Advanced Statistics

author: Bernhard Angele date: Class 1, October 2nd, 2014

What is advanced about these statistics?

- Goal is for you to understand the principles, not just the steps.
- Simulation approach:
- If you don't know how something about a statistical test, simulate it!
- Example questions you might ask:
  - What is the power of this test?
  - What happens if I violate the normality assumption for an ANOVA?
  - What happens if I don't correct for multiple comparisons?

How do I run simulations?

- Not very easy in SPSS
- Very easy in R
- You do have to learn how to communicate with R, though
- Don't worry, it's not as hard as you may think
- R is a programming language with a command line interface
- You write the commands, R does the work
- The R language is not horribly difficult, but there are some quirks.

Interacting with R (1)

- Easiest way: use RStudio
- But there are still some common issues that people get confused about:
- The command prompt:

```
>
```

  - This is where you type your commands.
  - What if the prompt looks like this?

```
+
```

- R thinks that the previous command isn't finished yet. (Maybe you forgot a closing parenthesis?)
- If you don't want to finish it, press ESC

Interacting with R (2)

- When working with R, you create and interact with **objects**
- All **objects** live in the **workspace**
- You can get a list of all the objects in the workspace by typing `ls()`

- You can delete an object (e.g. an object named `example_object`) by typing `rm(example_object)` (replace `example_object` with the name of the object you want to delete)
- You can combine these two commands in order to delete everything in the workspace: `rm(list = ls())`
- This is useful when you want to start over from scratch.
- It also demonstrates how you can nest commands: You're telling `rm` to use the output of `ls()` as the list of all the objects that should be deleted.

Interacting with R (3)

- The R interpreter makes some basic assumptionss.
- For example, if you tell it the name of an object, it will print its contents.

```
pi
```

```
## [1] 3.142
```

What went wrong here?

```
rm
```

```
## function (..., list = character(), pos = -1, envir = as.environment(pos),
##     inherits = FALSE)
## {
##     dots <- match.call(expand.dots = FALSE)$...
##     if (length(dots) && !all(sapply(dots, function(x) is.symbol(x) ||
##         is.character(x))))
##         stop("... must contain names or character strings")
##     names <- sapply(dots, as.character)
##     if (length(names) == 0L)
##         names <- character()
##     list <- .Primitive("c")(list, names)
##     .Internal(remove(list, envir, inherits))
## }
## <bytecode: 0x2830cc0>
## <environment: namespace:base>
```

What went wrong on the last slide?

- Answer: if you type the name of a command (or rather, a **function**) R prints its contents.
- In the case of a **function**, it will give you its definition
- If you want the function to do something (i.e. you want to **execute** it), you need to call it like this:

```
rm(this_object_does_not_exist)
```

```
## Warning: object 'this_object_does_not_exist' not found
```

- What went wrong now?
- `rm` is telling me that the object I wanted to delete doesn't exist (duh).

Errors and warnings

- Errors:
- mean that something critical went wrong and the command was not executed
- Warnings:
- mean that there was a less critical issue
- the command was still executed (but maybe didn't do what you thought it would do)
- What's the worst kind of problem?
- Error?
- Warning?
- No, it's when R does something you didn't intend to do but doesn't warn you!

Assignment

- You can create new objects from existing ones:

```
x <- 1
y <- "meow"
z <- ls
x
```

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

```
y
```

```
## [1] "meow"
```

```
z()
```

```
## [1] "x" "y" "z"
```

Let's do some maths

Addition

```
1+1
```

```
## [1] 2
```

Subtraction

```
1-1
```

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

More maths

Multiplication and division

```
4*3
```

```
## [1] 12
```

```
12/4
```

```
## [1] 3
```

Even more maths

Powers

```
5^2
```

```
## [1] 25
```

```
2^3
```

```
## [1] 8
```

Variables

```
x <- 5
x + 1
```

```
## [1] 6
```

Commands

```
x <- 5
x + 1
```

```
## [1] 6
```

```
x
```

```
## [1] 5
```

Is anyone surprised by this?

Functions

```r
addOne <- function(x) {
  x+1
}
addOne(5)
```

```
## [1] 6
```

```r
addOne(-3)
```

```
## [1] -2
```

Types of data

```r
x <- 1
y <- "test"
z <- c(1,2)
z
```

```
## [1] 1 2
```

z is a **vector**

```r
z + 1
```

```
## [1] 2 3
```

Vector operations

```r
z <- c(1,2,3,4,5)
z + 2
```

```
## [1] 3 4 5 6 7
```

```r
z - 1
```

```
## [1] 0 1 2 3 4
```

```r
z * 2
```

```
## [1]  2  4  6  8 10
```

More vector operations

```r
z <- c(1,2,3,4,5)
sum(z)
```

```
## [1] 15
```

```r
length(z)
```

```
## [1] 5
```

```r
sum(z)/length(z)
```

```
## [1] 3
```

Descriptive statistics

We could define a new function that calculates the mean. But maybe it's defined for us already?

```r
z <- c(1,2,3,4,5)
mean(z)
```

```
## [1] 3
```

What about other descriptive statistics?

```r
median(z)
```

```
## [1] 3
```

```r
sd(z)
```

```
## [1] 1.581
```

```r
var(z)
```

```
## [1] 2.5
```

Summary: lots of interesting descriptive statistics at once

```r
summary(z)
```

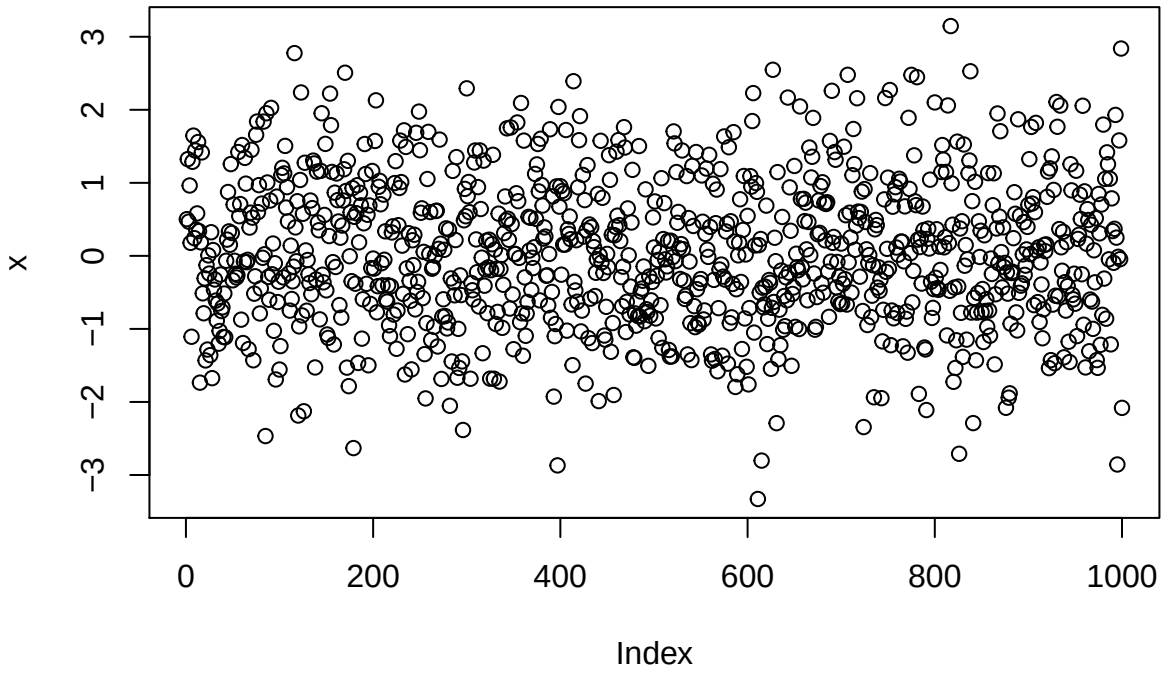
```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##       1       2       3       3       4       5
```

Let's simulate some data

```
x<-rnorm(1000)
head(x)
```

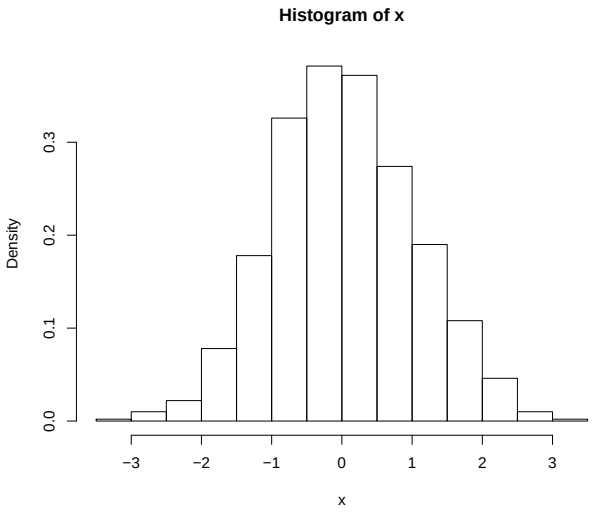
```
## [1]  0.5027  1.3244  0.4688  0.9636  0.1782 -1.1071
```

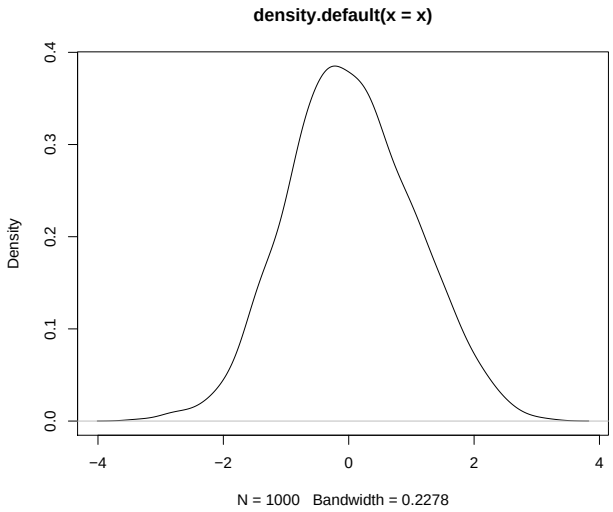
```
plot(x)
```



Distribution of the simulated data (histogram)

