Lab 07 - Part 1: Sampling Distributions (Normal Distribution & Central Limit Theorem)

By a small sample, we may judge of the whole piece. —Miguel de Cervantes from Don Quixote

Lab Introduction

- This lab focuses on simulating/faking data rather than using real data
- **Purpose:** Learning data simulation helps understand real data better.
- **Philosophy:** If you can't simulate expected data, you can't understand actual data well

Generating Numbers in R

There are many ways to make R generate numbers for you. In all cases you define how the numbers are generated. We'll go through a few of the many ways.

sample() The sample function is like an endless gumball machine. You put the gumballs inside with different properties, say As and Bs, and then you let sample endlessly take gumballs out. Check it out:

```
gumballs <- c("A","B")
sample_of_gumballs <-sample(gumballs, 10, replace=TRUE)
sample_of_gumballs
```

```
##  [1] "B" "B" "B" "A" "A" "B" "B" "A" "A" "A"
```

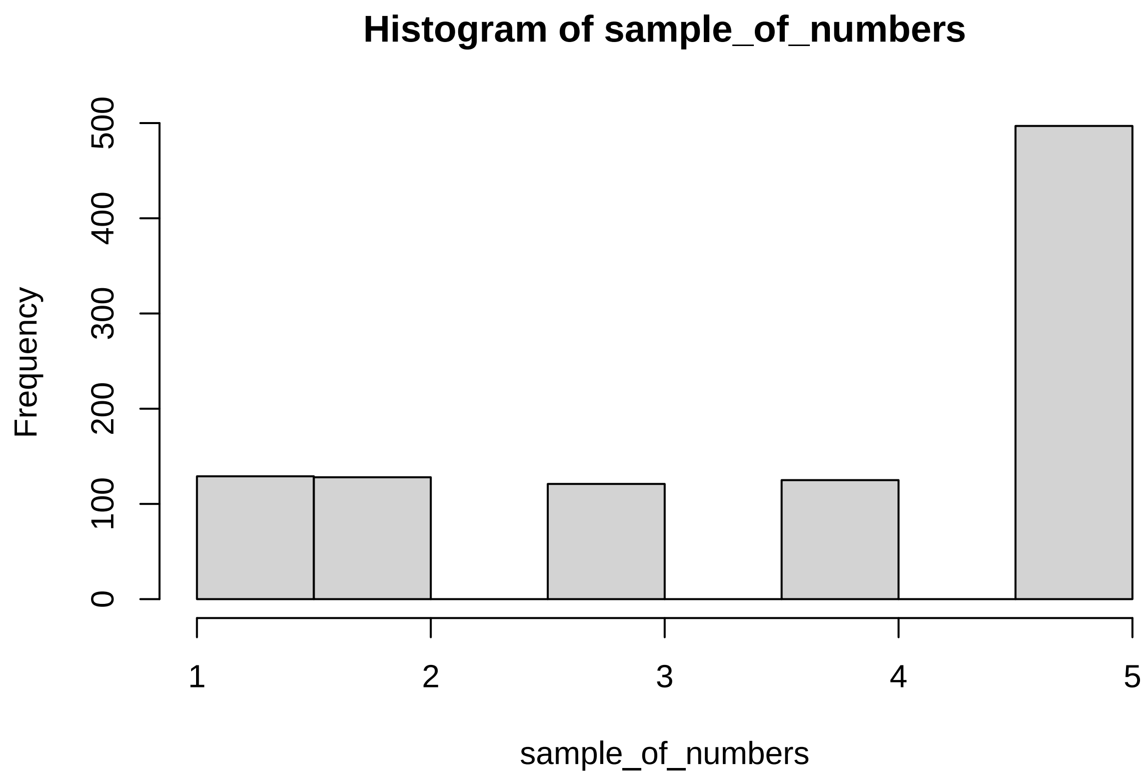
Here the sample function randomly picks A or B each time. We set it do this 10 times, so our sample has 10 things in it. We set `replace=TRUE` so that after each sample, we put the item back into the gumball machine and start again. Here's another example with numbers

```
some_numbers <- c(1,2,3,4,5,5,5,5)
sample_of_numbers <-sample(some_numbers, 20, replace=TRUE)
sample_of_numbers
```

```
##  [1] 4 5 4 1 5 5 5 1 2 5 2 5 1 4 5 5 5 5 1 4
```

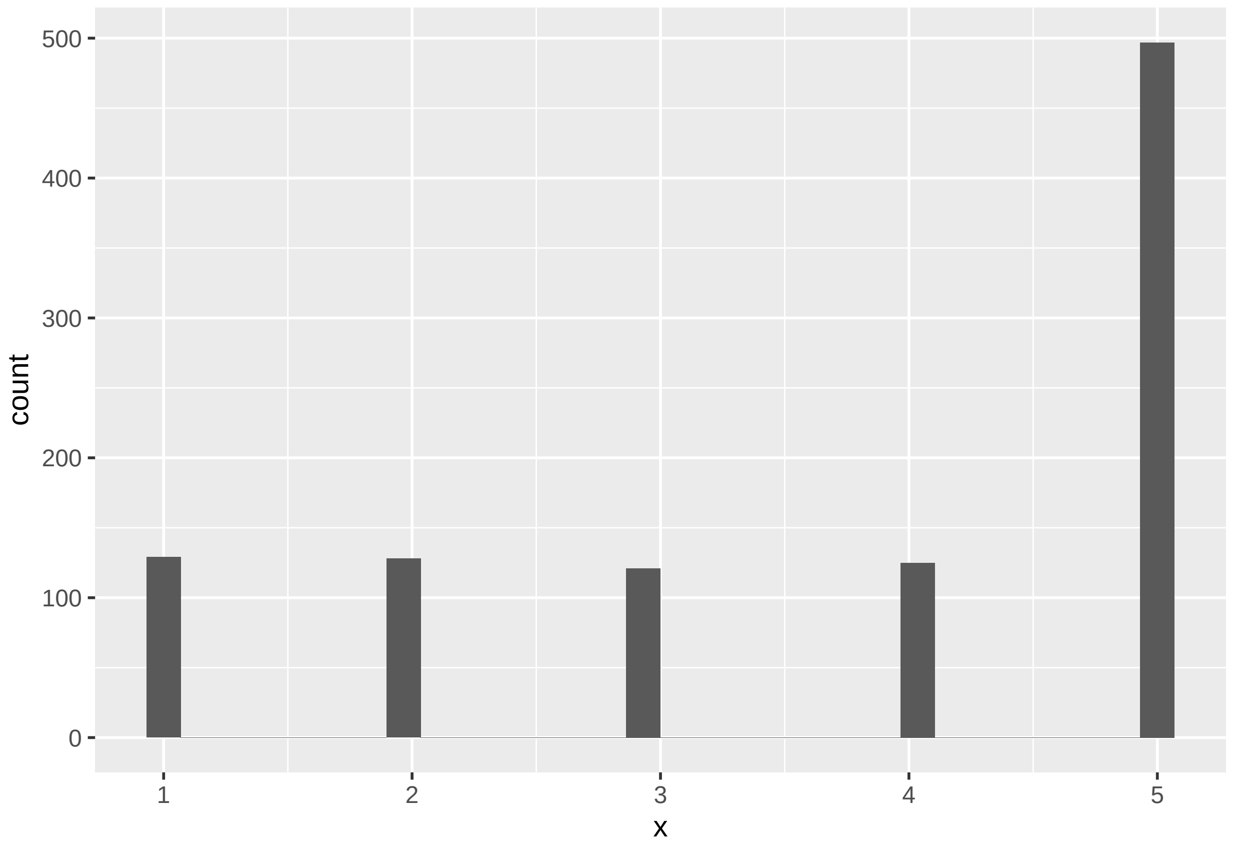
Let's do one more thing with sample. Let's sample 1000 times from our `some_numbers` variable, and then look at the histogram

```
#??
library(ggplot2)
some_numbers <- c(1,2,3,4,5,5,5,5)
sample_of_numbers <-sample(some_numbers, 1000, replace=TRUE)

#Histogram in Base R
hist(sample_of_numbers)
```



```
#Histogram in ggplot
ggplot(data.frame(x = sample_of_numbers), aes(x = x)) +
  geom_histogram()
```

```
## `stat_bin()` using `bins = 30`. Pick better value `binwidth`.
```



We are looking at lots of samples from our little gumball machine of numbers. We put more 5s in, and voila, more 5s come out of in our big sample of 1000.

The parameters taken by the sample() function are:

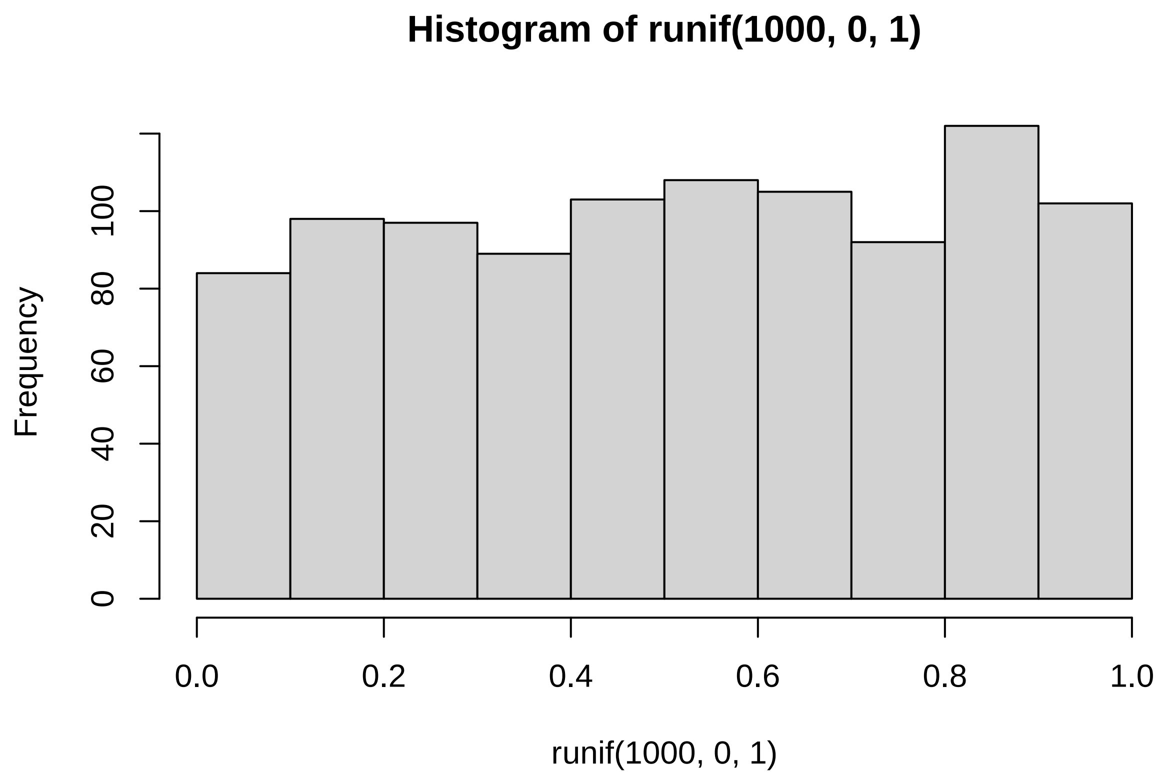
- x (vector of items to sample from),
- size (number of items to sample),
- and replace (TRUE/FALSE - whether to sample with replacement)

When `replace=False`, you essentially have a finite gumball machine. We are only going to be considering sampling with replacement in this lab - because that's more relevant.

runif uniform distribution We can sample random numbers between any range using the `runif(n, min=0, max = 1)` function for the uniform distribution.

