STATS 250 Lab 06

Simulation

Nick Seewald

nseewald@umich.edu

Week of 10/05/2020

Reminders 💡

Your tasks for the week running Friday 10/2 - Friday 10/9:

Task	Due Date	Submission
M-Write 1 Final Revision	Wednesday 10/7	Canvas
Homework 5	Friday 10/9 8AM ET	course.work
Lab 5	Friday 10/9 8AM ET	Canvas

Stop by office hours! You can attend anyone's -- not just mine!

M-Write office hours schedule on Canvas (see MWrite Info on home page)

Homework 4 Comments

- Statistics is not a branch of math. It is a *mathematical science.*
  - In statistics, it's important that we tie our conclusions back to data.

Homework 4 Comments

- Statistics is not a branch of math. It is a *mathematical science.*
  - In statistics, it's important that we tie our conclusions back to data.
- context context context context context

Homework 4 Comments

- Statistics is not a branch of math. It is a *mathematical science.*
  - In statistics, it's important that we tie our conclusions back to data.
- context context context context context
- *ALWAYS* put your answer back into the context of the problem.
  - What does R^2 mean in *this* situation?
  - Why is regression useful to address *this* question?
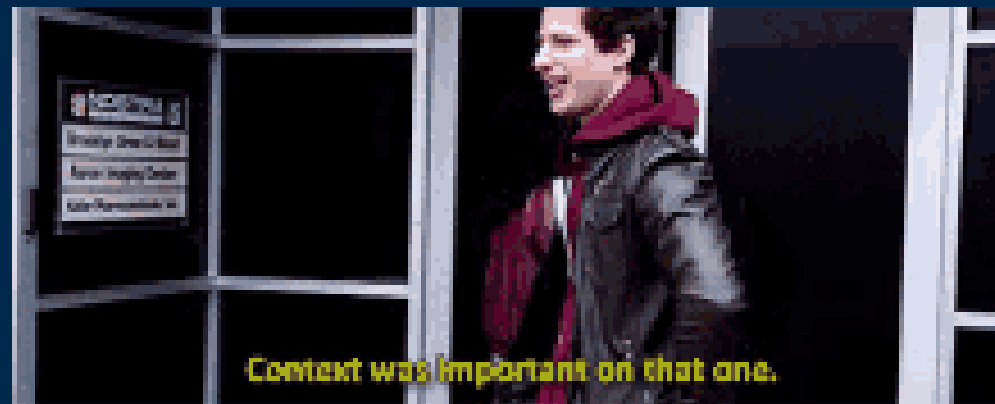
Context was important on that one.

Learning Objectives

Statistical Learning Objectives

1. Explore sample-to-sample variation
2. Investigate probability using long-run proportions

R Learning Objectives

1. Learn about reproducible randomness by "setting seeds"
2. Functions within functions: `table(sample())`
3. Line plots in R

Weekly Advice

- Randomness is **random**: your mileage may vary when you run code inside chunks.
- Check your HTML file before submitting it! You'll notice formatting issues you can easily fix (often by adding blank lines to your Rmd file).

Weekly Advice

- Randomness is **random**: your mileage may vary when you run code inside chunks.
- Check your HTML file before submitting it! You'll notice formatting issues you can easily fix (often by adding blank lines to your Rmd file).

Please try to follow along with this video. **It will help.**

Vectors (again)

A *character* vector is a vector where the elements are "strings" of text.

```
x <- c("hi", "this is", "a character vector.", "Are you impressed?")
x
```

```
[1] "hi"                "this is"              "a character vector."
[4] "Are you impressed?"
```

Again, note the use of the `c()` function.