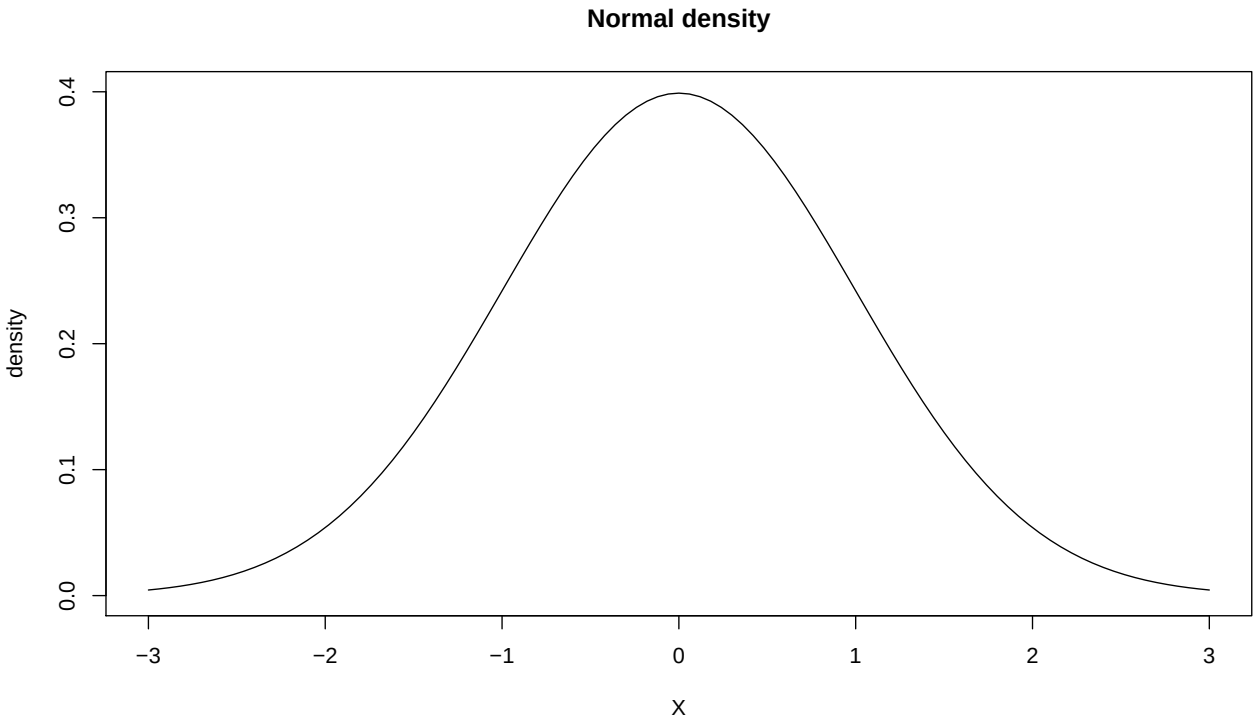
```
## plot density histogram:
par(mfrow=c(1,2)) # (little trick: two plots side-by-side)
hist(x,freq=F)
plot(density(x))
```





Probability density?

```
plot(function(x) dnorm(x), -3, 3,
main = "Normal density",ylim=c(0,.4),
ylab="density",xlab="X")
```

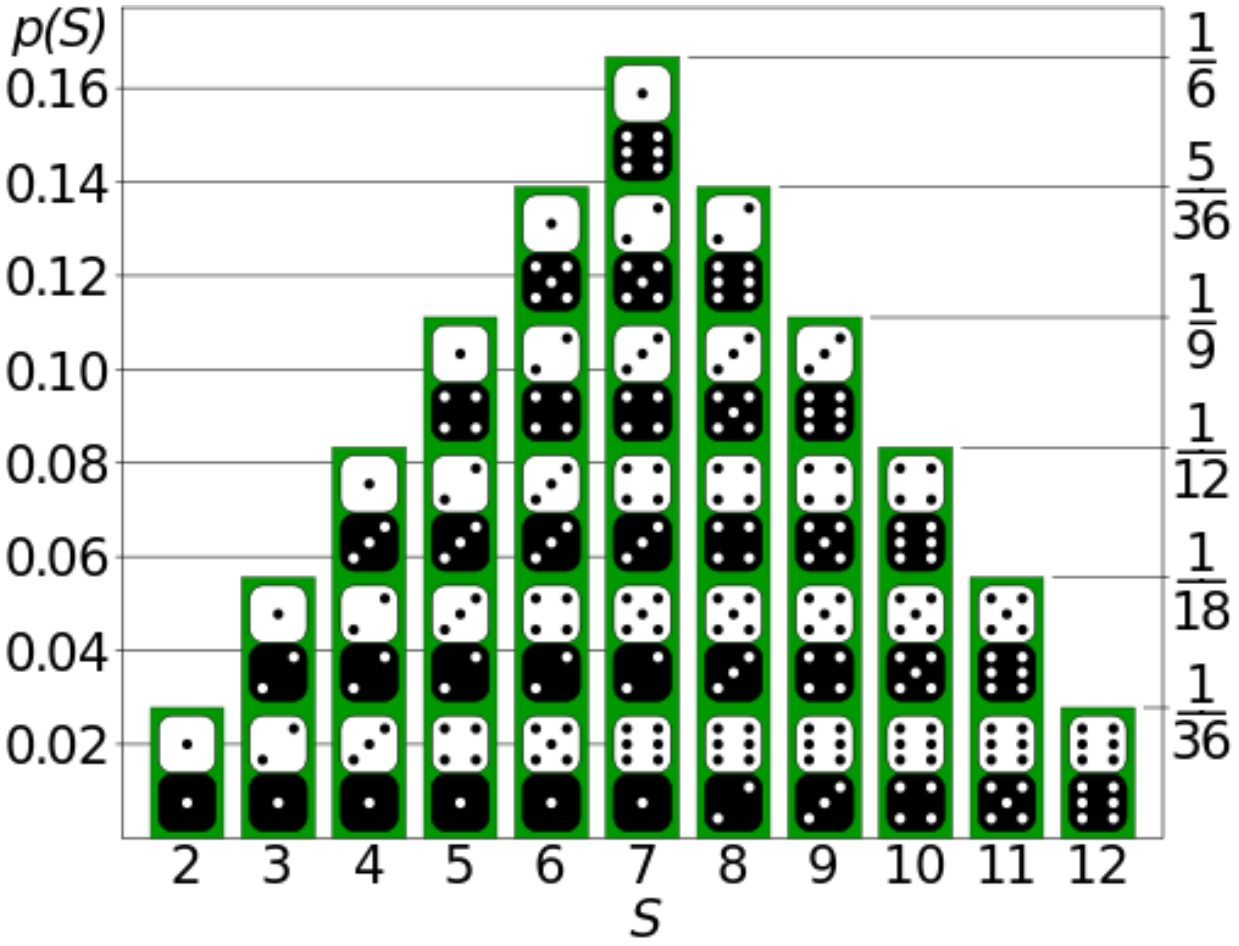


What is a probability distribution?

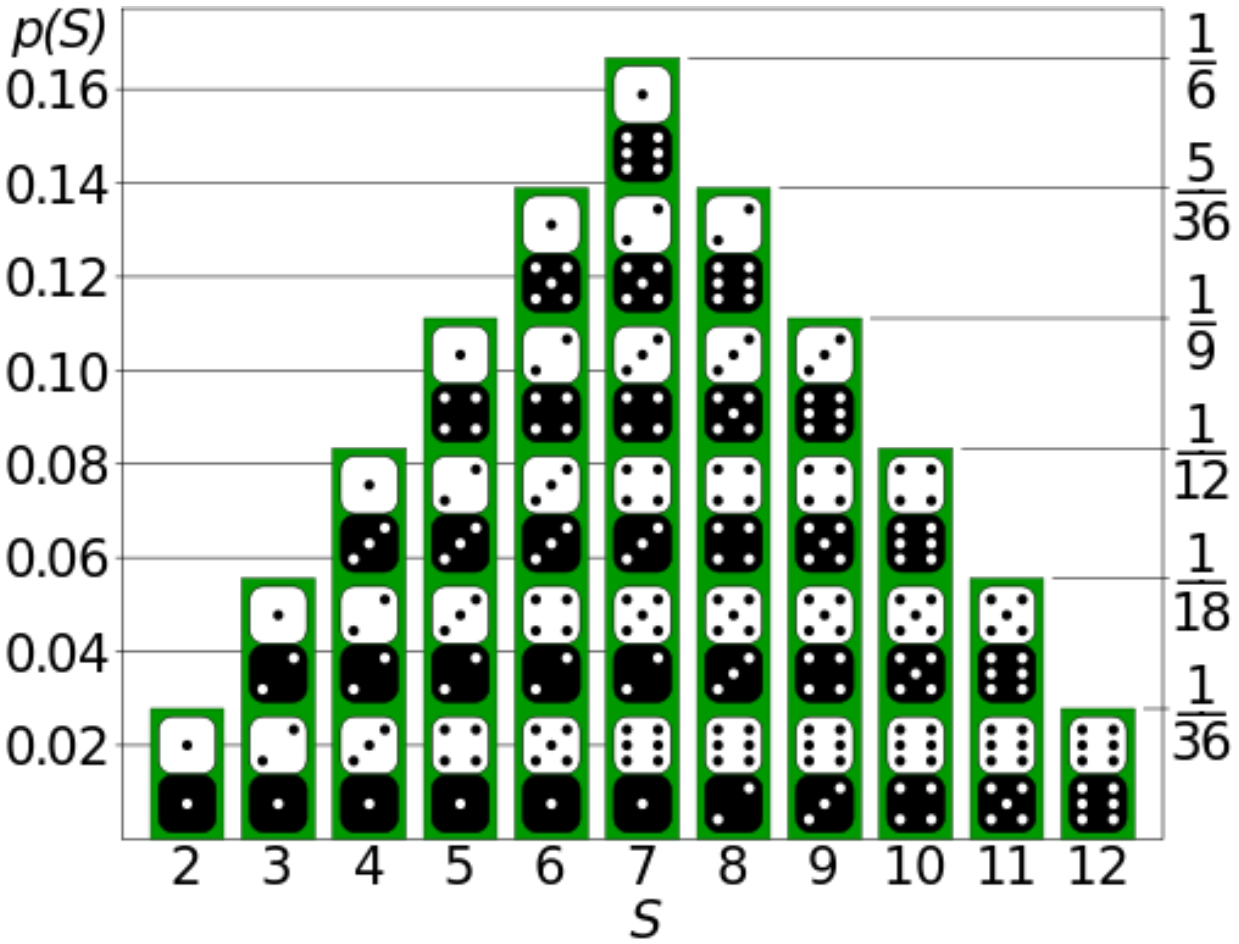
From Wikipedia: In probability and statistics, a probability distribution assigns a probability to each measurable subset of the possible outcomes of a random experiment, survey, or procedure of statistical

inference.