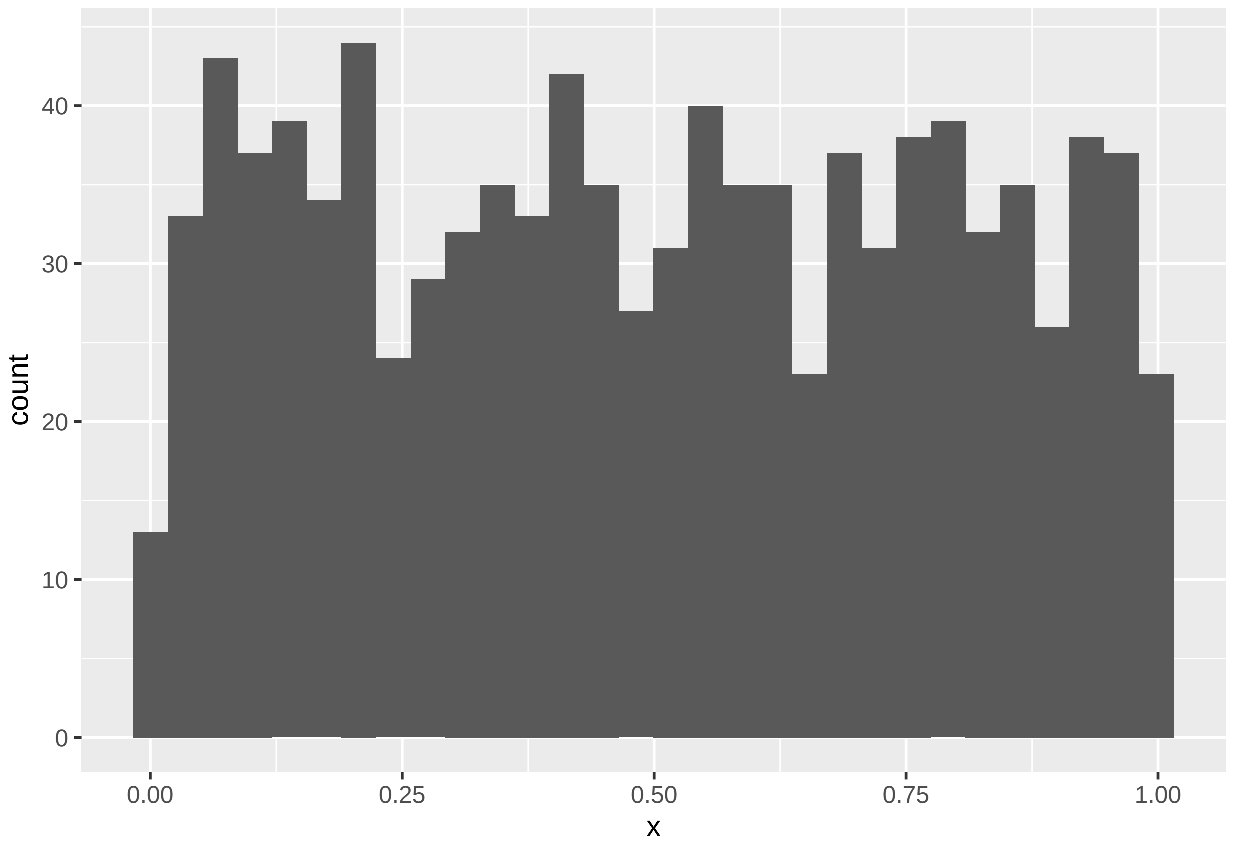
A uniform distribution is flat, and all the numbers between the min and max should occur roughly equally frequently. Let's take 1000 random numbers between 0 and 1 and plot the histogram. We'll just do it all in one line for speed.

```r
#Histogram in base R
hist(runif(1000,0,1))
```



```r
#Histogram in ggplot
ggplot(data.frame(x = runif(1000,0,1)), aes(x = x)) +
  geom_histogram()
```

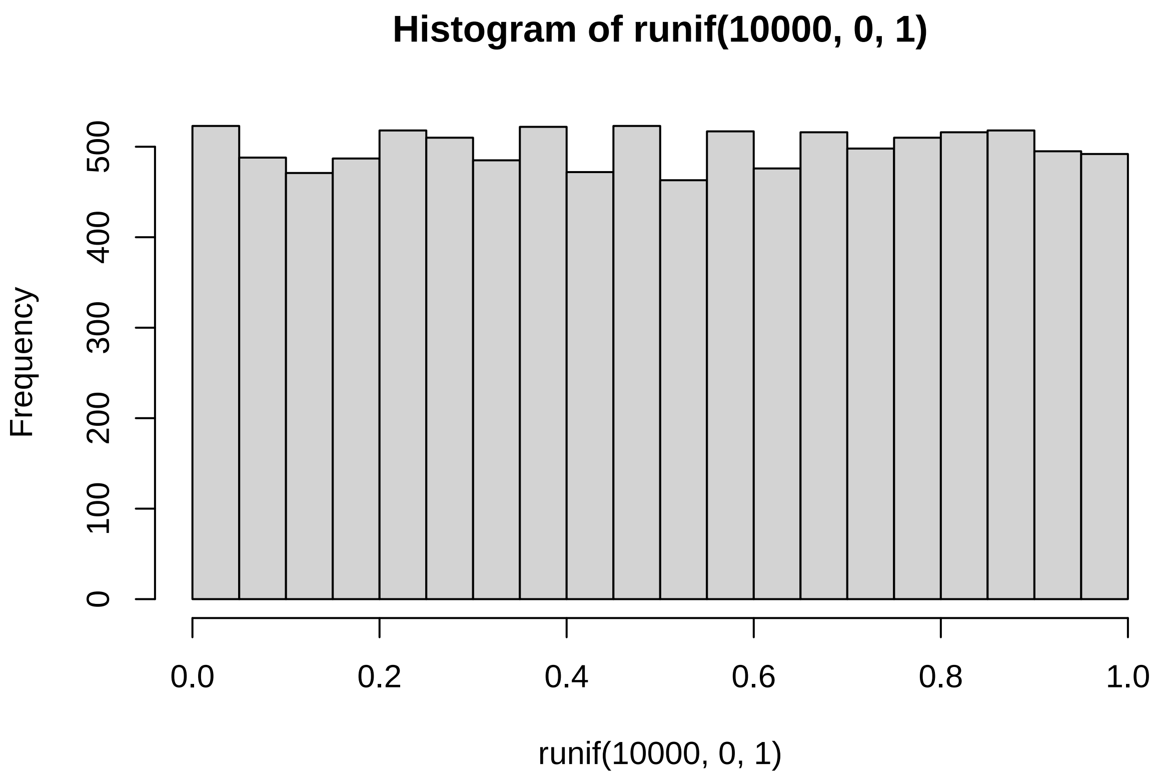
```
## `stat_bin()` using `bins = 30`. Pick better value `binwidth`.
```



This is histogram is flattish. Not perfectly flat, after all we only took 1000 samples.

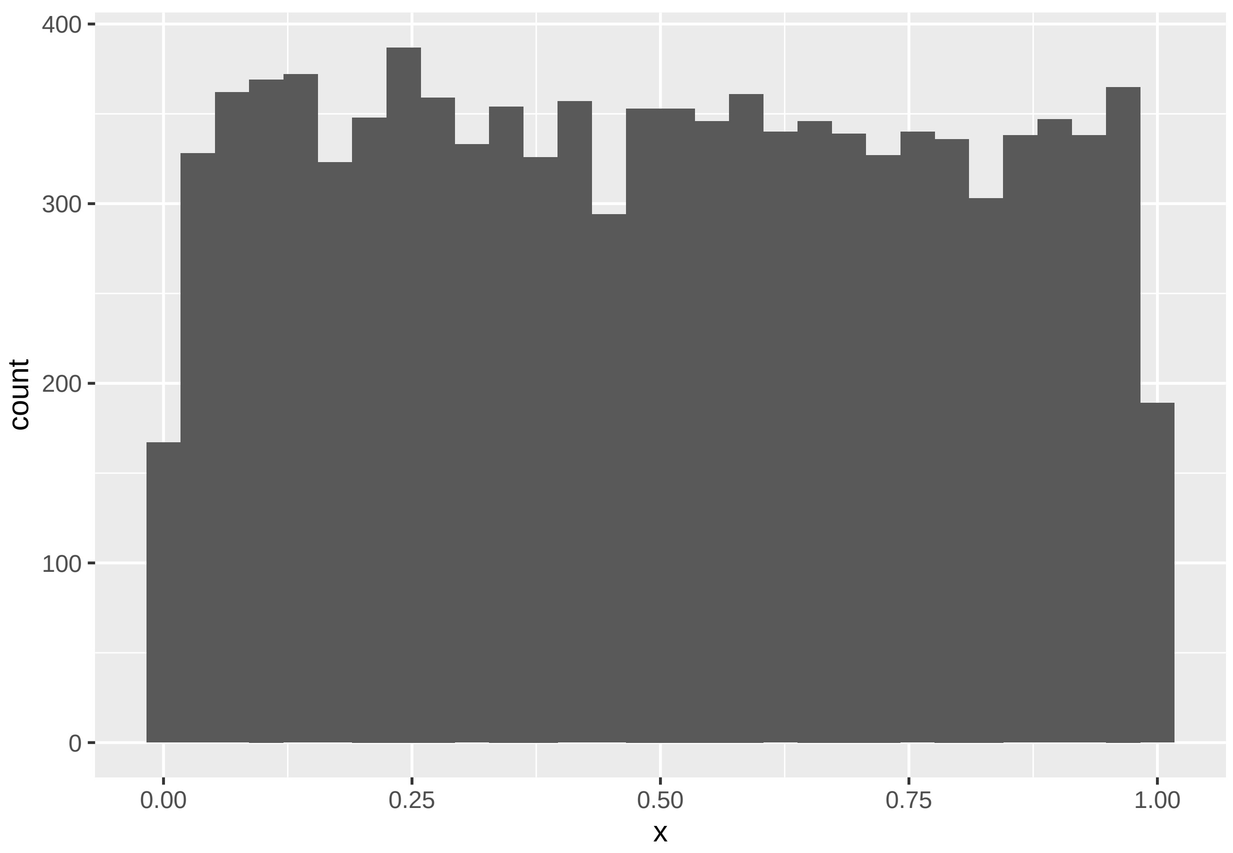
Q1.What if we took many more, say 10,000 total samples? Probably it will become more flat, since the sampling variation goes down.

```r
hist(runif(10000,0,1))
```



```r
ggplot(data.frame(x = runif(10000,0,1)), aes(x = x)) +
  geom_histogram()
```

```
## `stat_bin()` using `bins = 30`. Pick better value `binwidth`.
```



rbinom the binomial distribution The binomial distribution sounds like a scary word… binomial (AAGGGGHHHHH, stay away!).