Vectors (again)

A *character* vector is a vector where the elements are "strings" of text.

```
x <- c("hi", "this is", "a character vector.", "Are you impressed?")
x
```

```
[1] "hi"                  "this is"             "a character vector."
[4] "Are you impressed?"
```

Again, note the use of the `c()` function.

rep()

What's easier to code?

```
pets <- c("cat", "cat", "cat", "cat")
pets
```

```
[1] "cat" "cat" "cat" "cat"
```

rep()

What's easier to code?

```
pets <- c("cat", "cat", "cat", "cat")
pets
```

```
[1] "cat" "cat" "cat" "cat"
```

```
cats <- rep("cat", 4)
cats
```

```
[1] "cat" "cat" "cat" "cat"
```

rep()

What's easier to code?

```r
pets <- c("cat", "cat", "cat", "cat")
pets
```

```
[1] "cat" "cat" "cat" "cat"
```

```r
cats <- rep("cat", 4)
cats
```

```
[1] "cat" "cat" "cat" "cat"
```

rep(what you want to repeat, number of times to repeat it)

rep()

What's easier to code?

```
pets <- c("cat", "cat", "cat", "cat", "dog", "dog", "dog", "dog", "dog")
pets
```

```
[1] "cat" "cat" "cat" "cat" "dog" "dog" "dog" "dog" "dog"
```

```
pets2 <- c(rep("cat", 4), rep("dog", 5))
pets2
```

```
[1] "cat" "cat" "cat" "cat" "dog" "dog" "dog" "dog" "dog"
```
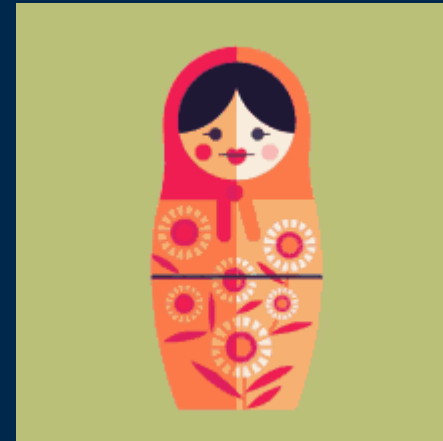
Functions in Functions

Arguments to functions can be functions! This is called **nesting**.

```
table(
  c(
    rep("heads", 5000),
    rep("tails", 5000)
  )
)
```



```
heads tails
 5000  5000
```

Functions in Functions

Arguments to functions can be functions! This is called **nesting**.

```
table(
  c(
    rep("heads", 5000),
    rep("tails", 5000)
  )
)
```



```
heads tails
 5000  5000
```

WATCH OUT FOR PARENTHESES

Remember `sample()`?

We used `sample()` to simulate rolling a die using the vector `1:6`.

We can also give `sample()` a character vector to sample from!

```r
coin <- c('heads', 'tails')
sample(coin, size = 30, replace = TRUE)
```

```
 [1] "tails" "heads" "tails" "heads" "heads" "tails" "tails" "heads" "heads"
[10] "heads" "heads" "tails" "tails" "heads" "tails" "heads" "tails" "heads"
[19] "tails" "tails" "heads" "tails" "tails" "heads" "heads" "heads" "tails"
[28] "tails" "tails" "heads"
```

The `prob` argument to `sample()`

We can simulate a *biased* coin using the `prob` argument.

- `prob` takes a vector of "probability weights", one per element of the vector to sample from
- `prob` applies the weights *in order*

```r
coin <- c('heads', 'tails')
sample(coin, size = 30, replace = TRUE, prob = c(0.3, 0.7))
```

```
 [1] "tails" "tails" "tails" "tails" "heads" "heads" "tails" "tails" "heads"
[10] "heads" "tails" "heads" "tails" "heads" "tails" "heads" "heads" "heads"
[19] "tails" "tails" "heads" "tails" "tails" "tails" "heads" "tails" "heads"
[28] "heads" "heads" "tails"
```