Here's an example of a discrete probability distribution:



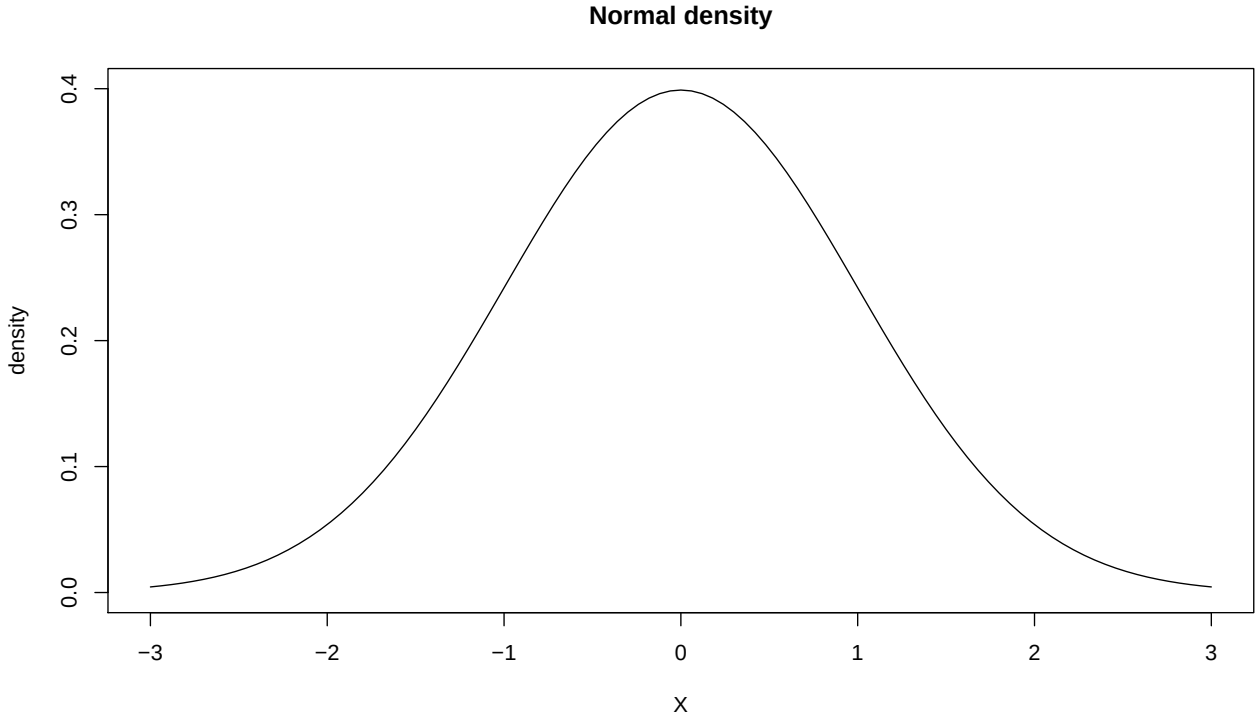
Discrete probability distribution



Every possible outcome (sum of the numbers rolled on two dice) is assigned a corresponding probability. This is called a *probability mass function*.

Important: all values sum to 1.

Continuous probability distribution

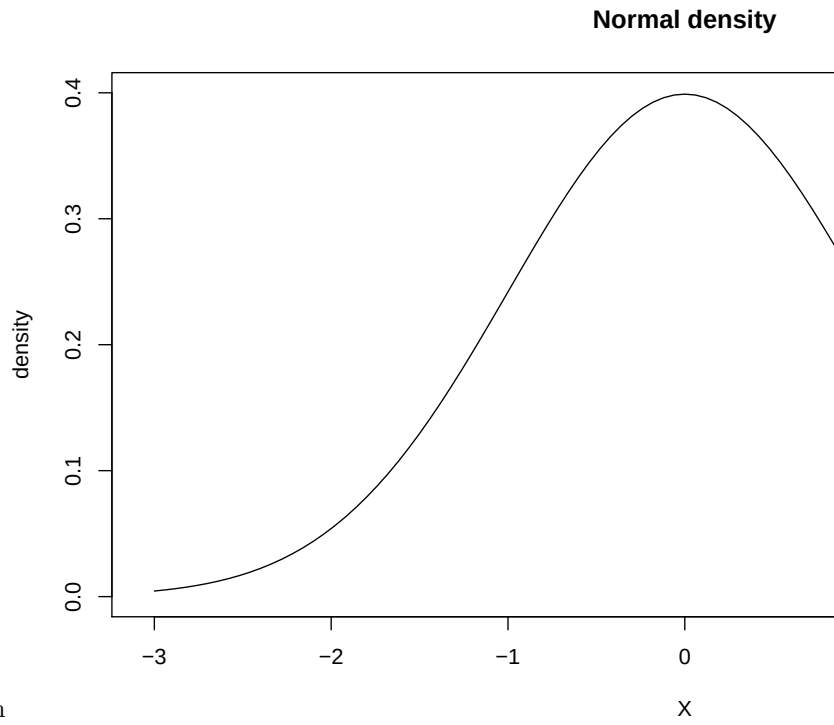


Here, the outcomes are continuous, so it doesn't make sense to ask about the probability of any point on the x-axis.

- What is the probability of x = 1?
- What do you mean by "1"? Does 1.00001 still qualify as 1?
- It makes more sense to ask these questions about intervals.
- Important: the total area under the curve is 1.

Normal probability density function (PDF)

$$f(x, \mu, \sigma) = \frac{1}{\sigma\sqrt{2\pi}} e^{-((x-\mu)^2/2\sigma^2)}$$

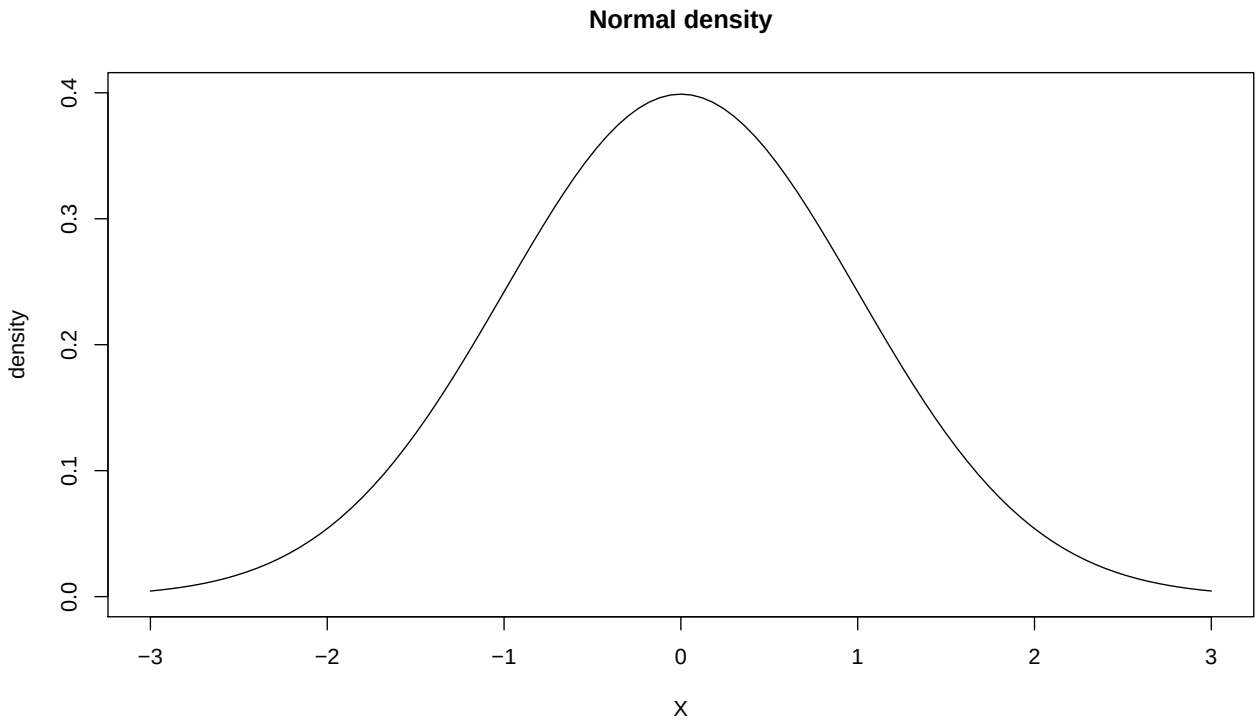


With $x =$ value, $\mu =$ mean, and $\sigma =$ standard deviation

Defining the normal PDF by hand (just in case you wanted to make sure)