The binomial can be a coin flipping distribution. You use `rbinom(n, size, prob)`.

- `n` gives the number of samples you want to take.
- We'll keep `size = 1` for now, it's the number of trials (let's not worry about this for the moment)
- `prob` is a little list you make of probabilities, that define how often certain things happen.

Now, consider flipping a coin. It will be heads or tails, and the coin, if it is fair, should have a 50% chance of being heads or tails.

Here's how we flip a coin 10 times using `rbinom`.

```
coin_flips <- rbinom(10,1,.5)
coin_flips
```

```
##  [1] 0 1 0 1 0 0 0 0 0 0
```

We get a bunch of 0s, and 1s. We can pretend 0 = tails, and 1 = heads.

Great, now we can do coin flipping if we want.

For example, if you flip 10 coins, how many heads do you get?

We can can do the above again, and then `sum(coin_flips)`. All the 1s are heads, so it will work out.

```r
coin_flips <- rbinom(10,1,.5)
sum(coin_flips)
```

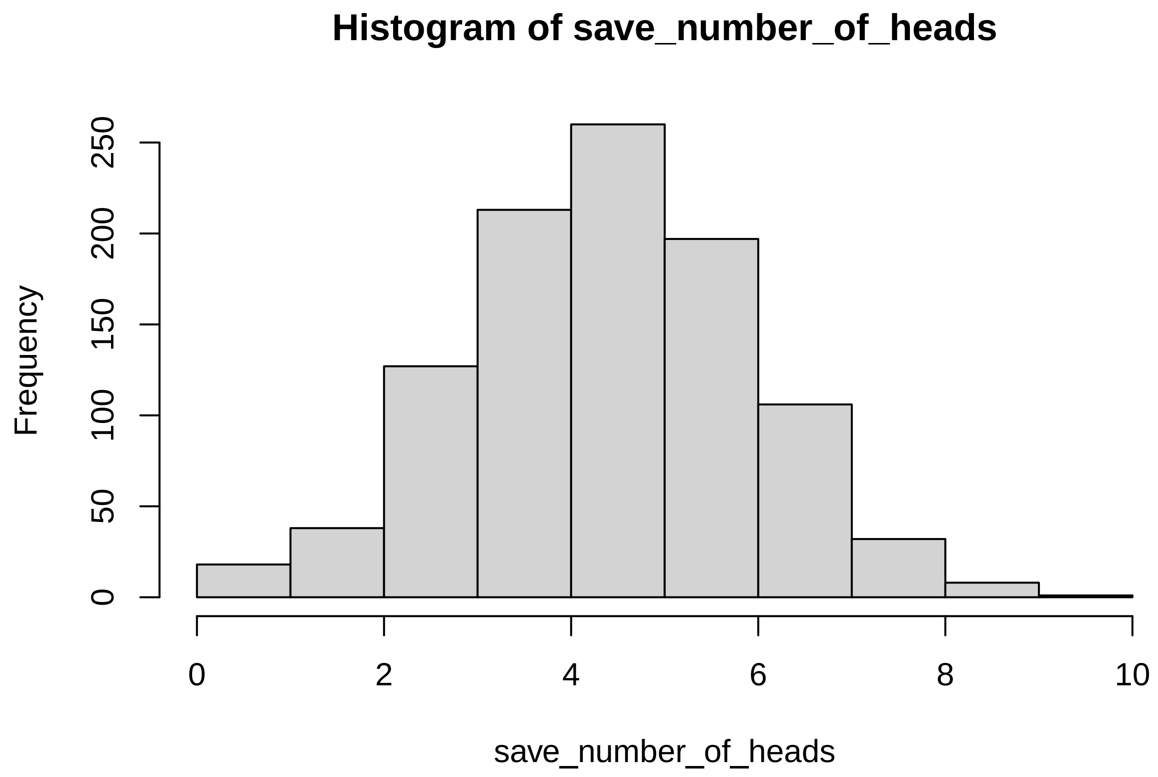
```
## [1] 7
```

Alright, so we get the sum, which tells us the number of heads. But, should we always get that number of heads if we flipped a coin 10 times?

If you keep redoing the above, you'll get different answers. 5 heads will be the most frequent answer, but you will get lots of other answers too.

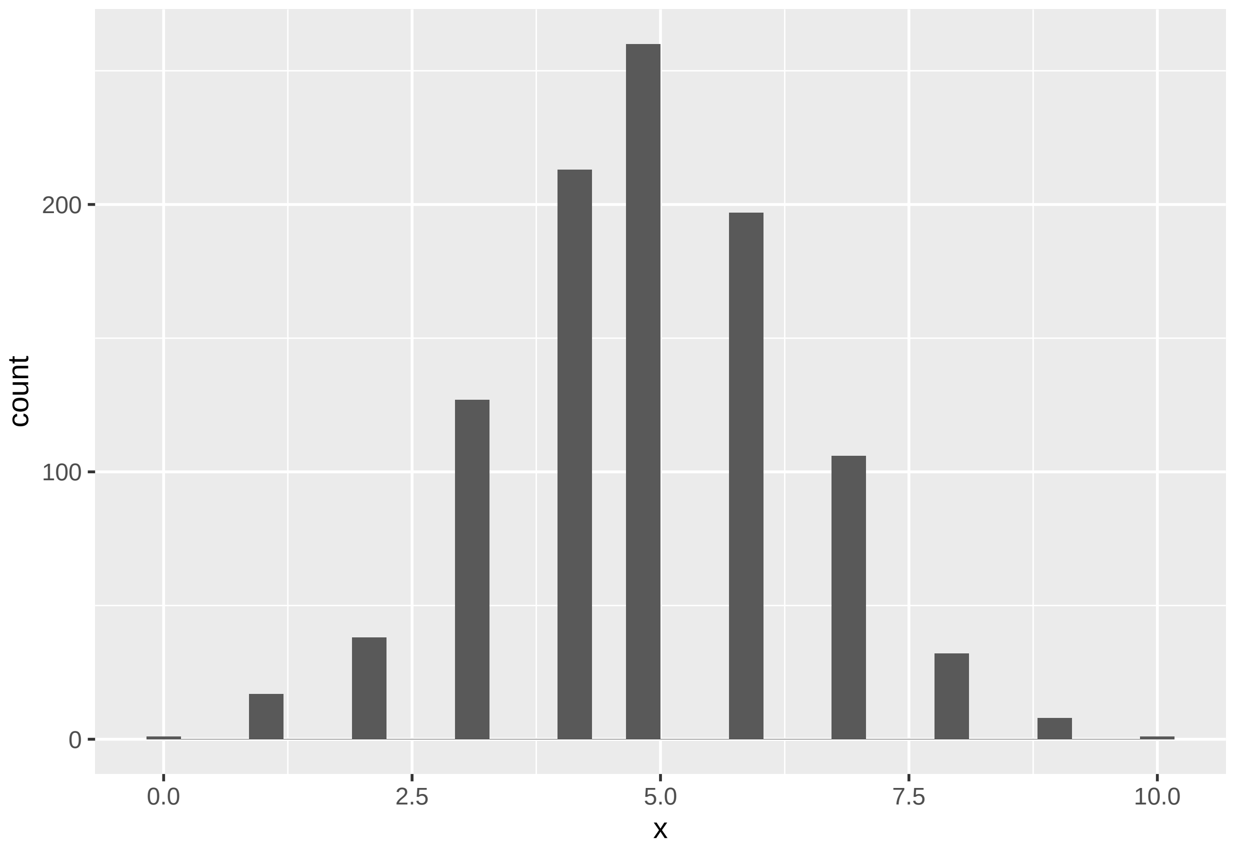
Hold on to your seats for this next one. With R, we can simulate the flipping of a coin 10 times (you already know that, you just did it), and we can do that over and over as many times as we want. For example, we could do it 1000 times over, saving the number of heads for each set of 10 flips. Then we could look at the distribution of those sums. That would tell us about the range of things that can happen when we flip a coin 10 times. We can do that in loop like this:

```r
# make an empty variable to save things in
save_number_of_heads<-length(1000)

for(i in 1:1000){
  save_number_of_heads[i] <- sum(rbinom(10,1,.5))
}

#Histogram in base R
hist(save_number_of_heads)
```



```r
#Histogram in ggplot
ggplot(data.frame(x = save_number_of_heads), aes(x = x)) +
  geom_histogram()
```

```
## `stat_bin()` using `bins = 30`. Pick better value `binwidth`.
```