Pseudo-random numbers

- Humans are very bad at generating random numbers.
- Computers only **seem** better.
- Computers produce *pseudo-random* numbers: if you know the "seed", you know the entire sequence of "random" numbers.
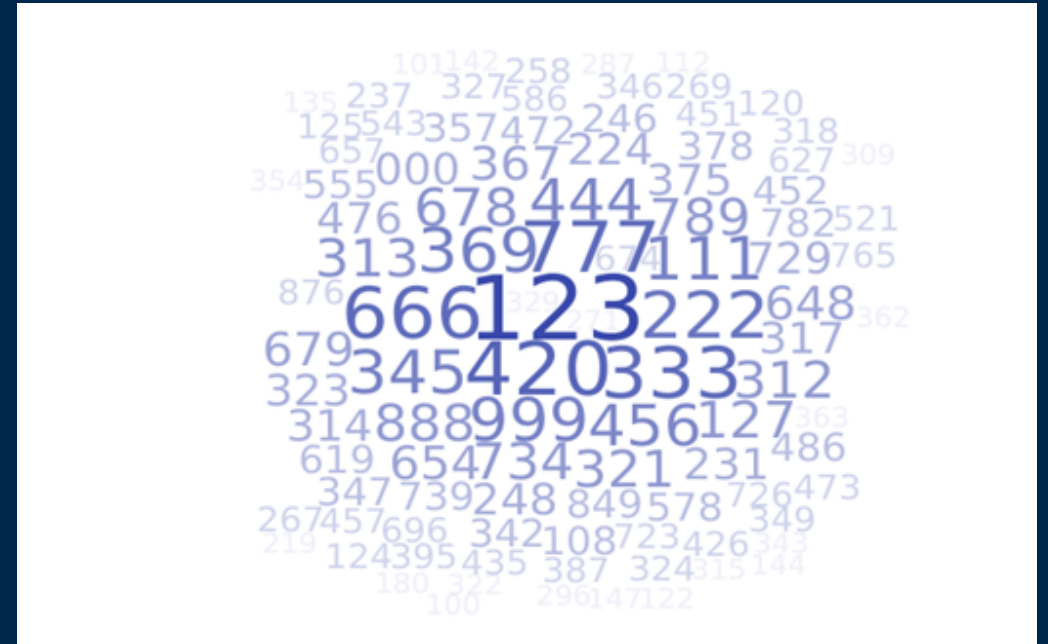
set.seed()

- We can tell R to use a particular "seed" with `set.seed()`.
- Setting the seed makes your randomness **reproducible**: you will now get the same answers (in your knitted document) as your peers, provided you use the same code.
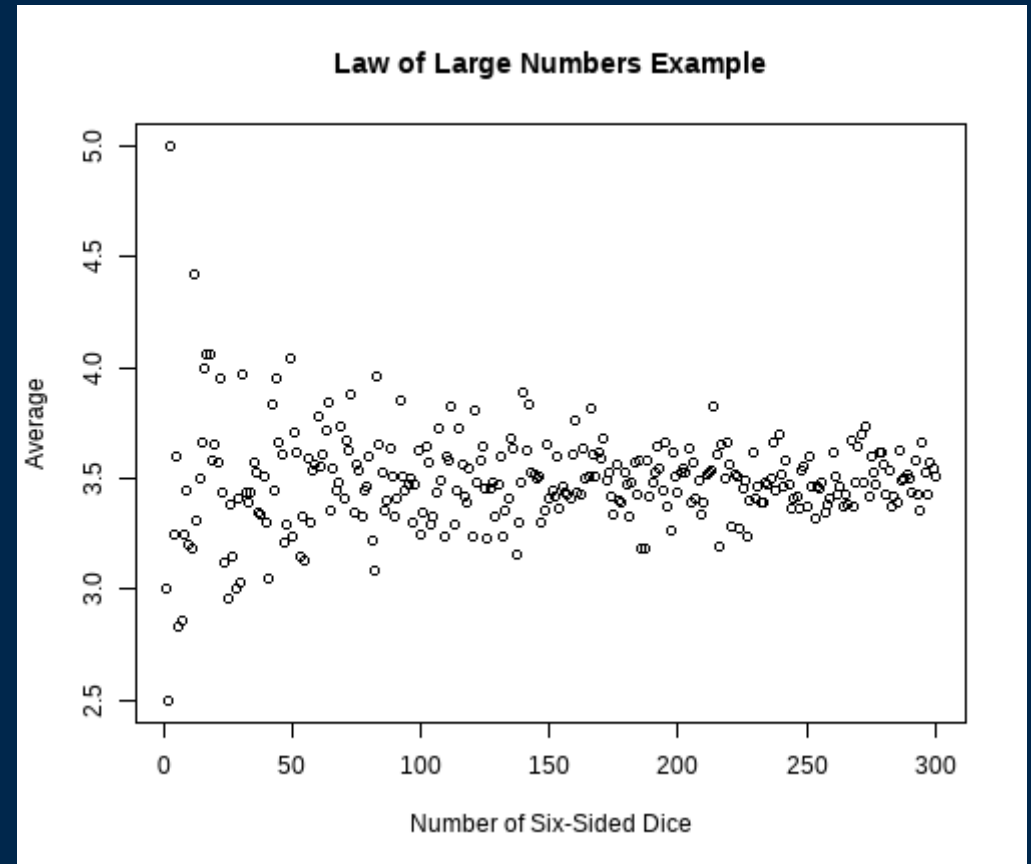
```r
set.seed(8362)
sample(1:5000, size = 3)
```

```
[1]  258 1834 2371
```