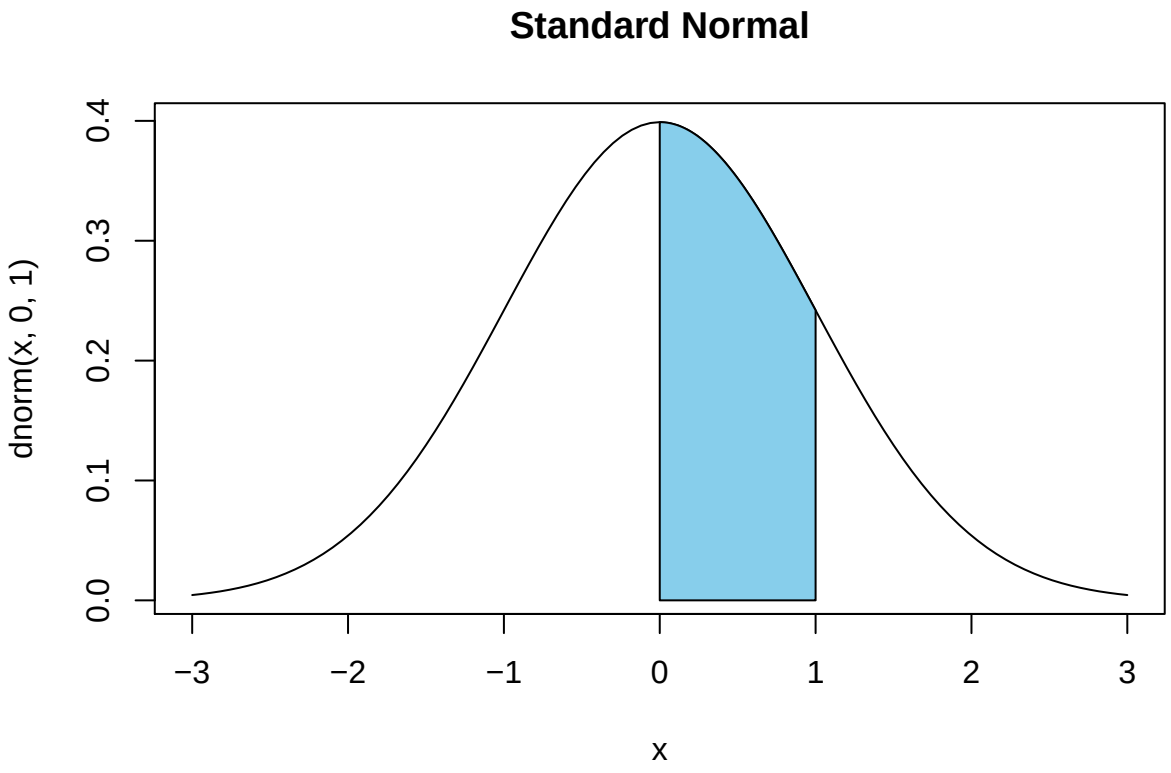
$$f(x, \mu, \sigma) = \frac{1}{\sigma\sqrt{2\pi}} e^{-((x-\mu)^2/2\sigma^2)}$$

```
dnorm_manual <- function(x, mu = 0, sigma = 1) {1/(sigma*sqrt(2*pi)) * exp(-((x-mu)^2/2*sigma^2))}
plot(function(x) dnorm_manual(x), -3, 3,
main = "Normal density",ylim=c(0,.4),
ylab="density",xlab="X")
```



Asking reasonable questions about continuous distributions

- What's the probability of x being between 0 and 1?



- How do we do this?

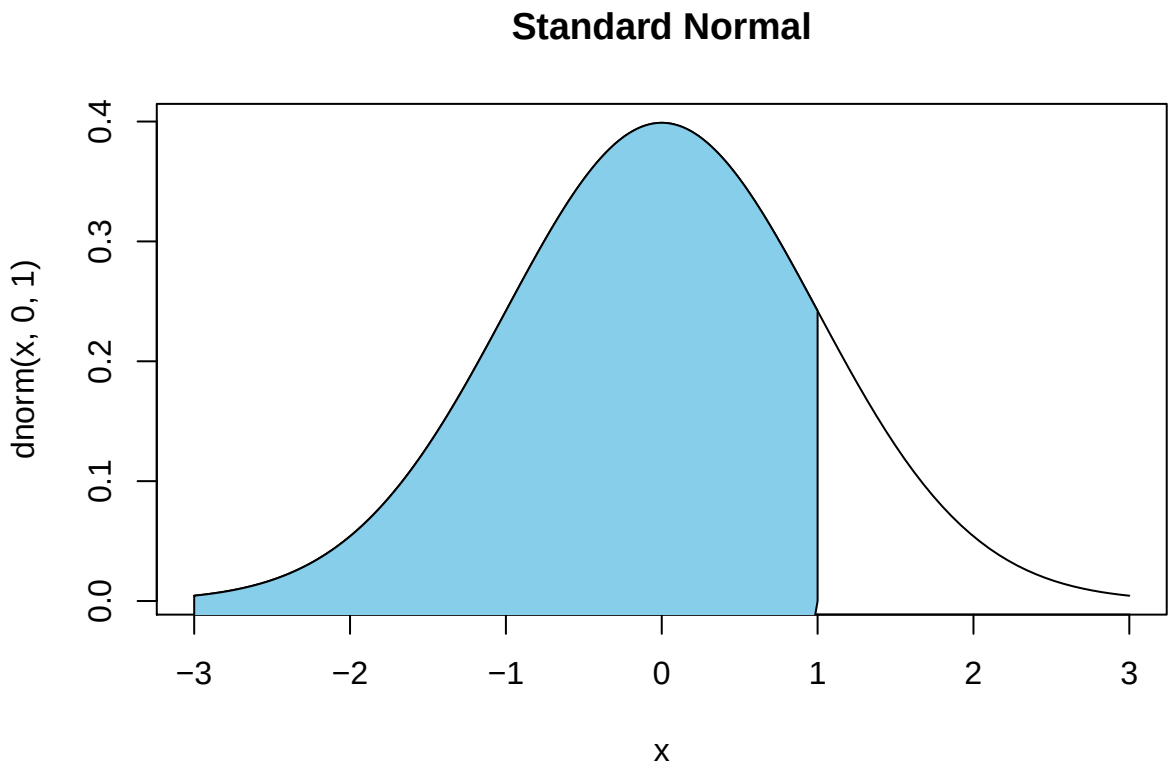
- We could integrate!
- But there's a more convenient way in R.

Using `pnorm`

- `pnorm(q)` gives you the probability of `x < q` (in the standard normal distribution)

```
pnorm(1)
```

```
## [1] 0.8413
```



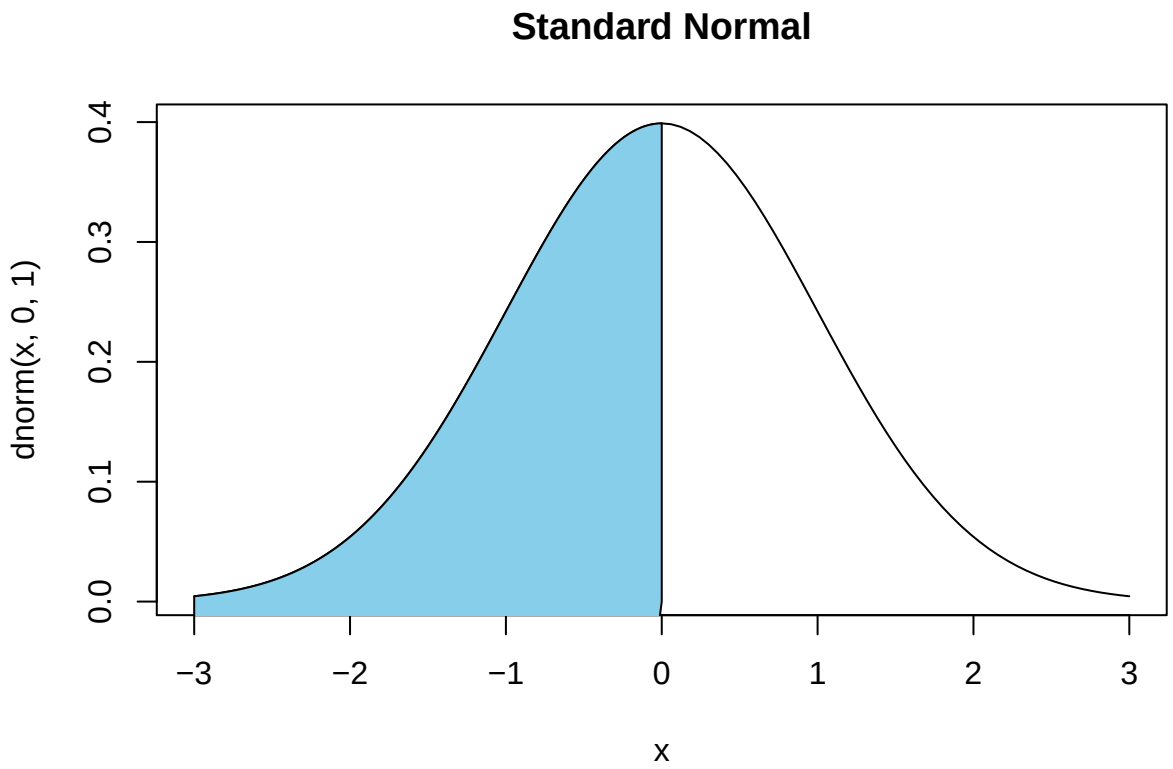
`pnorm(1)` is not quite what we wanted yet.