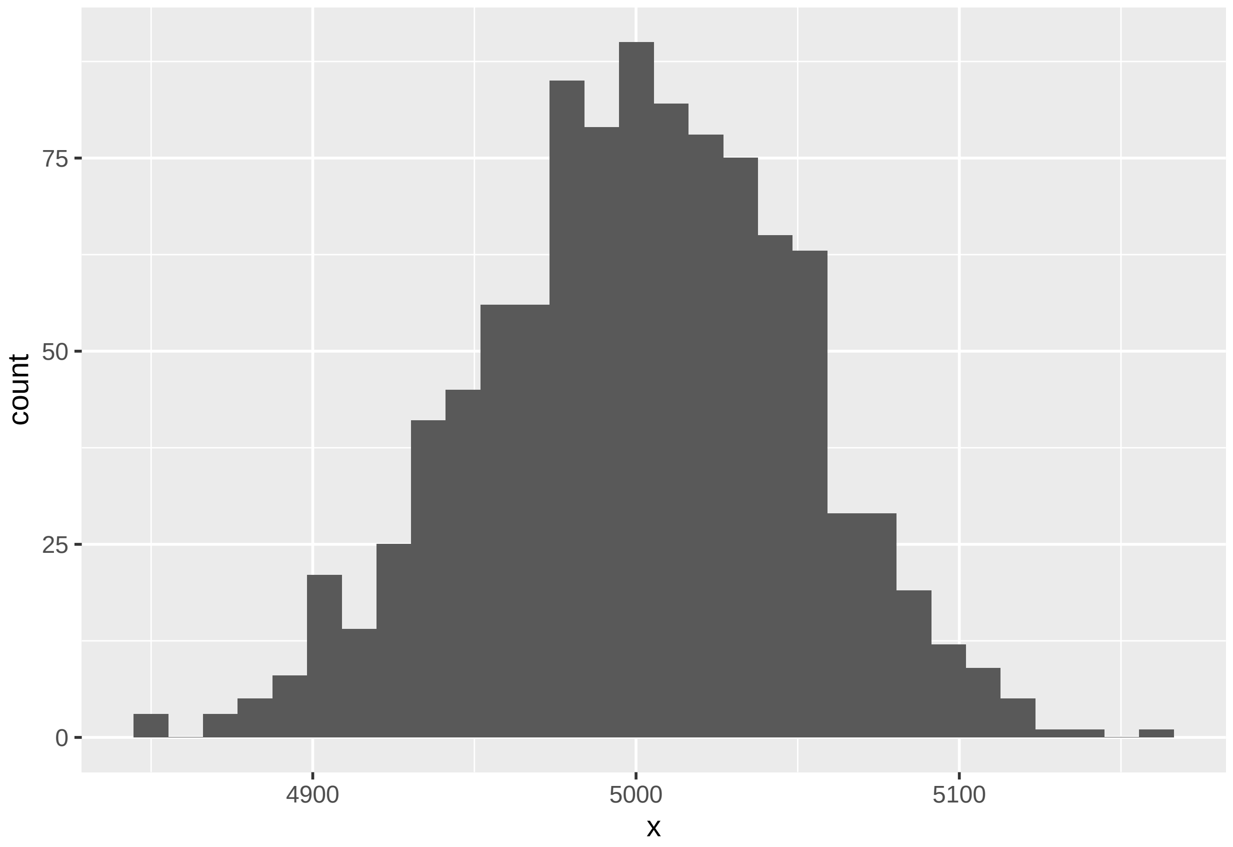
Q2. Intepret what is being plotted in the histogram. We made 10 coin-flipping scenarios to see how many heads we get, and repeated it for 1,000 times to see the overall distribution

See, that wasn't too painful. But even so, this is a common enough setting: X experiments (1000 in the above example) of size = N trials (10 in the above example). And that's what the size parameter in rbinom is trying to capture.

```r
for(i in 1:1000){
  save_number_of_heads[i] <- sum(rbinom(1000,10,.5))
}

ggplot(data.frame(x = save_number_of_heads), aes(x = x)) +
  geom_histogram()
```

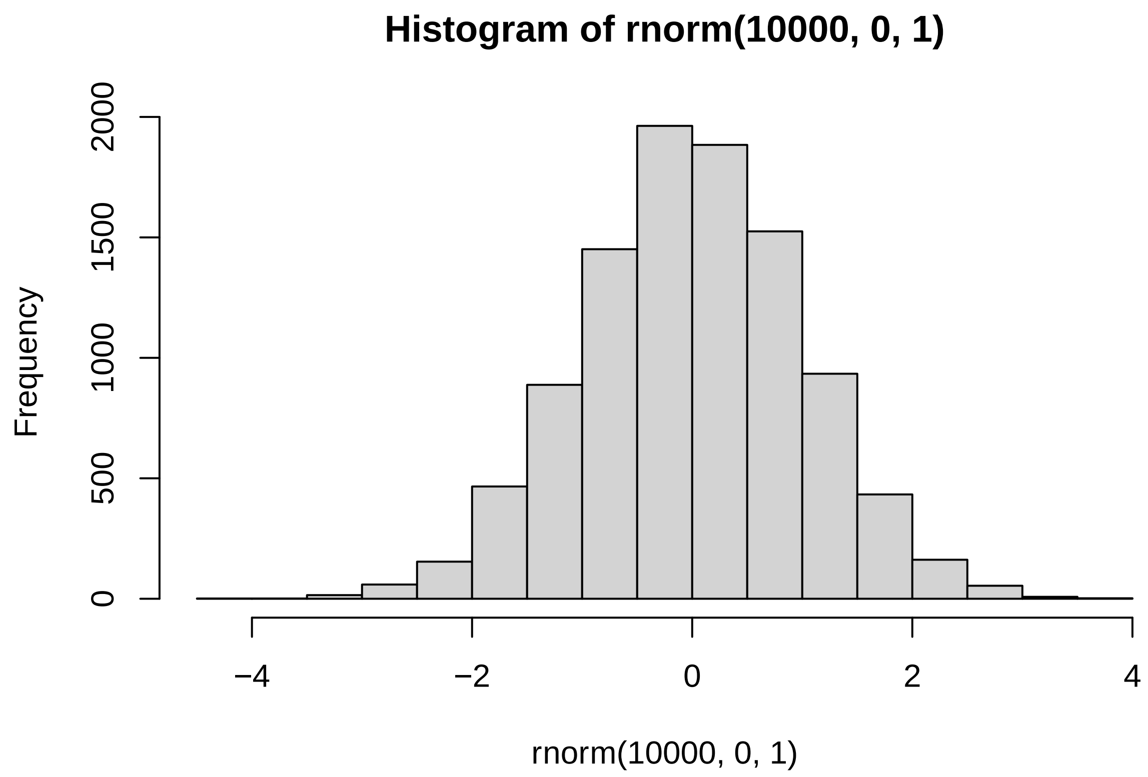
Q3. Carry out the same simulation as above (1000 experiments of 10 coin tosses each) with a single command by changing the size parameter in rbinom. Plot it.

```
## `stat_bin()` using `bins = 30`. Pick better value `binwidth`.
```



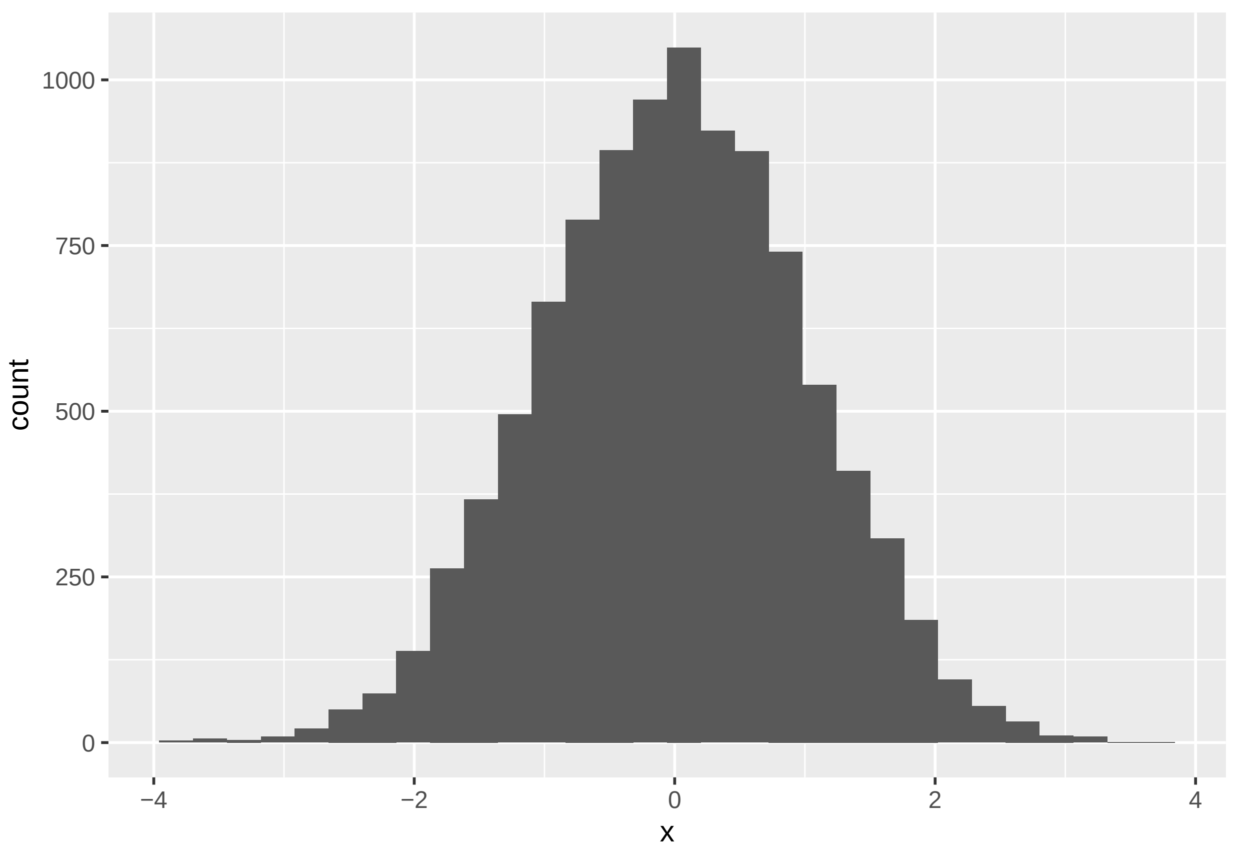
rnorm the normal distribution We'll quickly show how to use `rnorm(n, mean=0, sd=1)` to sample numbers from a normal distribution. And, then we'll come back to the normal distribution later, because it is important.

```
#Histogram in base R
hist(rnorm(10000,0,1))
```



```r
#Histogram in ggplot
ggplot(data.frame(x = rnorm(10000,0,1)), aes(x = x)) +
  geom_histogram()
```