Line Graphs ☑

Remember this?

```r
sixSidedDieRoll <- function(n) {
  mean(sample(1:6, size = n, replace = T))
}
plot(sapply(1:300, sixSidedDieRoll),
     main = "Law of Large Numbers Example",
     xlab = "Number of Six-Sided Dice",
     ylab = "Average")
```
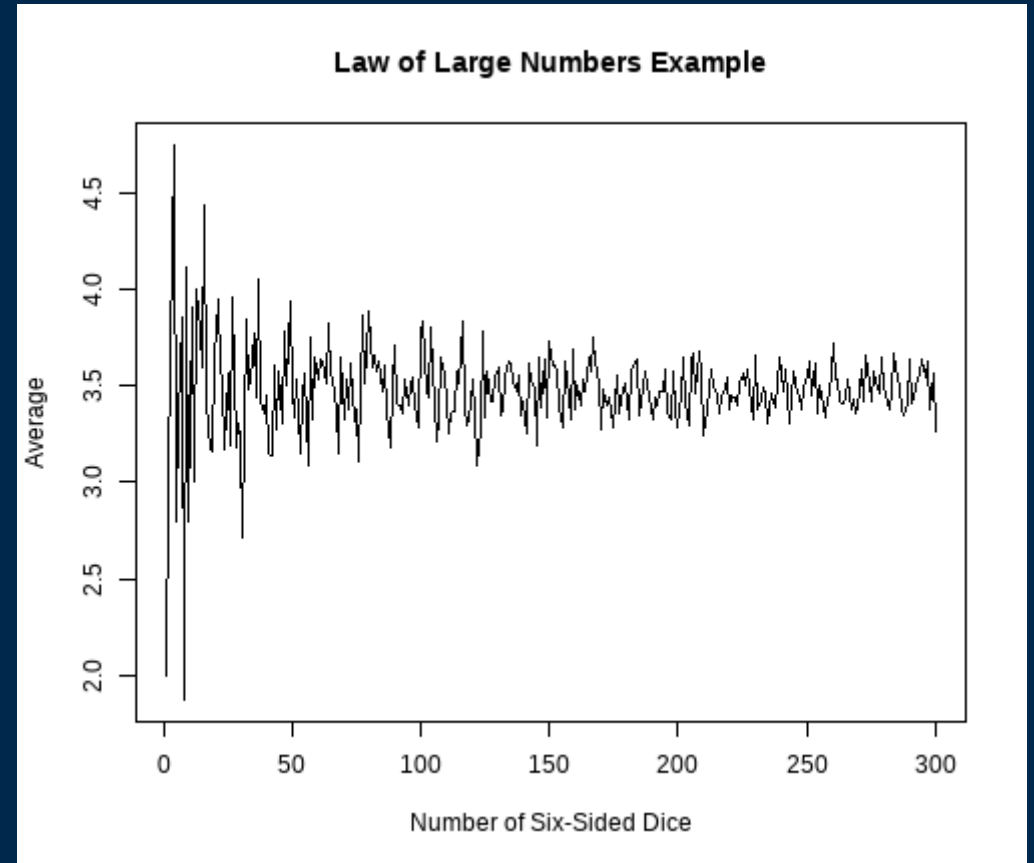