Using `pnorm` (2)

- What about `pnorm(0)`?

```
pnorm(0)
```

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



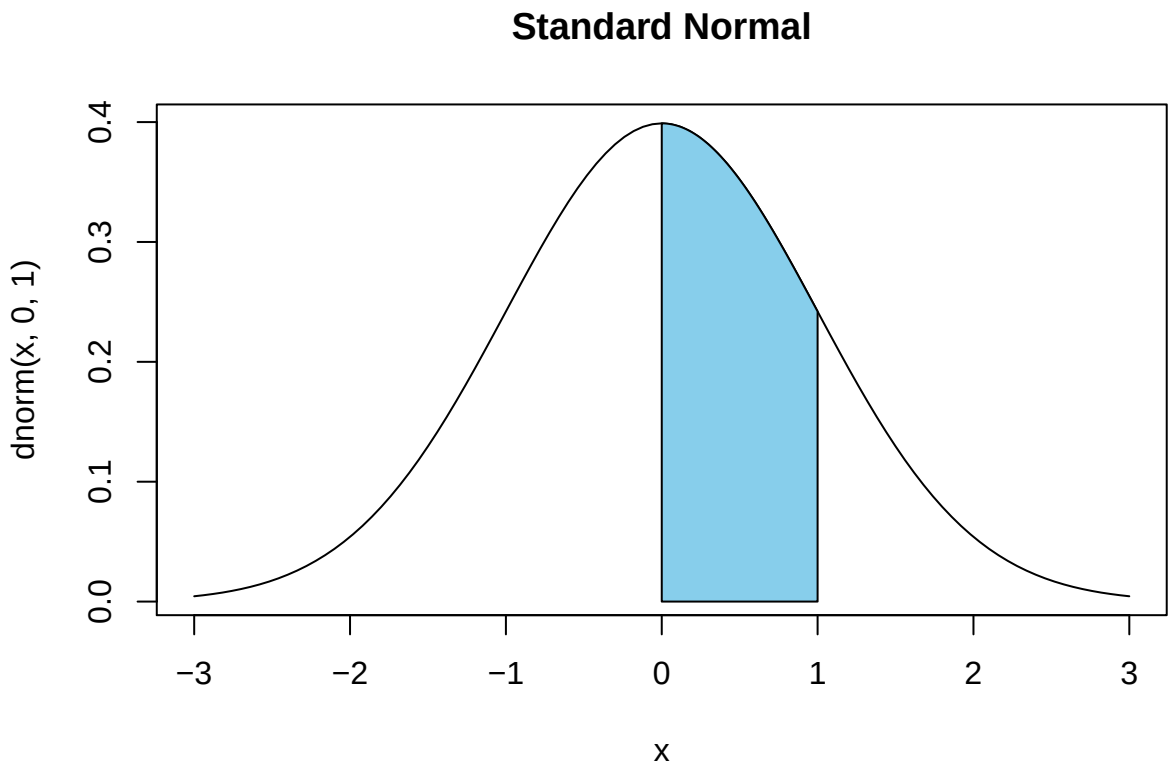
Well, look at that!

Finally getting our area under the curve

- I hope you can guess how to do it now:

```
pnorm(1)-pnorm(0)
```

```
## [1] 0.3413
```



- Success!

Things you can do with this knowledge

- Say I'm looking at random numbers from a standard normal distribution:

```
rnorm(5)
```

```
## [1] -0.07928  0.69067  0.02099 -0.78822  1.20904
```

- and I see that one of them is 4.
- That seems very unusual
- Just how unusual?
  - What's the probability of getting a value of 4 when sampling from a standard normal distribution (mean = 0, sd = 1)?

Just how unusual is a value of 4?

- Remember, when you have a continuous distribution, you can't think about point values (e.g. 5). Rather, what you want to know is:
  - What is the probability of getting a value of 4 *or greater*?

```
print(pnorm(4), digits = 5)
```

```
## [1] 0.99997
```

- Is that the probability? No, that's the probability of getting a value of 4 *or less*
- Just get the inverse probability:

```
1 - pnorm(4)
```

```
## [1] 3.167e-05
```

Why do we care about the normal distribution?

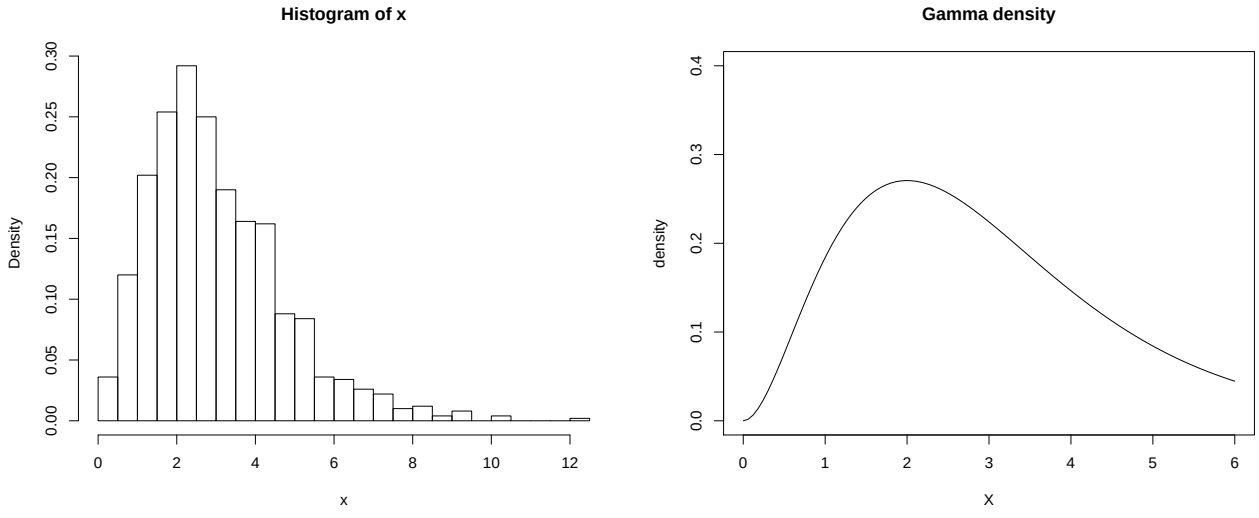
- Central limit theorem (CLT)

  When sampling from a population that has a mean, provided the sample size is large enough, the sampling distribution of the sample mean will be close to normal regardless of the shape of the population distribution

Let's see if that's actually true by running some simulations!

Non-normal distributions: Gamma

```
## plot density histogram:
x<-rgamma(n = 1000, shape = 3)
par(mfrow=c(1,2)) # (two plots side-by-side)
hist(x,freq=F, breaks = 20)
plot(function(x) dgamma(x, shape = 3), 0, 6,
main = "Gamma density",ylim=c(0,.4),
ylab="density",xlab="X")
```



Sampling from a gamma distribution (1)

```r
sample_size <- 100
number_of_simulations <- 1000

sample_means <- replicate(number_of_simulations, mean(rgamma(n = sample_size, shape = 3)))

mean(sample_means)
```

```
## [1] 2.993
```

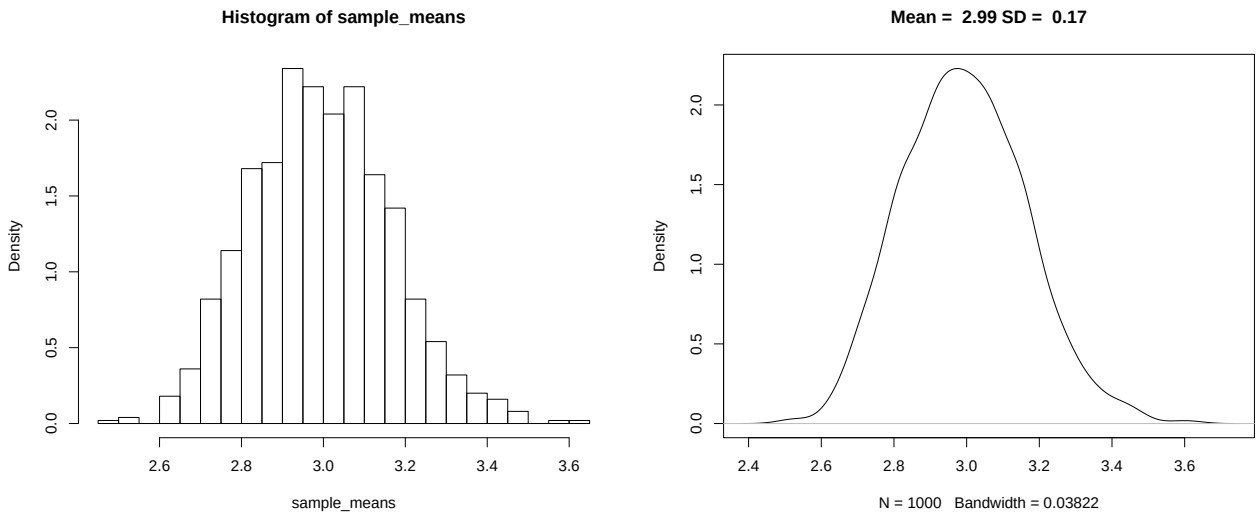
```r
sd(sample_means)
```

```
## [1] 0.1691
```

Sampling from a gamma distribution (2)

Make a function to combine histogram and plot:

