Teaching demo: Intro to simulation

Ciaran Evans

Context

- ► This lecture has been used as the first lecture in STA 279 Statistical Computing
- Students in Statistical Computing have taken STA 112 and are familiar with some R fundamentals; no other background is assumed
- First unit of Statistical Computing is on simulation studies
 - Allows review of some R basics
 - Provides context and motivation for fundamental computing concepts: data types, iteration, good coding practices

Today's lesson

- ► Learning goal: by the end of this lesson, students will be able to implement a short simulation to answer a probability question
- Topics reviewed or introduced:
 - Planning simulations
 - Vectors in R
 - Iteration
- Student participation:
 - Short neighbor/group discussions
 - Dialogue and questions throughout
 - Your turn: activity at the end of the lesson

Warm-up question

Problem: 10 people are at a party, and all of them are wearing hats. They each place their hat in a pile; when they leave, they choose a hat at random. What is the probability at least one person selected the correct hat?

Question: Work with your neighbor to discuss the following question:

Without calculating probabilities, how could you design an experiment to estimate this probability?

Designing an experiment Step 1: need to hots; Person 10 [7] [2] [10]	(ode ideas #\$ 1,2,,1((in a vector)
Step 2: randomly assign / shuffle nats Person 1 131 Derson 2 171 Step 3: check # of people who got original nat	randomly Sourph Custmat replacement
,	

Step 4: repeat many times? For loop

```
hats [3]

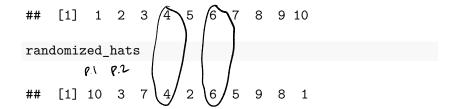
[3" entry

## [1] 3 = net 3
```

- ▶ hats is a **vector**, containing the numbers 1 to 10
- entries in a vector are accessed by their index

Step 2: everyone draws a random hat

take a rander sample from hats <- 1:10 randomized hats <- sample(hats, size = 10, replace = FALSE) without replacement hats



- The sample function creates a random sample from a vector
- How many people selected their original hat?

Step 3: check who got their original hat

hats

```
## [1] 1 2 3 4 5 6 7 8 9 10
```

 ${\tt randomized_hats}$

[1] 10 3 7 4 2 6 5 9 8 1

Step 3: check who got their original hat

hats

```
hats = randomized_hats
FALSE FALSE FALSE TRUE FALSE FALSE FALSE FALSE
             test for equality
             assignment
                        (similar to L-)
```

Step 3: check who got their original hat

```
hats
   [1] 1 2 3 4 5 6 7 8 9 10
randomized hats
   [1] 10 3 7 4 2 6 5 9 8 1
hats == randomized hats
                \
                      0 1
FALSE FALSE FALSE TRUE FALSE TRUE FALSE FALSE FALSE
# TRUE is 1, FALSE is 0
sum(hats == randomized hats)
                            gother hat back).
               2 people
## [1] 2
```

```
Step 3: check who got their original hat
   hats
      [1] 1 2 3 4 5 6 7 8 9 10
   ##
   randomized hats
   ## [1] 10 3 7 4 2 6 5 9 8 1
   hats == randomized hats
   FALSE FALSE FALSE TRUE FALSE TRUE FALSE FALSE FALSE
   # TRUE is 1, FALSE is 0
   sum(hats == randomized hats)
   ## [1] 2
   # did at least one person get their hat?
   sum(hats == randomized hats) > 0
                           at least one person get their
```

TRUE

reproducibility; setting a seed

```
set. seed (3)
```

```
sum(hats == randomized_hats) > 0
```

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

- ▶ In this case, at least one person received their original hat!
- Is this a good estimate of the probability?

Step 4: iteration

A for loop repeats code many times:

```
nsim <- 10000 # number of simulations

for(i in 1:nsim){

repect the following chunch of code

for index i=1,2,3,..., ~Sim''
}
```

Loop example

```
Stock of index is S
for(i in 1:5){
 print(3)
                          i= 1:
                          print(3)
  [1] 3
print(3)
## [1] 3
## [1] 3
```

Loop example

```
for(i in 1:5){
  print(i)
}
```

Loop example

Note: cook chunk in ar leap can depend on all index

```
for(i in 1:5){
    print(i)
}

## [1] 1
## [1] 2
## [1] 3
## [1] 4
## [1] 5
```

Step 4: iteration

A for loop repeats code many times:

```
## [1] TRUE
## [1] TRUE
## [1] TRUE
## [1] FALSE
```

[1] EVICE

[1] TRUE ## [1] TRUE

initial: results NA NA ... NA i=1: results T NA ... NA Step 4: iteration A for loop repeats code many times: 3=2 results T T NA .. NA nsim <- 10000 # number of simulations</pre> hats <- 1:10 , repeat NA vsiresults <- rep(NA, nsim) # vector to store results reate a vector called "results" for(i in 1:nsim){ randomized_hats <- sample(hats, size = 10, replace = FALSE) results[i] <- sum(hats == randomized_hats) > 0 storing in experition in it entry of essults

[1] TRUE TRUE FALSE FALSE TRUE

head(results)

0.632120S

A for loop repeats code many times:

```
nsim <- 10000 # number of simulations
hats <- 1:10
results <- rep(NA, nsim) # vector to store results
for(i in 1:nsim){
 randomized_hats <- sample(hats, size = 10,
                           replace = FALSE)
 results[i] <- sum(hats == randomized_hats) > 0
 fraction of times result was TRUE
                          recall Tisl
mean(results)
```

[1] 0.6307

Class activity

For the remainder of class, work with a neighbor on the class activity (link below and on the course website):

https://sta279-example.github.io/class_activities/ca_lecture_1.html

What comes next?

- Continuing probability simulations (gambler's ruin, airplane seating, Monty Hall problem, etc.)
 - setting seeds
 - good coding practices
 - for and while loops
 - nested loops
 - if...else if...else statements
- Statistical simulations
 - answering questions about linear regression models (e.g., does constant variance matter?)
 - ADFMP framework¹
 - introduction to writing functions

¹"Using simulation studies to evaluate statistical methods" (Morris *et al.* 2019)