Law of Large Numbers Example

Line Graphs ☑

We can make a line graph with the
type argument to **plot()**:

```
plot(sapply(1:300, sixSidedDieRoll),
     main = "Law of Large Numbers Example",
     xlab = "Number of Six-Sided Dice",
     ylab = "Average",
     type = "l")
```

Use **type = l** for a **l**ine graph (that's
a **lowercase L**)

Line Graphs ☑

```
plot(sapply(1:300, sixSidedDieRoll),
     main = "Law of Large Numbers Example",
     xlab = "Number of Six-Sided Dice",
     ylab = "Average",
     type = "o",
     pch = 20)
```
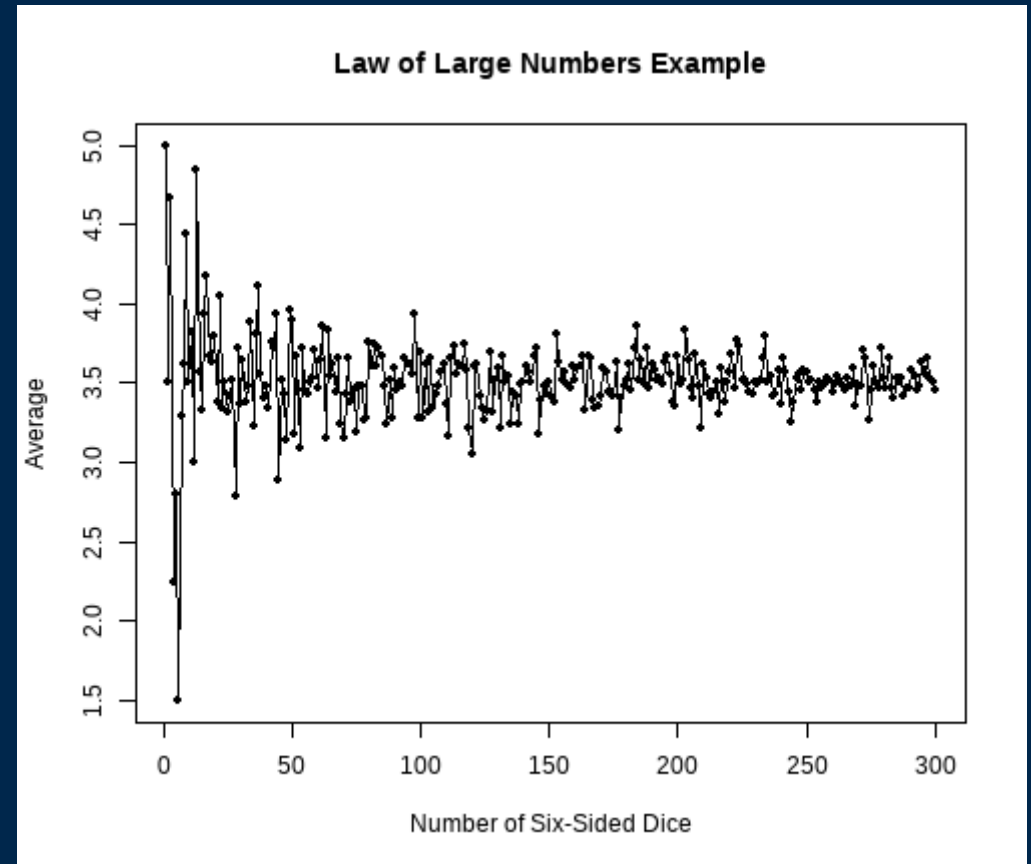
Use type = o to draw lines between points (and pch is back!)