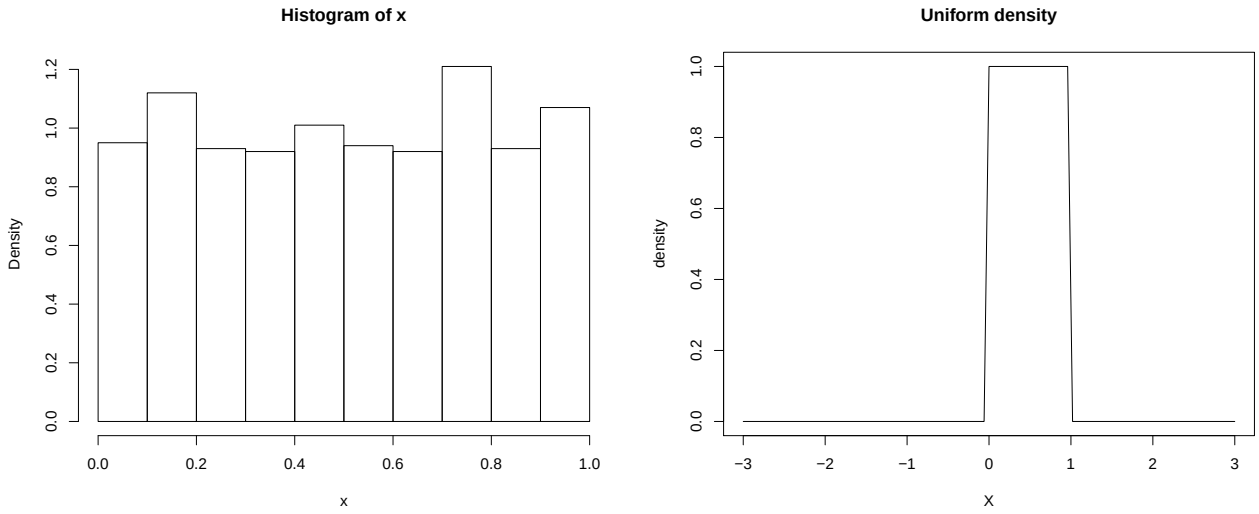
```r
make_hist_and_plot <- function(sample_means){
  par(mfrow=c(1,2)) # (two plots side-by-side)
  hist(sample_means,freq=F, breaks = 30)
  plot(density(sample_means), main = paste("Mean = ", round(mean(sample_means),2) ,
                                           "SD = ", round(sd(sample_means),2)))
} # label the plot with mean and sd
make_hist_and_plot(sample_means)
```



Non-normal distributions: Uniform

```r
## plot density histogram:
x <- runif(n = 1000)
par(mfrow=c(1,2)) # (two plots side-by-side)
```

```r
hist(x,freq=F)
plot(function(x) dunif(x), -3, 3,
main = "Uniform density",ylim=c(0,1),
ylab="density",xlab="X")
```



Sampling from a uniform distribution (1)

```r
sample_size <- 100
number_of_simulations <- 1000

sample_means <- replicate(number_of_simulations, mean(runif(n = sample_size)))

mean(sample_means)
```

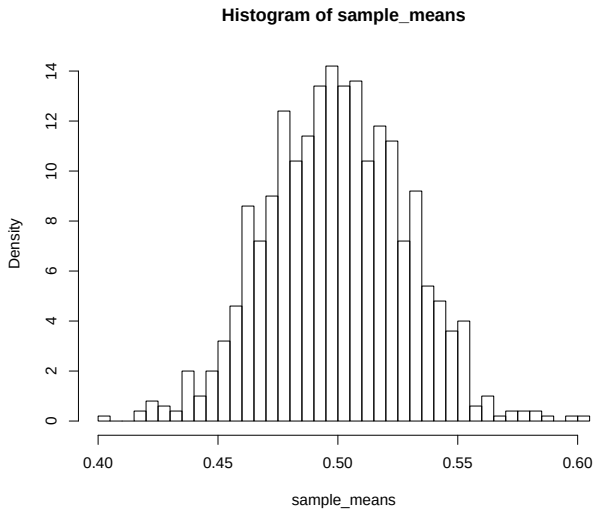
```
## [1] 0.4995
```

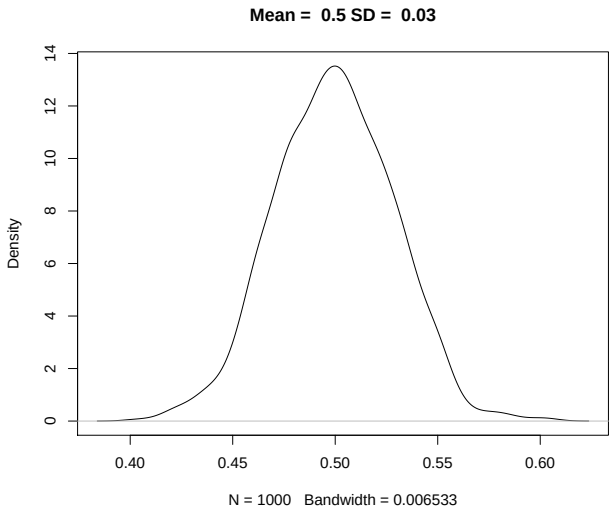
```r
sd(sample_means)
```

```
## [1] 0.0289
```

Sampling from a uniform distribution (2)

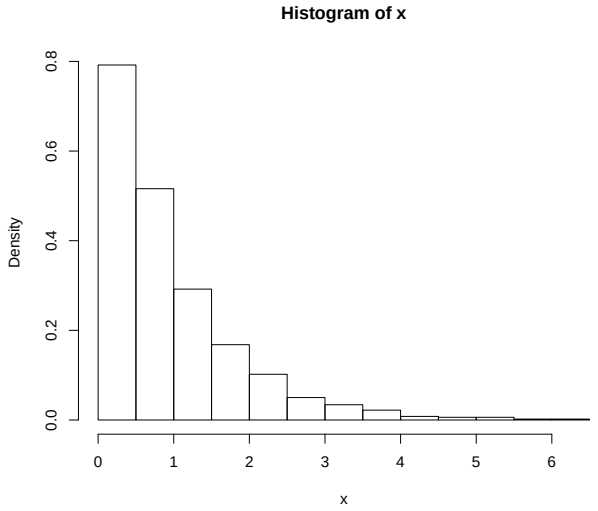
```r
make_hist_and_plot(sample_means)
```

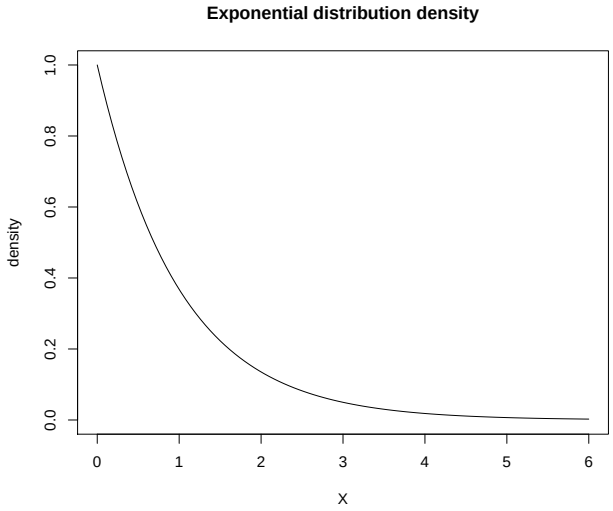




Non-normal distributions: Exponential

```
## plot density histogram:
x <- rexp(n = 1000)
par(mfrow=c(1,2)) # (two plots side-by-side)
hist(x,freq=F)
plot(function(x) dexp(x), 0, 6,
main = "Exponential distribution density",ylim=c(0,1),
ylab="density",xlab="X")
```





Sampling from an exponential distribution (1)

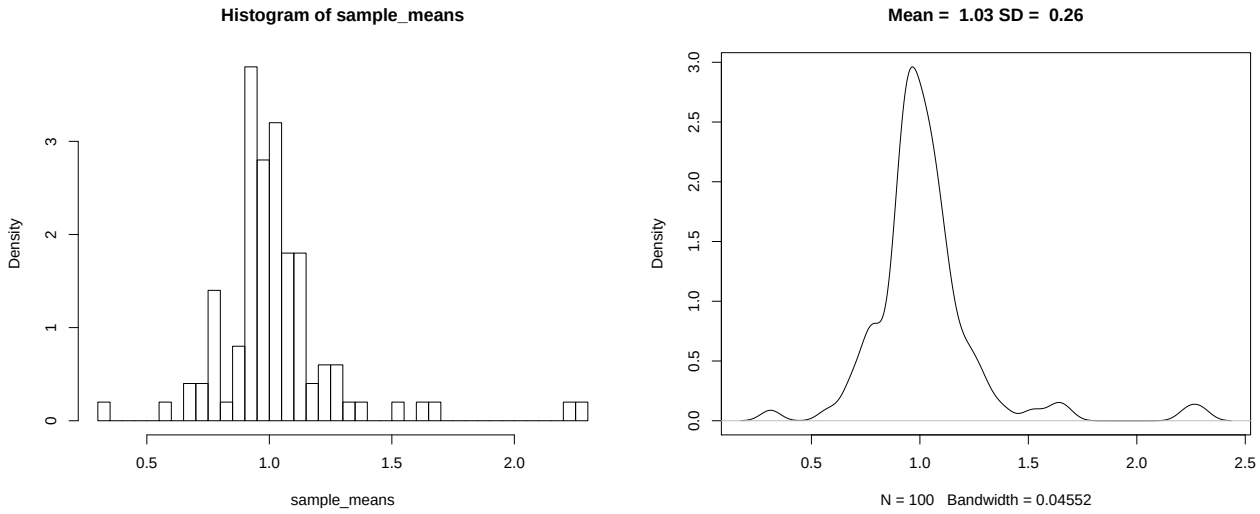
```
number_of_simulations <- 100

sample_mean_from_exp<- function(number_of_samples){
```

```
  samples <- rexp(number_of_samples)
  mean(samples)
}

sample_means <- sapply(1:number_of_simulations, sample_mean_from_exp)
summary(sample_means)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   0.312   0.917   0.988   1.030   1.090   2.300
```

Sampling from an exponential distribution (2)

```
make_hist_and_plot(sample_means)
```



“‘

The sampling distribution of the mean (1)

- Notice something about the means of the samples?
- They seem to cluster around the population mean
- Let’s play with this some more.
- First, make a function for running the simulations so that we don’t have to type all the code from the previous slides again and again
  - We’re going to sample from the normal distribution again since it’s easy to specify mean and sd for it.