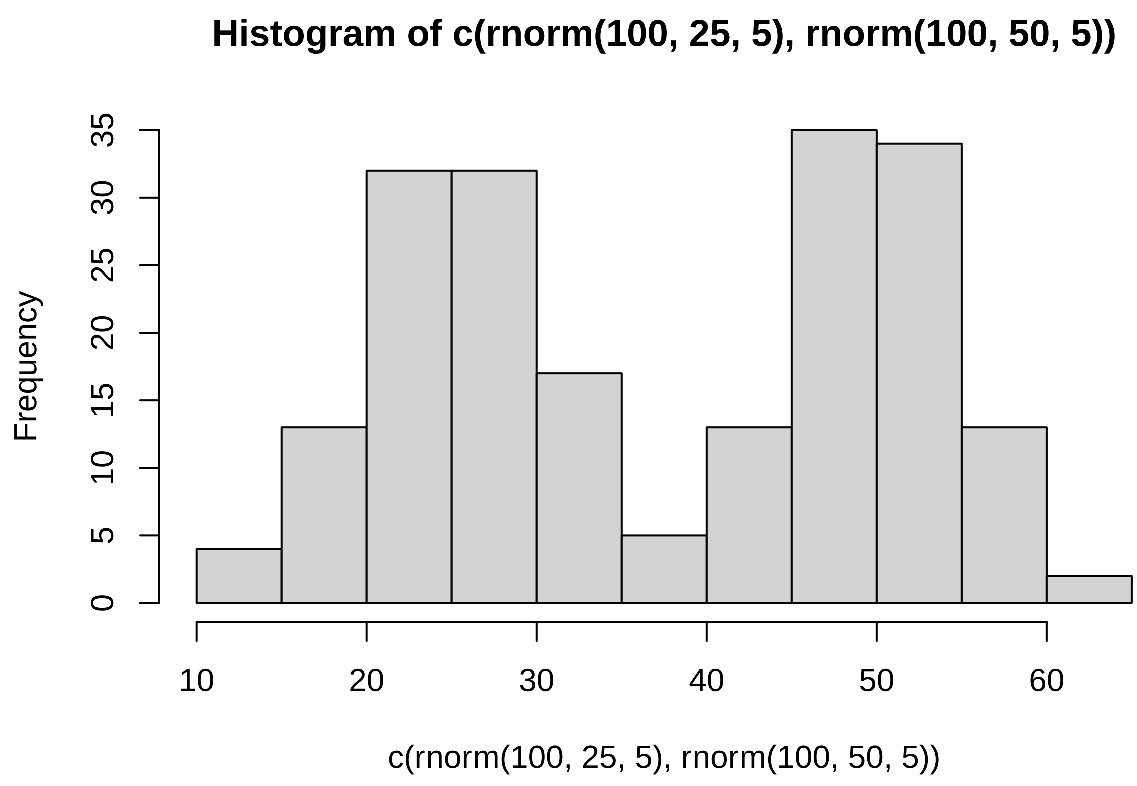
```
## ‘stat_bin()‘ using ‘bins = 30‘. Pick better value ‘binwidth‘.
```



There it is, a bell-shaped normal distribution with a mean of 0, and a standard deviation of 1. You've probably seen things like this before. Now you can sample numbers from normal distributions with any mean or standard deviation, just by changing those parts of the `rnorm` function.

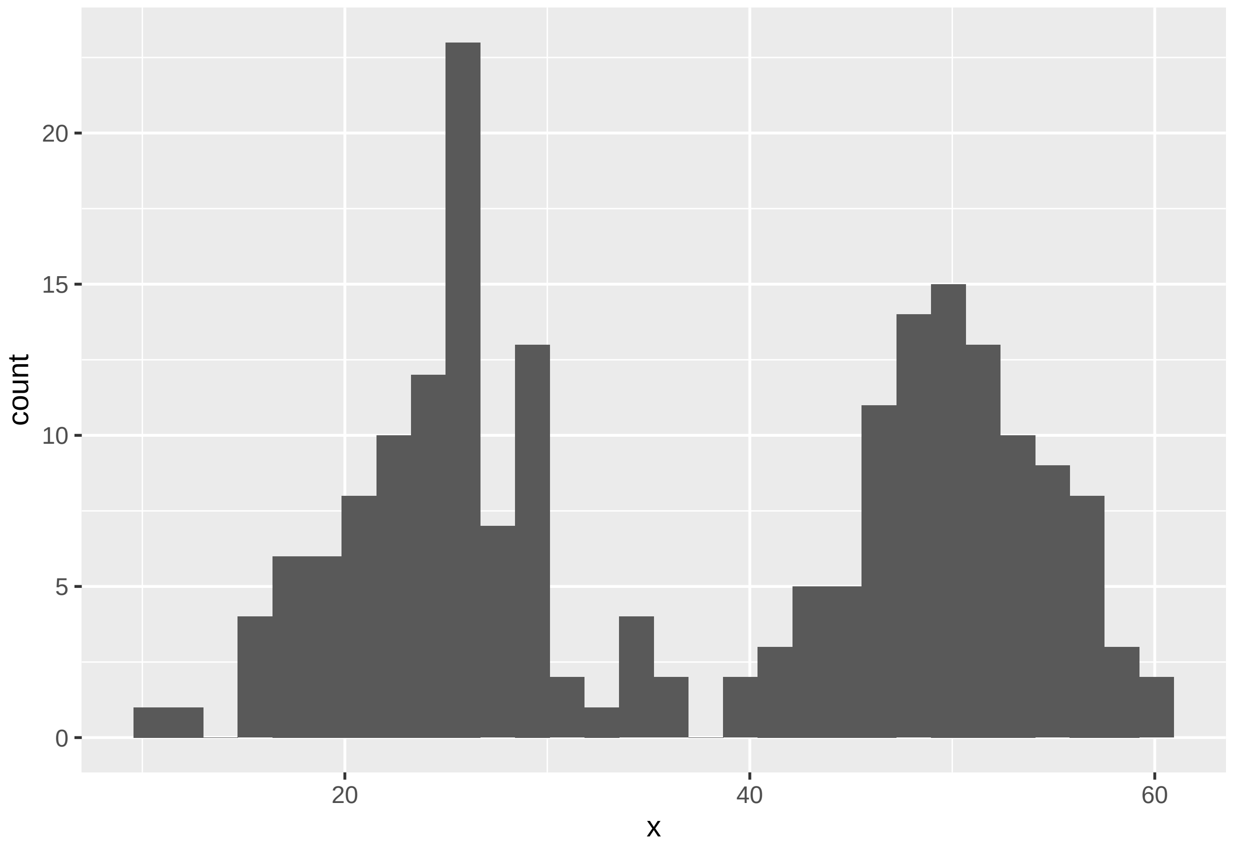
mixing it up The r functions are like Legos, you can put them together and come up with different things. What if wanted to sample from a distribution that looked like a two-humped camel's back? Just sample from `rnorm` twice like this... mix away.

```
#Histogram in base R
hist( c( rnorm(100,25,5), rnorm(100,50,5)) )
```



```r
#Histogram in ggplot
ggplot(data.frame(x = c( rnorm(100,25,5), rnorm(100,50,5))), aes(x = x)) +
  geom_histogram()
```

```
## ‘stat_bin()‘ using ‘bins = 30‘. Pick better value ‘binwidth‘.
```



The fancy term for what you such distributions which are a combination of simpler distribution is mixture distributions / mixture models. Bimodal / Multimodal observed data are often modeled by such mixtures whose mean and sds are fitted to the observations.

summary You can generate as many numbers as your computer can handle with R. PSA: Don't ask R to generate a bajillion numbers or it will explode (or more likely just crash, probably won't explode, that's a metaphor).

sampling distribution of the mean.

Now, with these tools, we can look at the sampling distribution of the sample means, for a sample size from a population.

(As we've seen, we can take samples from distributions in R. We can take as many as we want. We can set our sample-size to be anything we want. And, we can take multiple samples of the same size as many times as we want.)

Taking multiple samples of the same size Let's take 10 samples from a normal distribution (mean = 0, and SD = 1). Let's set the sample-size for each to be 20. Then, we'll put them all in a data frame and look at 10 different histograms, one for each sample.

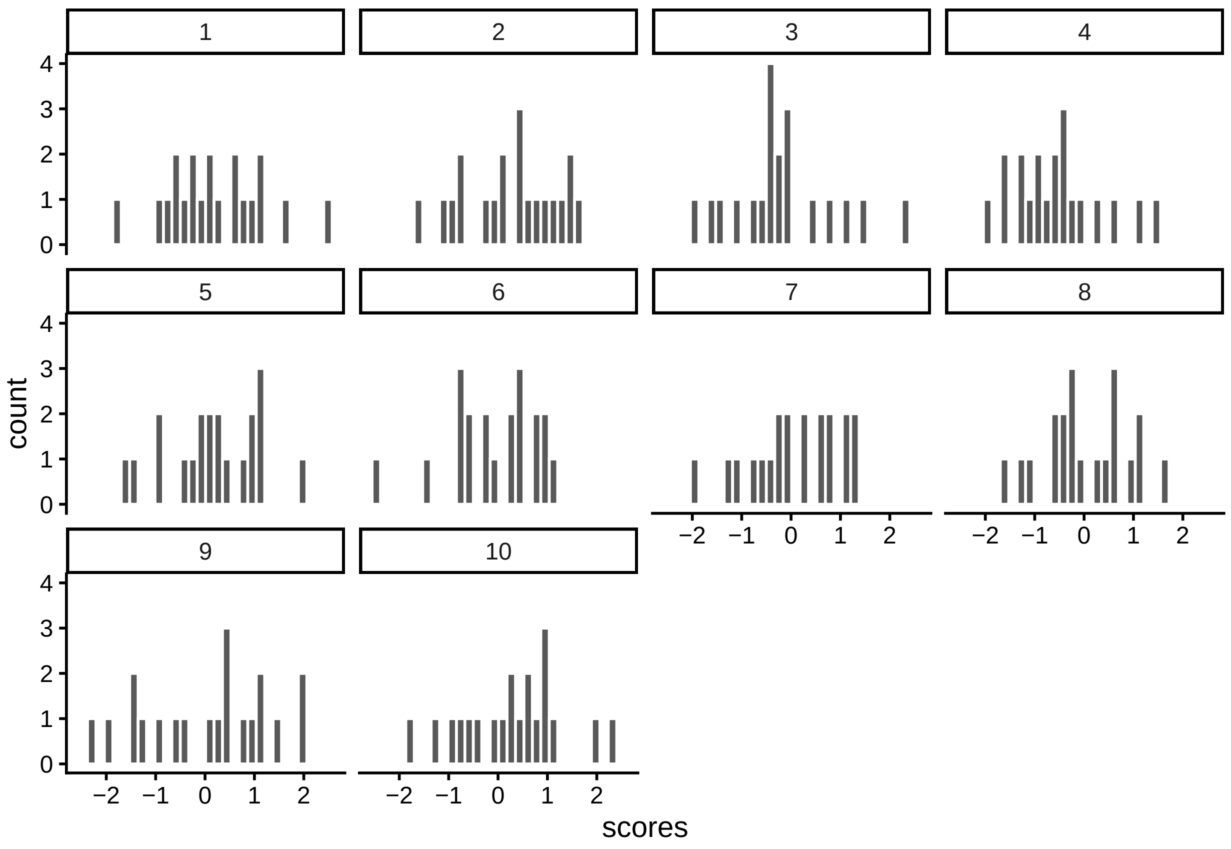
```
scores <- rnorm(10*20,0,1)
samples <- rep(1:10,each=20)
my_df <- data.frame(samples,scores)
head(my_df)
```

```
##   samples     scores
## 1       1  1.0563051
## 2       1  0.6702201
## 3       1  2.4896877
## 4       1  0.6712513
## 5       1 -1.7316601
## 6       1  0.9133404
```

