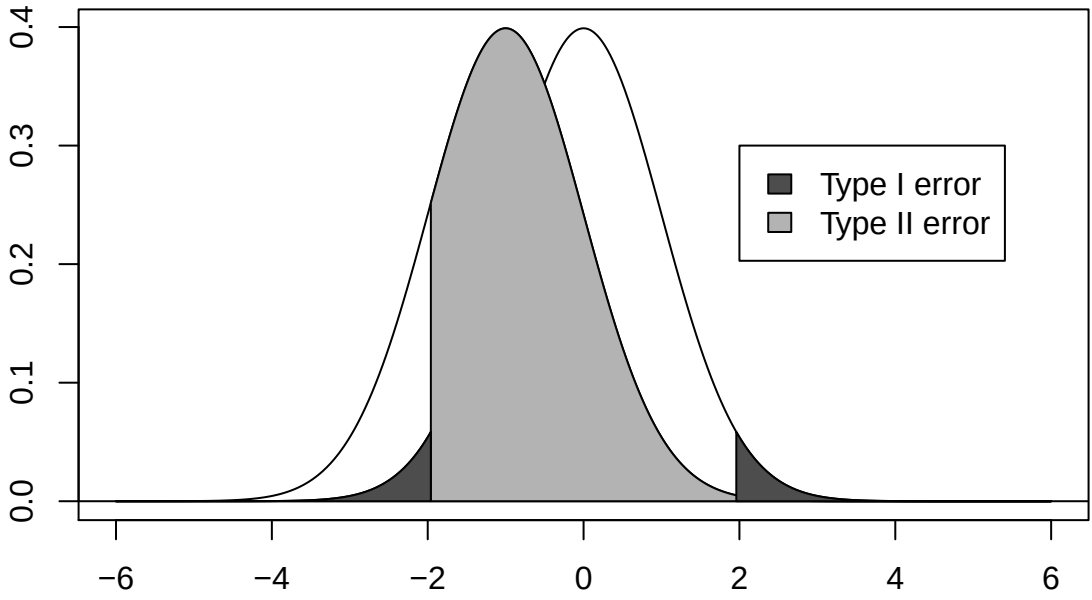
- Power: The probability of correctly rejecting the H_0 given that there is an actual effect in the population.
- By the way, you **never** accept the H_0, you just fail to reject it (the table in Shravan's statistics notes got that wrong).

Plotting the situation



- Here, the true mean is -4

A much worse situation with severe power issues



- Here, the true mean is -1.

Power simulations

- For simple (and even more complex) designs, you can compute power analytically.
- But simulations are a lot easier!
- Let's go back to the sleep example and assume that the true mean was 1 (that means that on average, people get one hour more sleep when using the medication) and the sd was 1.

```
t_test_sim <- function(n, mean = 1, sd = 1){
  t_results <- t.test(rnorm(n, mean, sd))
  t_results$p.value <= .05}
table(replicate(1000, t_test_sim(5, 1, 1)))
```

```
##
## FALSE  TRUE
##   600   400
```

Not so great!

How to increase power

Let's try some different scenarios: - The effect size in the population is larger (people sleep longer on the medication)

```
table(replicate(1000, t_test_sim(n = 5, mean = 2, sd = 1)))
```

```
##
## FALSE  TRUE
##    98   902
```

- The standard deviation (i.e. the noise) in the population is lower (people don't vary as much in their response to the medication)

```
table(replicate(1000, t_test_sim(n = 5, mean = 1, sd = .5)))
```

```
##
## FALSE  TRUE
##   100   900
```

How to increase power (realistically!)

You don't really have any direct control over population mean (i.e. effect size) or sd (i.e. noise). Let's focus on the one variable that you do have control over. - The sample size is larger

```
table(replicate(1000, t_test_sim(n = 7, mean = 1, sd = 1))) # not quite enough
```

```
##
## FALSE  TRUE
##   378   622
```

```r
table(replicate(1000, t_test_sim(n = 10, mean = 1, sd = 1))) # now we're talking!
```

```
##
## FALSE  TRUE
##   180   820
```

Double-checking our results

- Let's just check analytically that we have this correctly: If we want to show in the one-sample t-test that a mean of 1 is different from 0 (when sd = 1), we need about 10 subjects. R has a function for that!

```r
power.t.test(n = 10, delta = 1, sd = 1, type = "one.sample")
```

```
##
##      One-sample t test power calculation
##
##               n = 10
##           delta = 1
##              sd = 1
##       sig.level = 0.05
##           power = 0.8031
##     alternative = two.sided
```

Remarkably similar to our simulation! (See Vasishth, Chapter 3 if you want to know how R calculates this!)