The sampling distribution of the mean (2)

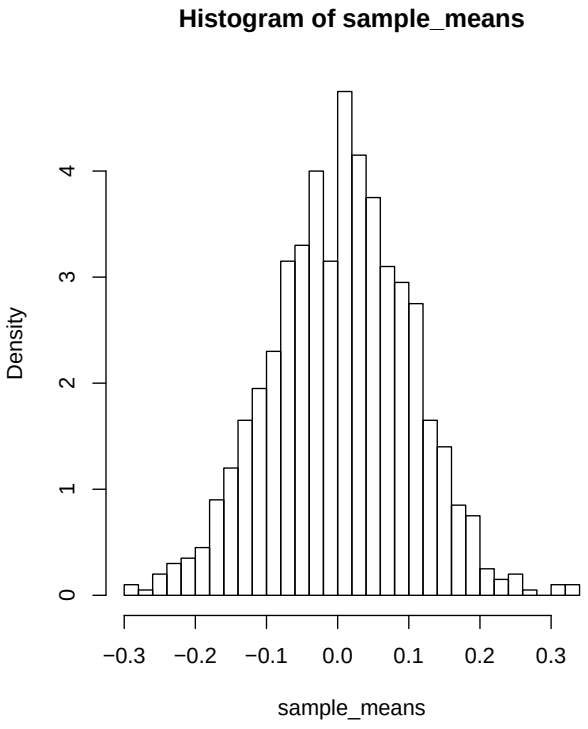
This is our convenience function for running the simulations. Don’t worry if you don’t understand everything yet!

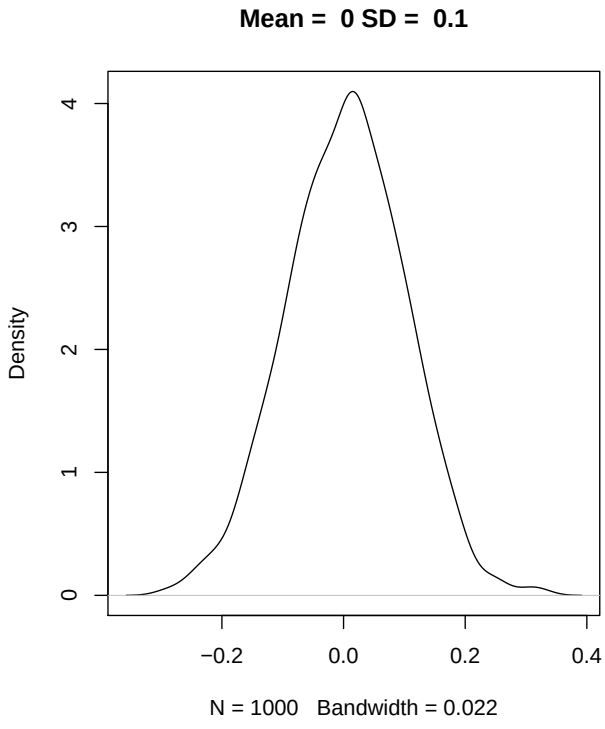
```
run_simulation <- function(sample_size = 100,
                           number_of_simulations = 1000,
                           population_mean = 0,
                           population_sd = 1)
  {

sample_means <- replicate(number_of_simulations,
                          mean(rnorm(n = sample_size,
                                     mean = population_mean,
                                     sd = population_sd)))

make_hist_and_plot(sample_means)
}
```

The sampling distribution of the mean (2)

It works:

```
run_simulation(sample_size = 100,
               number_of_simulations = 1000,
               population_mean = 0,
               population_sd = 1)
```

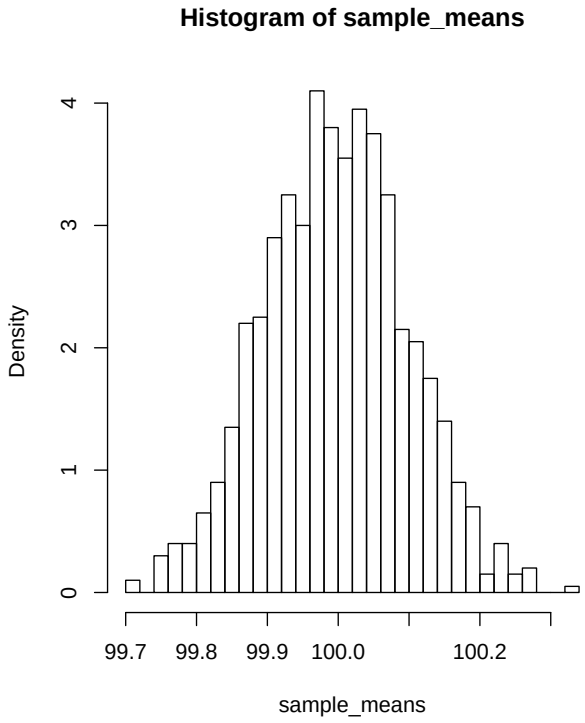


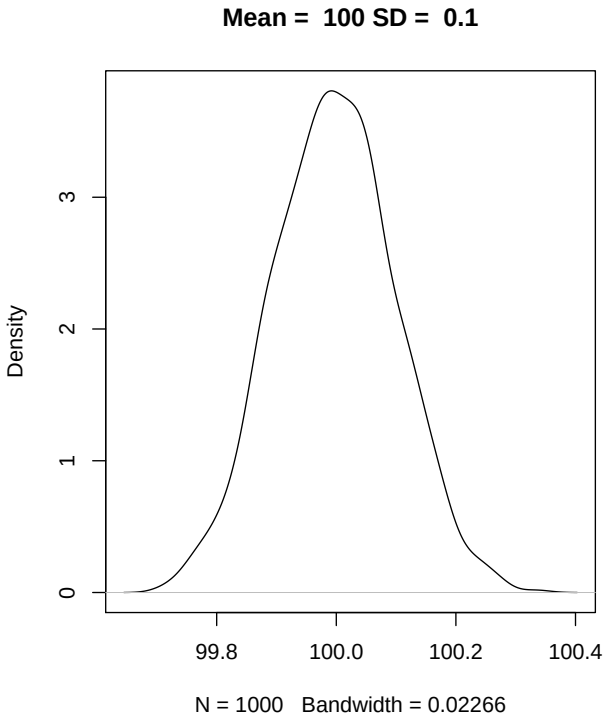


The sampling distribution of the mean (3)

Now, let's try different parameters. What happens if we change the mean of the population?

```
run_simulation(sample_size = 100,
               number_of_simulations = 1000,
               population_mean = 100,
               population_sd = 1)
```

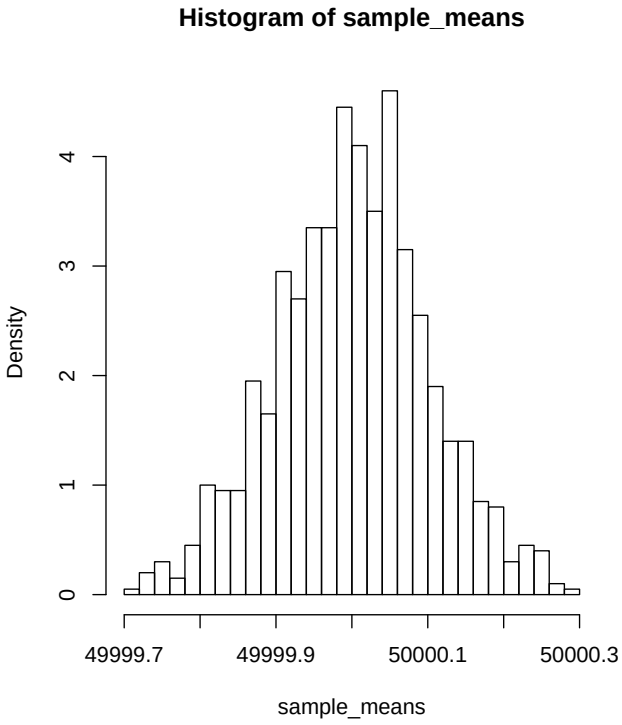


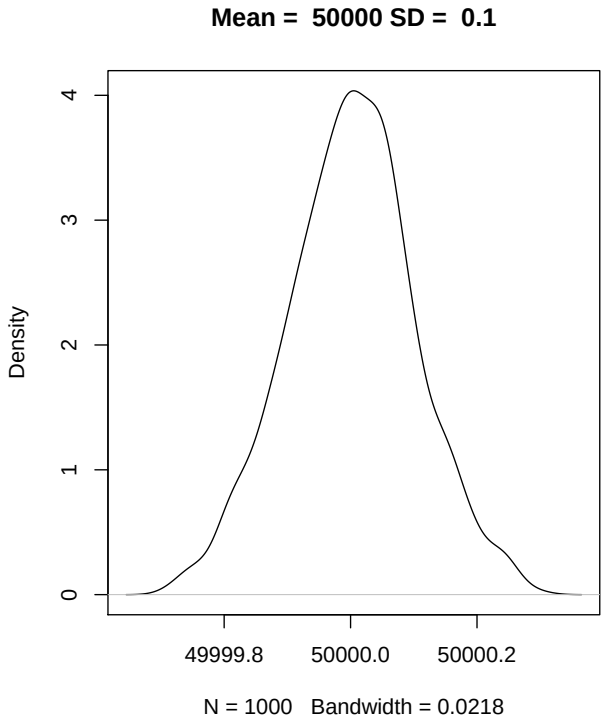


The sampling distribution of the mean (3)

The sampling distribution of the mean has the same mean as the population! You can (hopefully) see how this might be useful.

```
run_simulation(sample_size = 100,
               number_of_simulations = 1000,
               population_mean = 50000,
               population_sd = 1)
```