First, look at the new my_df data frame. You can see there is a column with numbers 1 to 10, these are the sample names. There are also 20 scores for each in the scores column. Let's make histograms for each sample, so we can see what all of the samples look like:

```
ggplot(my_df, aes(x=scores))+
  geom_histogram(color="white")+
  facet_wrap(~samples)+
  theme_classic()
```

```
## ‘stat_bin()‘ using ‘bins = 30‘. Pick better value ‘binwidth‘.
```



Notice, all of the samples do not have the same looking histogram. This is because of random sampling error. All of the samples are coming from the same normal distributions, but random chance makes each sample a little bit different (e.g., you don't always get 5 heads and 5 tails when you flip a coin right)

Getting the means of the samples Now, let's look at the means of the samples, we will use `dplyr` to get the means for each sample, and put them in a table:

```r
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```r
sample_means <- my_df %>%
                group_by(samples) %>%
                summarise(means=mean(scores))

knitr::kable(sample_means)
```

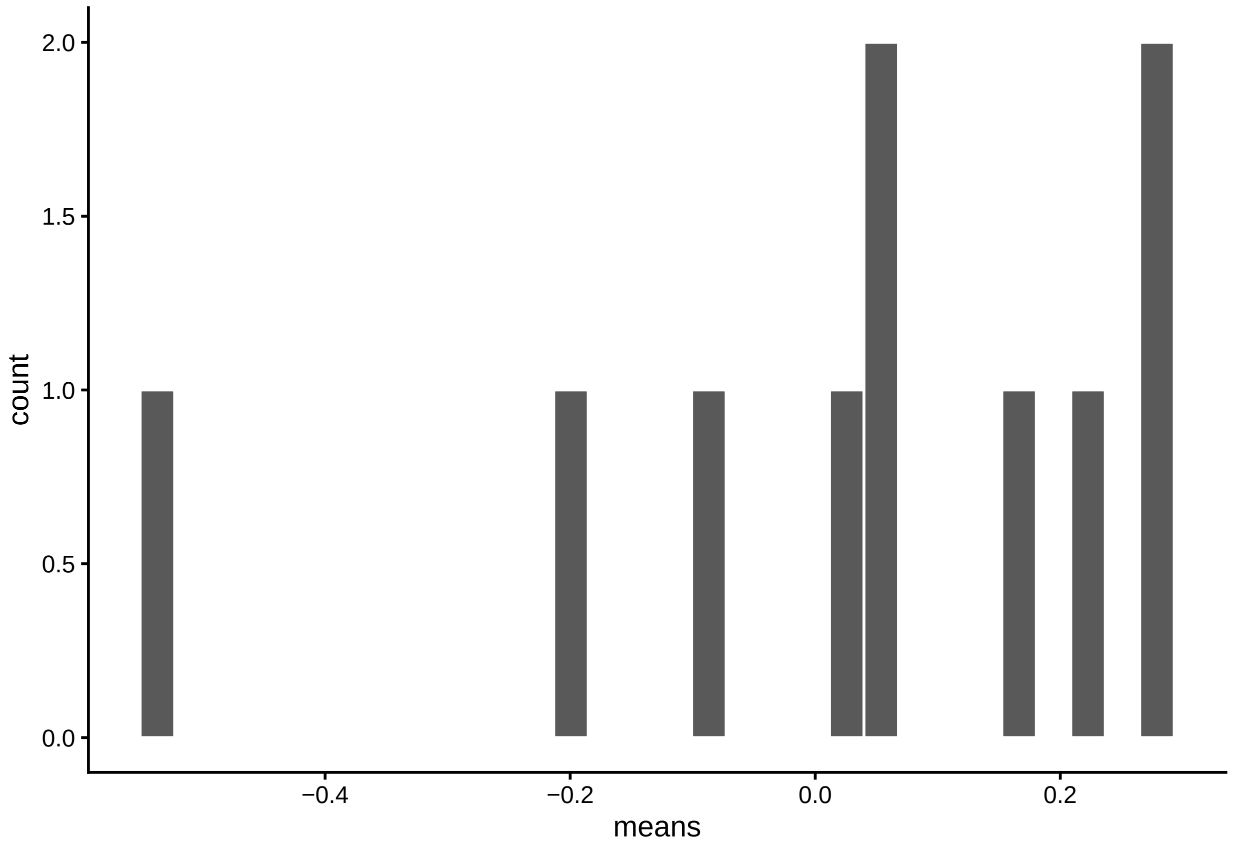
samples	means
1	0.2194361
2	0.2776398
3	-0.1996223
4	-0.5368882
5	0.1659198
6	-0.0791768
7	0.0672344
8	0.0452384
9	0.0343473
10	0.2788749

So, those are the means of our samples. What should the means be? Well, we would hope they are estimating the mean of the distribution they came from, which was 0. Notice, the numbers are all not 0, but they are kind of close to 0.

histogram for the means of the samples What if we now plot these 10 means (of each of the samples) in their own distribution?

```r
 ggplot(sample_means, aes(x=means))+
  geom_histogram(color="white")+
  theme_classic()
```

```
## `stat_bin()` using `bins = 30`. Pick better value `binwidth`.
```



That is the distribution of the sample means. It doesn't look like much eh? That's because we only took 10 samples right.

Notice one more thing... What is the mean of our 10 sample means? This is a mean of means. Remember that.

```
mean(sample_means$means)
```

```
## [1] 0.02730034
```

Well, that's pretty close to zero. Which is good. When we average over our samples, they better estimate the mean of the distribution they came from.

simulating the distribution of sample means Our histogram with 10 sample means looked kind of sad. Let's give it some more friends. How about we repeat our little sampling experiment 1000 times.

That is,... we take 1000 samples. Each sample takes 20 scores from a normal distribution (mean=0, SD=1). Then we find the means of each sample (giving us 1000 sample means). Then, we plot that distribution.

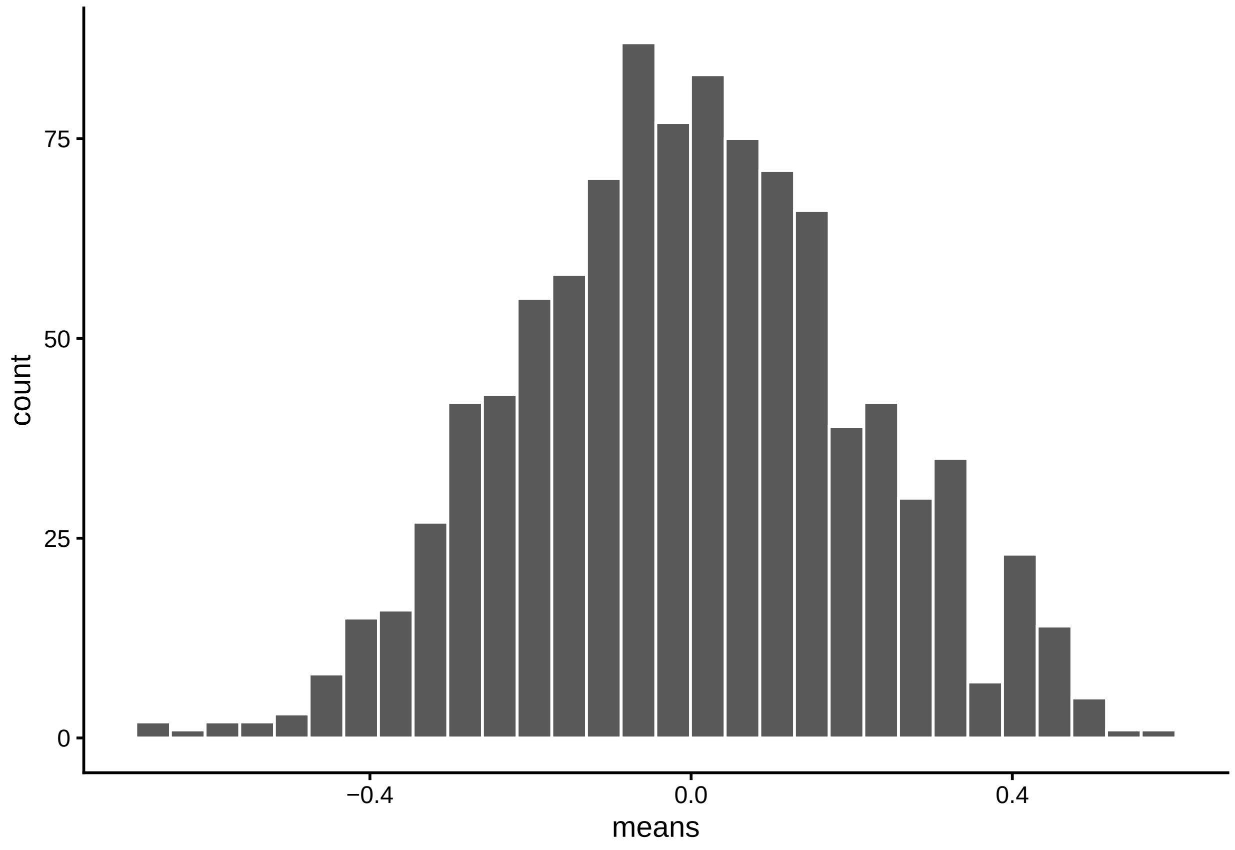
```
# get 1000 samples with 20 scores each
scores2 <- rnorm(1000*20,0,1)
samples2 <- rep(1:1000,each=20)
my_df2 <- data.frame(samples2,scores2)
head(my_df2)
```

Q4. Enter code for this scenario

```
##   samples2    scores2
## 1        1  0.3472993
## 2        1 -1.1338592
## 3        1  1.9757626
## 4        1  0.8103691
## 5        1  2.7898436
## 6        1  0.3664955
```

```
# get the means of the samples
sample_means2 <- my_df2 %>%
                group_by(samples2) %>%
                summarise(means=mean(scores2))
# make a histogram
ggplot(sample_means2, aes(x=means))+
  geom_histogram(color="white")+
  theme_classic()
```

```
## ‘stat_bin()‘ using ‘bins = 30‘. Pick better value ‘binwidth‘.
```



Q5. Describe what you observe: I see a huge histogram depicting the count of 1,000 sample means in total. It looks like it’s bell shaped.