Setting yourself up for success (or failure)

- You don't want to run an underpowered study. Most likely, you'll get a null result that tells you nothing about the true state of the world.
- How can you avoid this?
- Run a realistic number of participants so you reach acceptable power (the APA recomments .8):

```r
power.t.test(delta = 1, sd = 1, power = .8, type = "one.sample")
```

```
##
##      One-sample t test power calculation
##
##               n = 9.938
##           delta = 1
##              sd = 1
##       sig.level = 0.05
##           power = 0.8
##     alternative = two.sided
```

Exercise

> An experimenter knows for a fact that the average number of friends people have on Facebook is 70, with an sd of 10. She knows this because she works for Facebook and has access to all your personal data. The experimenter wants to know if people who post lots of photos of cats have more or fewer friends than the average Facebook user. Automatically tagging cat photos is hard, so our experimenter just asks an unpaid intern to compile a sample of 100 cat-posting people and find out their friend numbers. How big does the effect (in friends gained/lost) have to be so it would be detectable at an acceptable power level of .8?

Solution

```
power.t.test(n = 100, sd = 10, power = .8, type = "one.sample")
```

```
##
##      One-sample t test power calculation
##
##               n = 100
##           delta = 2.829
##              sd = 10
##       sig.level = 0.05
##           power = 0.8
##     alternative = two.sided
```

A difference in as little as 2.9 friends would be detectable.

Don't cheat!

- How about the following strategy?

  > Just run the hypothesis test on the data after every new sample and stop as soon as you get a significant result.

- Let's see just what happens to α if you do that.
- Run a simulation where there is no effect (i.e. where we know the H_0 is true)

```
t_test_cheating_sim <- function(n_max = 30, n_increments = 2, sd = 1){
  samples <- NULL
  significant <- FALSE

  while(length(samples) <= n_max & significant == FALSE){
      samples <- c(samples, rnorm(n_increments, mean = 0, sd = sd))
      significant <- t.test(samples)$p.value  <= .05
    }
  return(significant)
  }
```

The consequences of cheating

Let's run this simulation 1000 times and make a table with the results:

```
table(replicate(1000, t_test_cheating_sim(n_max = 30, n_increments = 2, sd = 1)))
```

```
##
## FALSE  TRUE
##   744   256
```

- Whoa! False positive alert!
- α is at 25%, instead of 5% where it should be.
- Unfortunately, this strategy of using stopping rules ("data peeking") is quite common.
- Solution: do a power analysis, set your sample size beforehand, and stick to it!

Testing more interesting hypotheses

- So far, we have been testing the null hypothesis that our sample mean is 0.
- This is not what we usually do in Psychology.
- Instead, we want to know if there is a significant difference between the means of two (or more) samples.
- For example, you might give only one group an intervention against anxiety, with the other one serving as the control.
  - Does the intervention work?
  - Do people in the treatment group report lower anxiety?
  - Can we generalise this to the population?
  - Should we use this intervention in clinical practice?
- A lot of effort and money may be wasted if you get these questions wrong.

The two-sample t-test

- Remember, we are comparing two samples now. We'll call the sample means μ_1 and μ_2.
- Our null hypothesis is $H_0 : \mu_1 = \mu_2$
- We can rephrase this as $H_0 : \mu_1 - \mu_2 = \delta = 0$
- We already know the logic of this: we just want to find out if δ is extreme enough so we can reject the H_0.
- Let's see how δ is distributed.

```
simulate_d <- function(n1, n2, mean1, mean2, sd1, sd2){
  mean(rnorm(n1, mean1, sd1)) - mean(rnorm(n2, mean2, sd2))
}
```

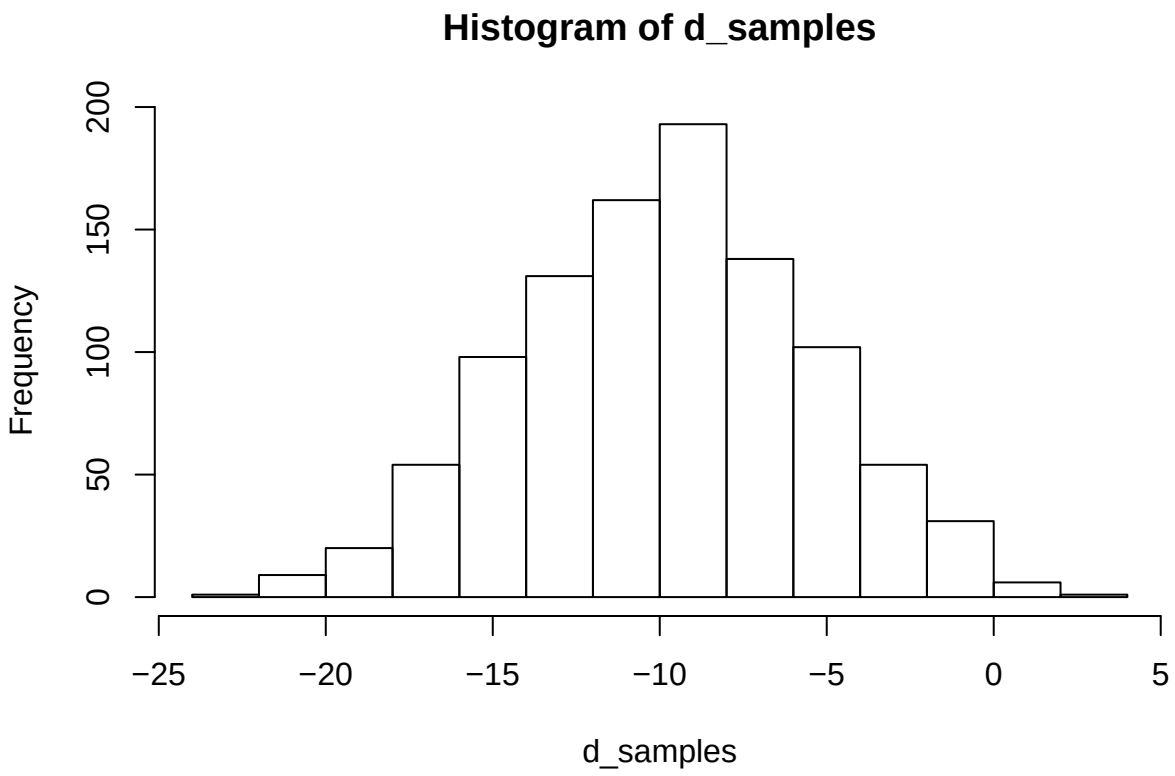
The two-sample t-test (2)