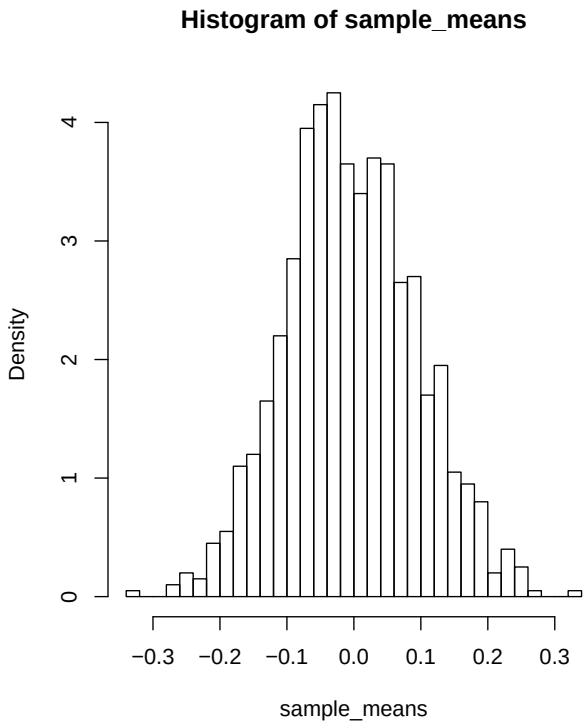


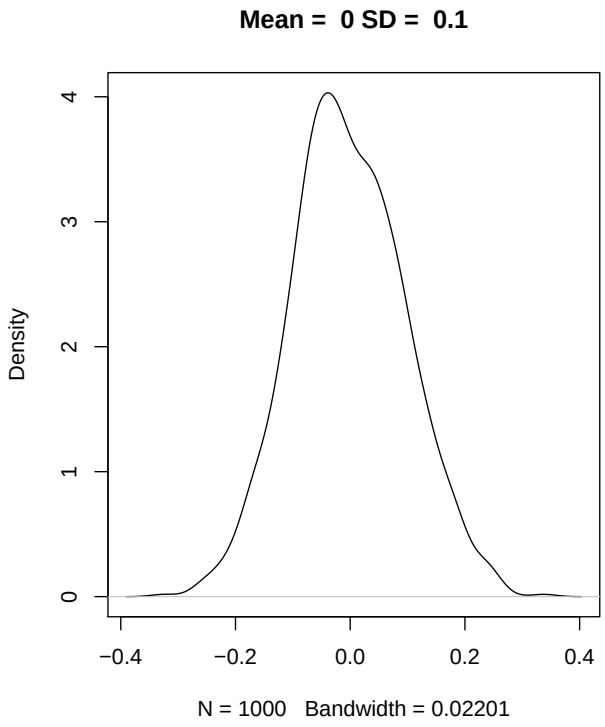
Sample mean and population mean (1)

- OK, *why* is it useful?
- Well, usually, we don't know the population mean.
- We have to estimate it somehow.
- Solution: just use the sample mean as an estimator for the population mean.
  - Can this go wrong?
  - Yes, definitely.

Sample mean and population mean (2)

- Remember the plots we just made:



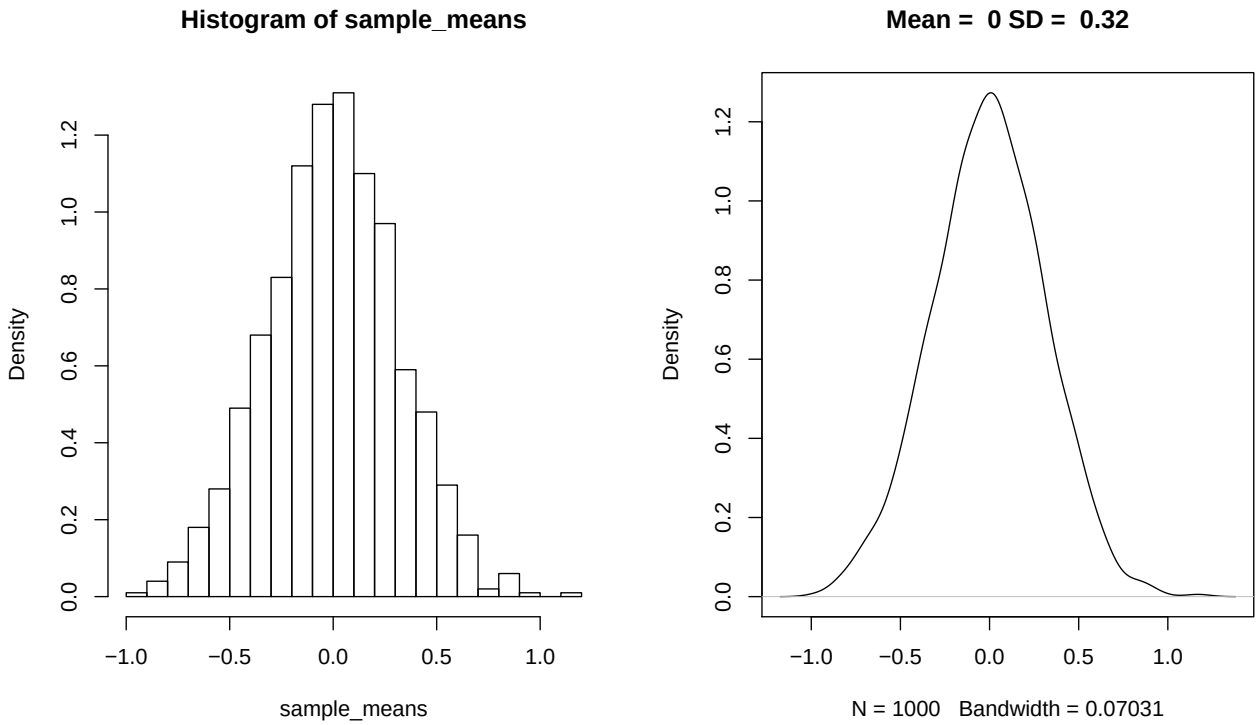


- Notice that the sample mean is not **not always** the same as the population mean (0 in this case).
- This is due to the random nature of drawing a sample from the population.

Sample mean and population mean (3)

- Let's try reducing the sample size

```
run_simulation(sample_size = 10,
               number_of_simulations = 1000,
               population_mean = 0,
               population_sd = 1)
```

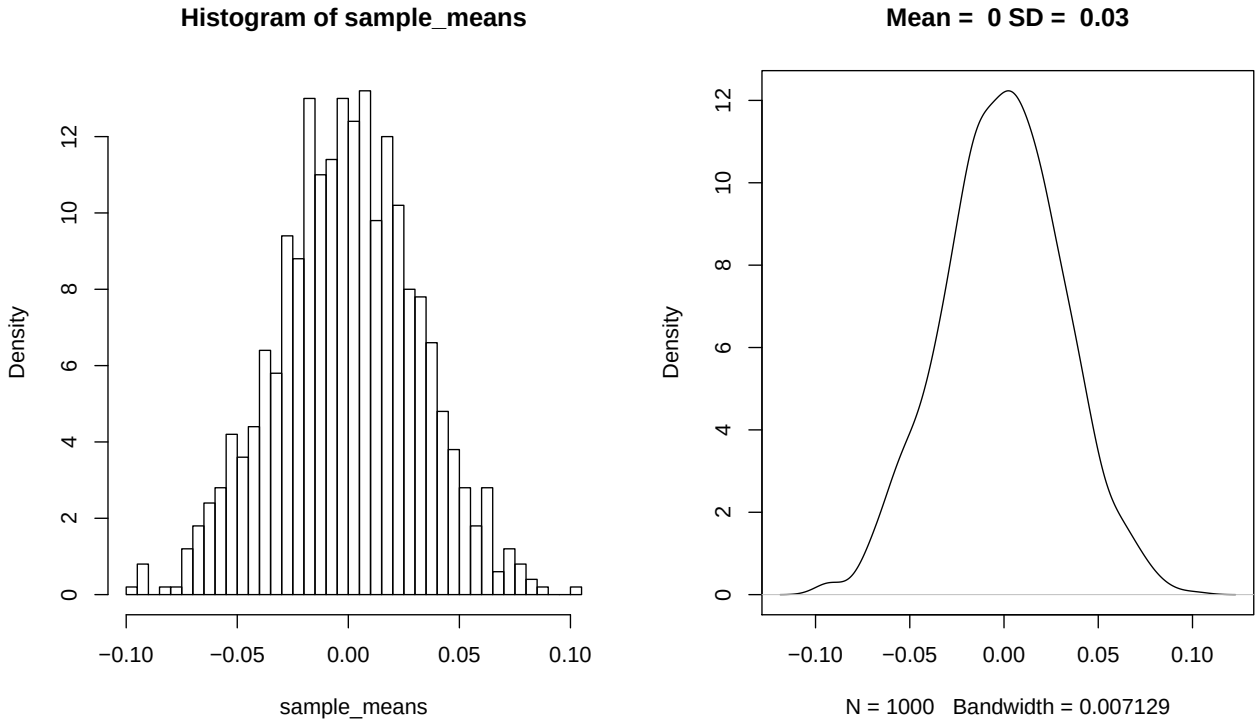


- Things got a bit noisier (note that the x-axis is scaled automatically) - The sd of the distribution of the sample means went up.

Sample mean and population mean (4)

- Let's try increasing the sample size

```
run_simulation(sample_size = 1000,
               number_of_simulations = 1000,
               population_mean = 0,
               population_sd = 1)
```



- Things got a lot less noisy (note that the x-axis is scaled automatically) - The sd of the distribution of the sample means went down.

Sample mean and population mean (5)

- Note that when we changed the sample size, the sd of the distribution of sample means changed.
- The mean stayed the same though!
- The larger your sample is, the closer your average sample mean is going to be to the true population mean.
- Formally speaking, the sample mean is an **unbiased estimator** of the population mean.
- I could show you the mathematical proof for that, but I won't.

Standard error of the mean (SE) (1)

Population SD = 1

Sample size	SD of the sample mean
10	.31
100	.1
1000	.03

See a pattern?

This relationship holds in general:

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

where $\sigma_{\bar{x}}$ is the standard deviation of the sample means, σ is the population standard deviation, and n is the sample size

Standard error of the mean (SE) (2)

- Usually, we don't know what σ is, but we can still can estimate $\sigma_{\bar{x}}$ from our sample:

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

where $SE_{\bar{x}}$ is the estimated **standard error of the mean**, s is the population standard deviation, and n is the sample size. - More about that next week.