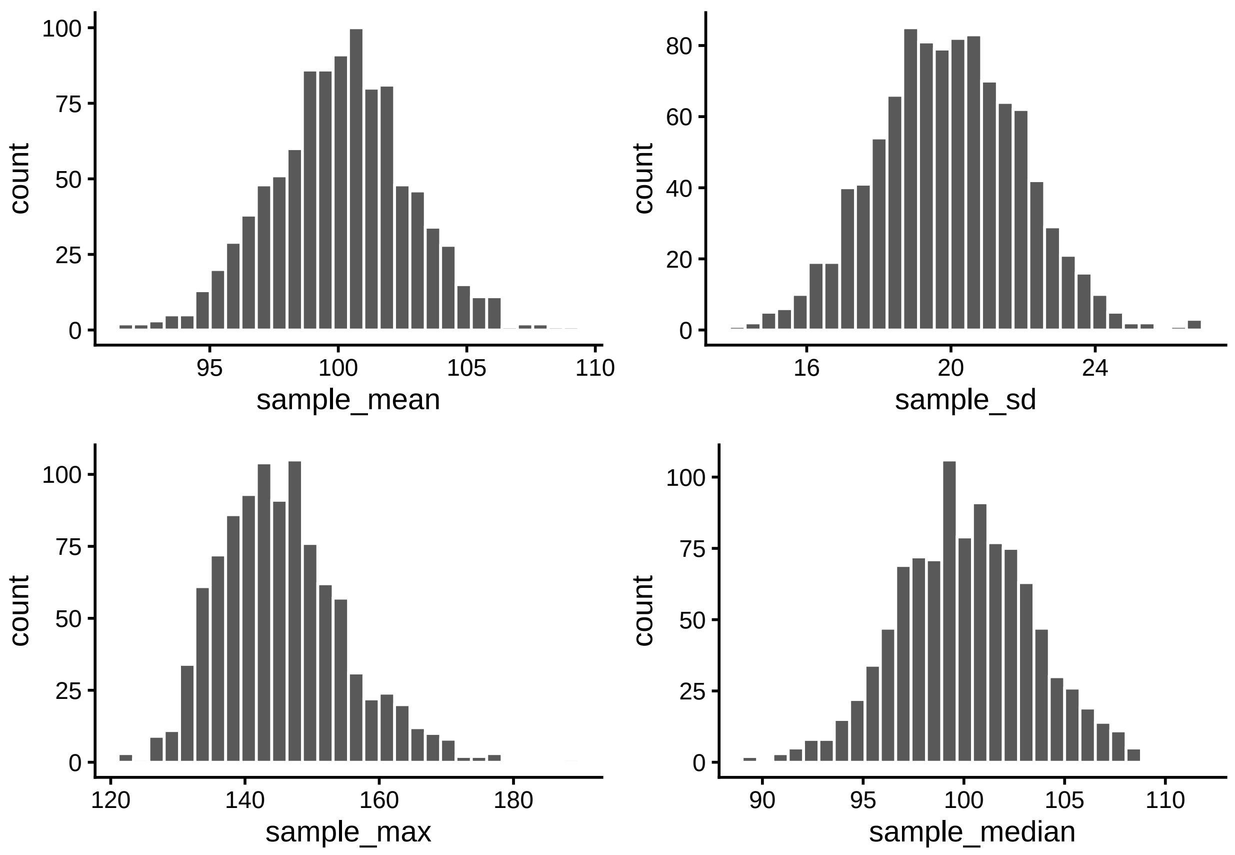
We will use things like the sampling distribution of the mean to make inferences about what chance can do in your data later on in this course.

Sampling distributions for any statistic

Just for fun here are some different sampling distributions for different statistics. We will take a normal distribution with mean = 100, and standard deviation =20. Then, we'll take lots of samples with n = 50 (50 observations per sample). We'll save all of the sample statistics, then plot their histograms. We do the sample means, standard deviations, maximum values, and medians. Let's do it.

```r
all_df<-data.frame()
for(i in 1:1000){
  sample<-rnorm(50,100,20)
  sample_mean<-mean(sample)
  sample_sd<-sd(sample)
  sample_max<-max(sample)
  sample_median<-median(sample)
  t_df<-data.frame(i,sample_mean,sample_sd,sample_max,sample_median)
  all_df<-rbind(all_df,t_df)
}

library(ggpubr)
a<-ggplot(all_df,aes(x=sample_mean))+
  geom_histogram(color="white")+
  theme_classic()
b<-ggplot(all_df,aes(x=sample_sd))+
  geom_histogram(color="white")+
  theme_classic()
c<-ggplot(all_df,aes(x=sample_max))+
  geom_histogram(color="white")+
  theme_classic()
d<-ggplot(all_df,aes(x=sample_median))+
  geom_histogram(color="white")+
  theme_classic()

ggarrange(a,b,c,d,
          ncol = 2, nrow = 2)
```

```
## `stat_bin()` using `bins = 30`. Pick better value `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value `binwidth`.
```



As we go through these exercises, you should be able to start thinking about why these sampling statistic distributions might be useful…

For now, just know that you can make a sampling statistic for pretty much anything in R, just by simulating the process of sampling, measuring the statistic, doing it over a bunch, and then plotting the histogram. This gives you a pretty good estimate of the distribution for that sampling statistic.

Central limit theorem

We have been building you up for the central limit theorem.

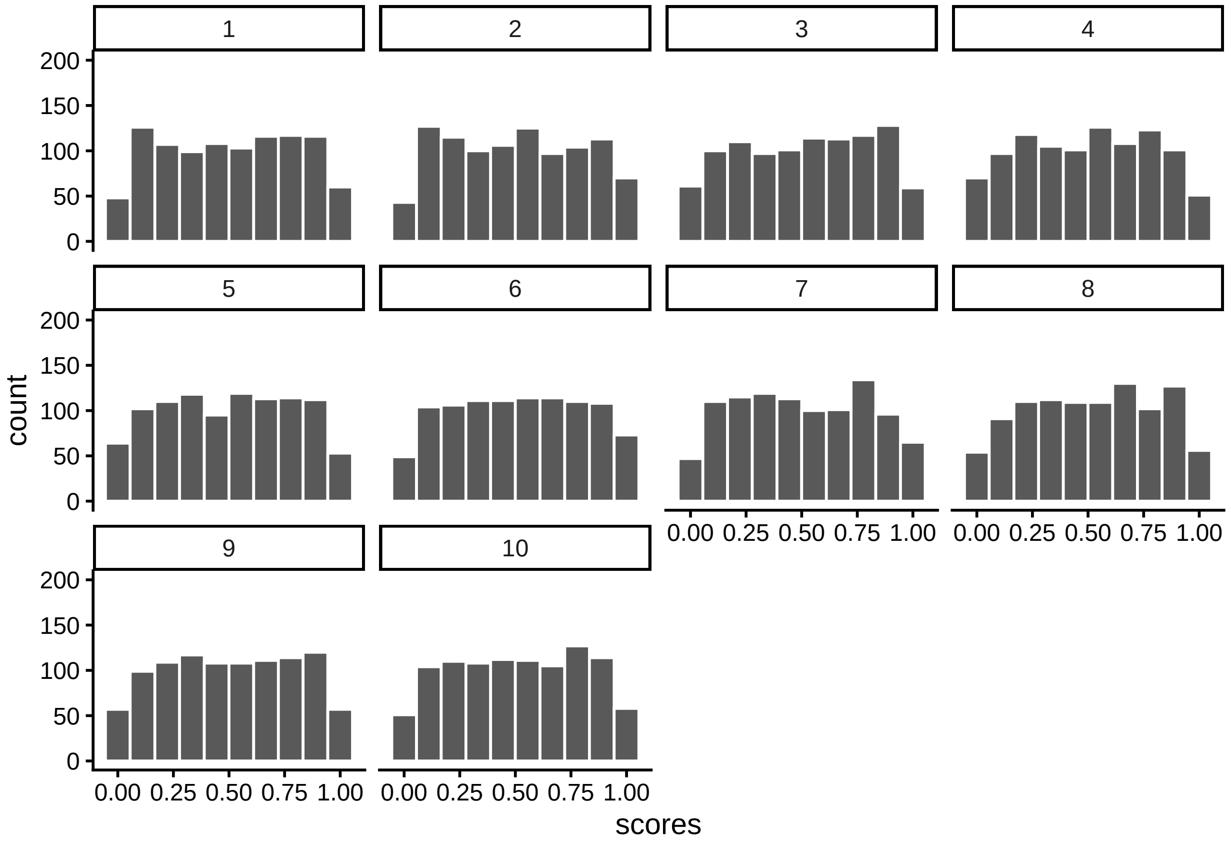
The central limit theorem is basically that the distribution of sample means will be a normal curve.

We already saw that before. But, the interesting thing about it, is that the distribution of your sample means will be normal, even if the distribution the samples came from is not normal. Huh what?