Now run the simulation and make a histogram.

```r
d_samples <- replicate(1000, simulate_d(n1 = 10, n2 = 10, mean1 = 50, mean2 = 60, sd1 = 10, sd2 = 10))
paste("Mean =", round(mean(d_samples), 2), "SD = ", round(sd(d_samples),2))
```

```
## [1] "Mean = -9.84 SD =  4.37"
```

```r
hist(d_samples)
```



The two-sample t-test (3)

- OK, this looks normally distributed enough
- The mean of this distribution seems to be centered on the difference between μ_1 and μ_2.
- But what is the standard deviation? Let's try what happens if we change the sd of the two groups.

```r
d_samples <- replicate(1000, simulate_d(n1 = 10, n2 = 10, mean1 = 50, mean2 = 60, sd1 = 20, sd2 = 10))
sd(d_samples)
```

```
## [1] 7.161
```

```r
d_samples <- replicate(1000, simulate_d(n1 = 10, n2 = 10, mean1 = 50, mean2 = 60, sd1 = 10, sd2 = 20))
sd(d_samples)
```

```
## [1] 7.044
```

The two-sample t-test (3)

- Let's also change the sample size

```
d_samples <- replicate(1000, simulate_d(n1 = 20, n2 = 10, mean1 = 50, mean2 = 60, sd1 = 20, sd2 = 10))
sd(d_samples)
```

```
## [1] 5.538
```

```
d_samples <- replicate(1000, simulate_d(n1 = 10, n2 = 20, mean1 = 50, mean2 = 60, sd1 = 20, sd2 = 10))
sd(d_samples)
```

```
## [1] 6.57
```

- Looks like the sd of this distribution goes up as the sd of the two sample populations goes up and goes down as the size of one or both of the samples goes down.

The two-sample t-test (4)

- We could play with the simulations until we've figured the relationship out, but as a shortcut, here it is:

$$\sigma_{\bar{x}_1-\bar{x}_2} = \sqrt{\frac{\sigma_1^2}{n_1} + \frac{\sigma_2^2}{n_2}}$$

- Based on this, we could calculate CIs or just a z-value (why not t? Hold your horses!)
- Remember our $H_0 : \mu_1 - \mu_2 = 0$
- The z-value would then be: $z = \frac{(\bar{x}_1-\bar{x}_2)-(\mu_1-\mu_2)}{\sqrt{\frac{\sigma_1^2}{n_1}+\frac{\sigma_2^2}{n_2}}} = \frac{(\bar{x}_1-\bar{x}_2)}{\sqrt{\frac{\sigma_1^2}{n_1}+\frac{\sigma_2^2}{n_2}}}$
- (since our null hypothesis is that $\mu_1 - \mu_2 = 0$)

The two-sample t-test (5)

- Of course, in real life we don't know the population sd
- So we have to estimate it using s^2
- This would be a *t*-value, not a z-value

$$t = \frac{(\bar{x}_1-\bar{x}_2)}{\sqrt{\frac{s_1^2}{n_1}+\frac{s_2^2}{n_2}}}$$

- Only problem: what is the df of that test?
- There are some shortcuts that we can take if the sample sizes and population variances are the same, but is there a general solution?
- Shravan suggests we should just use the lower of the sample sizes, but this will cost us power
- This was actually a big problem in statistics, but B. L. Welch found an approximate solution (called *Welch's t-test*)
- You can look the details up on Wikipedia, but R knows them and applies them automatically when you use `t.test`.

The dependent t-test for paired samples

- This is actually a lot easier. Since we have two samples per person/group/analysis unit, we can simply compute the differences between measurements and then use the one-sample t-test to check if they are 0.
- Since the sd of the differences will be a lot lower than the overall sd, the power of this test is quite a bit higher.
- We'll get back to that next week when we're talking about ANOVAs.
- In R, use `t.test(..., paired = TRUE)` to perform this test.