Line Graphs ☑

```
plot(sapply(1:300, sixSidedDieRoll),
     main = "Law of Large Numbers Example",
     xlab = "Number of Six-Sided Dice",
     ylab = "Average",
     type = "o",
     pch = 20,
     lty = "dotted",
     lwd = 2)
```
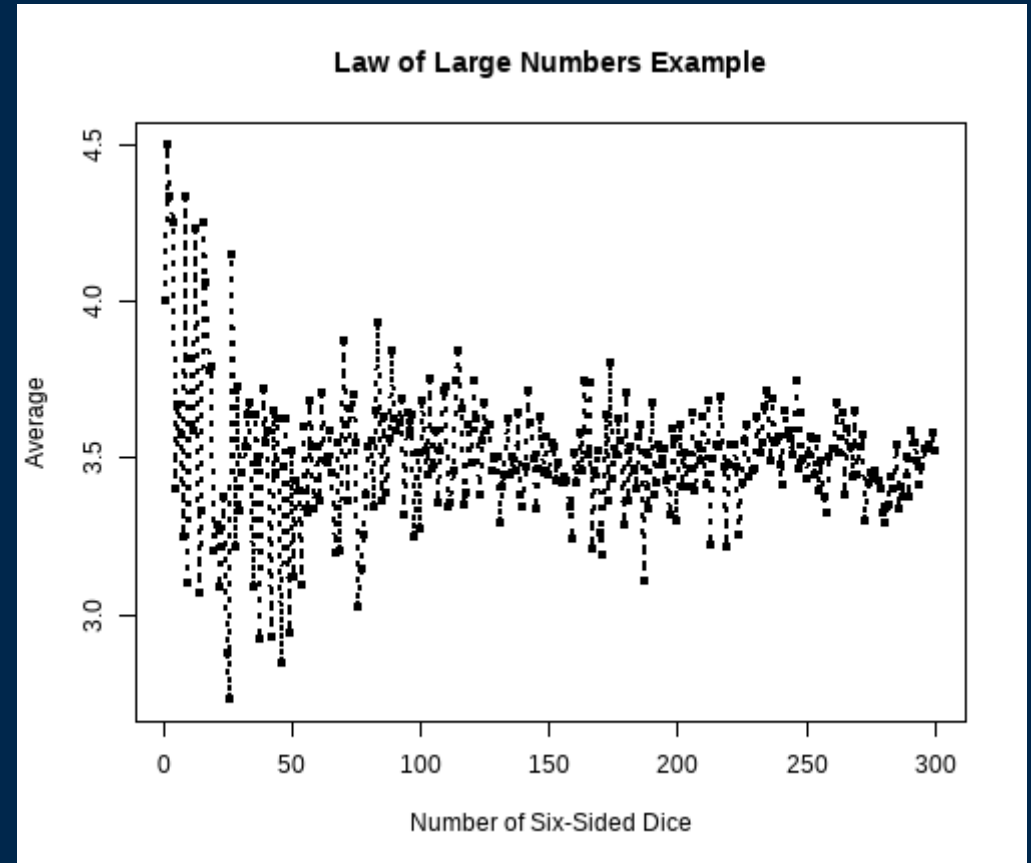

- Use `lty` to specify line type: (0=blank, 1=solid (default), 2=dashed, 3=dotted, 4=dotdash, 5=longdash, 6=twodash)
- Use `lwd` to specify line width (default is 1)

Lab Project 🖮

You will be **randomly** moved to a breakout room with 2-3 others for the rest of the lab (minus ~10 minutes)

Your tasks

1. Introduce yourself to your collaborators!
2. **Work together** to complete the "Try It!" and "Dive Deeper" sections.
3. **DO NOT LEAVE PEOPLE BEHIND.** You're all in this together.

How to get help

- I'll be floating around between breakout rooms to check on everyone
- Use the "Ask for help" button to flag me down
- Let me know when you're done

What questions do you have? Any issues?

"Exit Ticket"

Please take 1-2 minutes to complete the survey at

https://bit.ly/250ticket6

Reminders 💡

Your tasks for the week running Friday 10/2 - Friday 10/9:

Task	Due Date	Submission
M-Write 1 Final Revision	Wednesday 10/7	Canvas
Homework 5	Friday 10/9 8AM ET	course.work
Lab 5	Friday 10/9 8AM ET	Canvas

Stop by office hours! You can attend anyone's -- not just mine!

M-Write office hours schedule on Canvas (see MWrite Info on home page)