To demonstrate this the next bit of code is modified from what we did earlier. We create 100 samples. Each sample has 1000 observations. All of them come from a uniform distribution between 0 to 1. This means all of the numbers between 0 and 1 should occur equally frequently. Below I plot histograms for the first 10 samples (out of the 100 total, 100 is too many to look at). Notice the histograms are not normal, they are roughly flat.

```r
scores <- runif(100*1000,0,1)
samples <- rep(1:100,each=1000)
my_df <- data.frame(samples,scores)
```

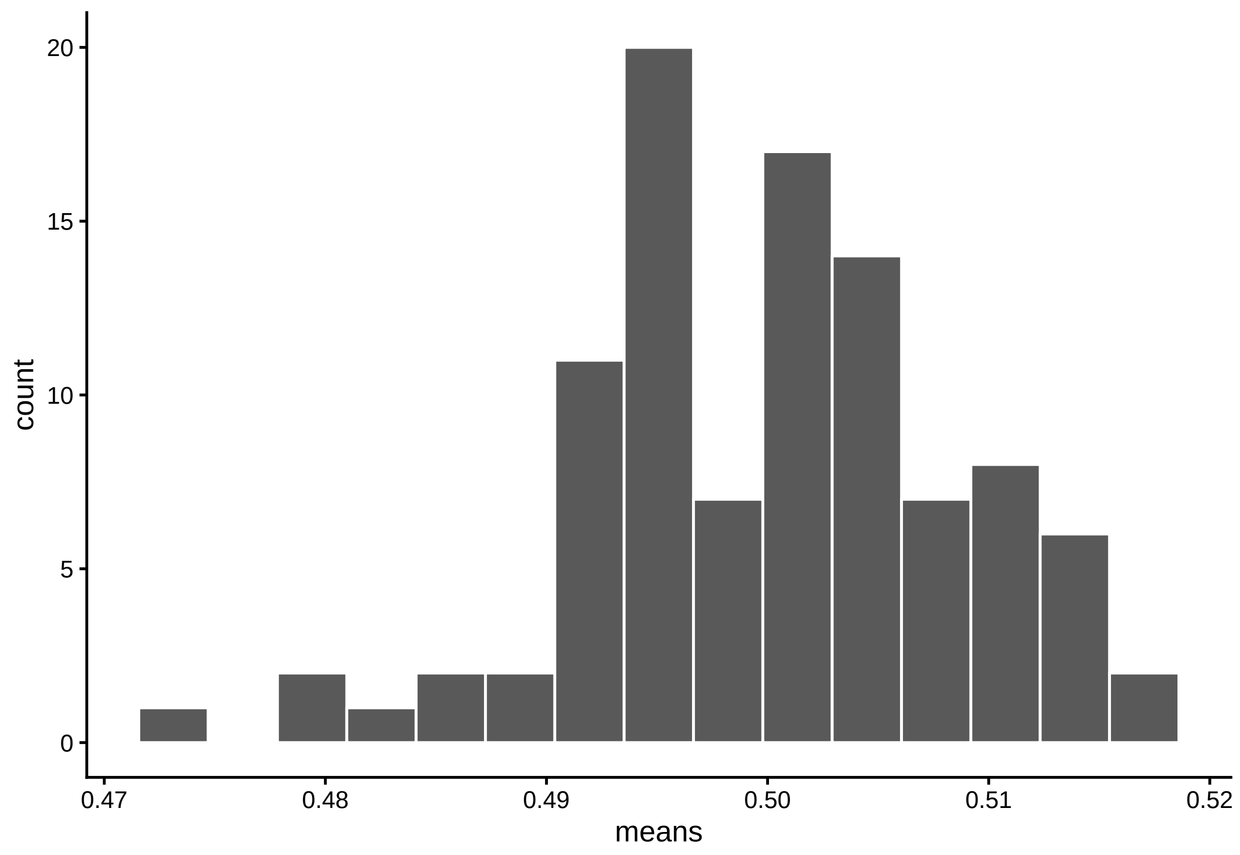
```r
ggplot(my_df[1:(10*1000),], aes(x=scores))+
  geom_histogram(color="white", bins=10)+
  facet_wrap(~samples)+
  theme_classic()+
  ylim(0,200)
```



We took samples from a flat uniform distribution, and the samples themselves look like that same flat distribution.

HOWEVER, if we now do the next step, and compute the means of each of our 100 samples, we could then look at the sampling distribution of the sample means. Let's do that:

```r
sample_means <- my_df %>%
                group_by(samples) %>%
                summarise(means=mean(scores))

# make a histogram

 ggplot(sample_means, aes(x=means))+
  geom_histogram(color="white", bins=15)+
  theme_classic()
```



As you can see, the sampling distribution of the sample means is not flat. It's shaped kind of normal-ish. If we had taken many more samples, found their means, and then looked at a histogram, it would become even more normal looking. Because that's what happens according to the central limit theorem.

The normal distribution

"Why does any of this matter, why are we doing this, can we stop now!!!!!! PLEEEEAASSEE, somebody HELP".

The reason the central limit theorem is important, is because researchers often take many samples, then analyse the means of their samples. That's what they do.

An experiment might have 20 people. You might take 20 measurements from each person. That's taking 20 samples. Then, because we know that samples are noisy. We take the means of the samples.

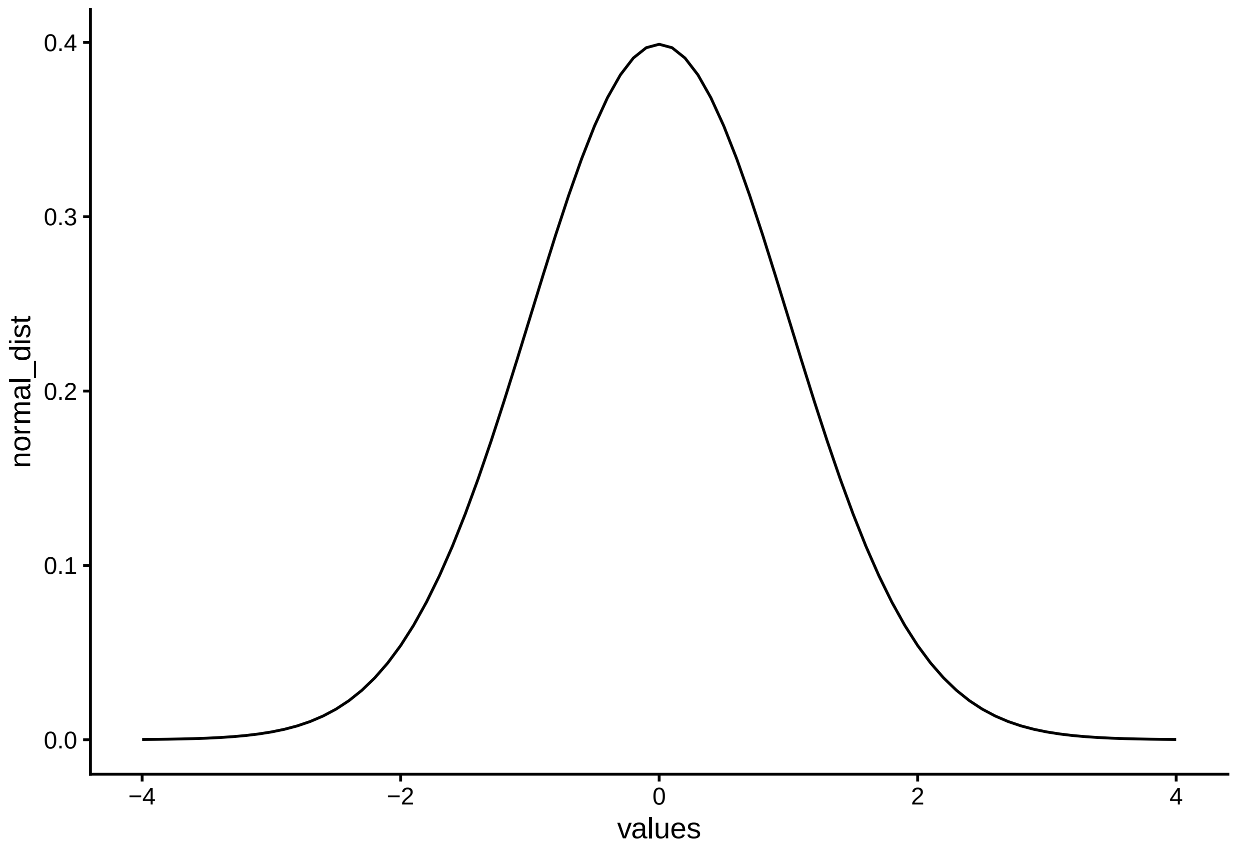
So, what researchers are often looking at, are means of samples. Not just the samples. And, now we know that means of samples (if we had a lot of samples), look like they are distributed normally (the central limit theorem says the should be).

We can use this knowledge. If we learn a little bit more about normal distributions, and how they behave and work, we can take that and use it to understand our sample means better. This will become more clear as head into the topic of statistical inference.

To continue the build-up we now look at some more properties of the normal distribution.

Graphing the normal distribution "Wait, I thought we already did that". We sort of did. We sampled numbers and made histograms that looked like normal distributions. But, a "normal distribution" is more of an abstract idea. It looks like this in the abstract:

```r
normal_dist <- dnorm(seq(-4,4,.1), 0, 1)
values <-seq(-4,4,.1)
normal_df <-data.frame(values,normal_dist)

ggplot(normal_df, aes(x=values,y=normal_dist))+
  geom_line()+
  theme_classic()
```



A really nice shaped bell-like thing. This normal distribution has a mean of 0, and standard deviation of 1. The heights of the lines tell you roughly how likely each value is. Notice, it is centered on 0 (most likely that numbers from this distribution will be near 0), and it goes down as numbers get bigger or smaller (so bigger or smaller numbers get less likely). There is a range to it. Notice the values don't go much beyond -4 and +4. This because those values don't happen very often. Theoretically any value could happen, but really big or small values have really low probabilities.

calculating the probability of specific ranges. We can use R to tell us about the probability of getting numbers in a certain range. For example, when you think about. It should be obvious that you have a 50% probability of getting the number 0 or greater. Half of the distribution is 0 or greater, so you have a 50% probability.

We can use the `pnorm` function to confirm this:

```r
pnorm(0, mean = 0, sd= 1, lower.tail=FALSE)
```

```
## [1] 0.5
```

Agreed, `pnorm` tells us the probability of getting 0 or greater is .5.

```
pnorm(2, mean = 0, sd= 1, lower.tail=FALSE)
```

Q6. Well, what is the probability of getting a 2 or greater?

```
## [1] 0.02275013
```

```
ps<-pnorm(c(-1,1), mean = 0, sd= 1, lower.tail=FALSE)
ps[1]-ps[2]
```

Q7. What is the probability of getting a score between -1 and 1? Does this seem familiar?

```
## [1] 0.6826895
```

What about the numbers between 1 and 2?

```
ps<-pnorm(c(1,2), mean = 0, sd= 1, lower.tail=FALSE)
ps[1]-ps[2]
```

```
## [1] 0.1359051
```

About 13.5% of numbers fall in that range.

How about between 2 and 3?

```
ps<-pnorm(c(2,3), mean = 0, sd= 1, lower.tail=FALSE)
ps[1]-ps[2]
```

```
## [1] 0.02140023
```

Again a very small amount, only 2.1 % of the numbers, not a a lot.

summary pnorm You can always use `pnorm` to figure how the probabilities of getting certain values from any normal distribution. That's great.

Mnemonic:

rnorm –> r is for random –> used to generate random samples from a normal distribution

pnorm –> p is for probability –> used to find the probability of values in a normal distribution