Lecture 2

Warm-up question

- A roulette wheel has 38 slots numbered 00, 0, and 1–36. Two are green, 18 are red, and 18 are black.
- If a gambler bets based on color, the return on a $1 bet is $2
- A gambler has $50, and will continuously bet $1 on red until they double their money (have $100) or lose the money they came with
- What is the probability the gambler doubles their money?

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

Designing an experiment

Step 1: representing the roulette wheel

```
1 wheel <- c(rep("green", 2), rep("black", 18), rep("red", 18))
2
3 wheel
```

```
 [1] "green" "green" "black" "black" "black" "black" "black" "black"
"black"
[10] "black" "black" "black" "black" "black" "black" "black" "black"
"black"
[19] "black" "black" "red"   "red"   "red"   "red"   "red"   "red"
"red"
[28] "red"   "red"   "red"   "red"   "red"   "red"   "red"   "red"
"red"
[37] "red"   "red"
```

- `rep` repeats a value a specified number of times
- `c()` combines vectors into a single vector

Step 2: spin the wheel!

```
1 spin <- sample(wheel, size = 1)
2
3 spin
```

```
[1] "red"
```

Step 3: change in money

```
money <- 50
spin <- sample(wheel, size = 1)

if(spin == "red"){
  money <- money + 1
} else {
  money <- money - 1
}

spin
```

```
[1] "red"
```

```
money
```

```
[1] 51
```

- if the result was red, gain a dollar
- otherwise, lose a dollar

Step 3: change in money

Another way of writing the conditional statement:

```
money <- 50
spin <- sample(wheel, size = 1)

money <- ifelse(spin == "red", money + 1, money - 1)

spin
```

```
[1] "red"
```

```
money
```

```
[1] 51
```

Step 4: keep spinning

The gambler continues to bet until they have $0 or $100.

Question: Is a `for` loop appropriate for iterating the betting process?

Step 4: keep spinning

```
money <- 50 # starting money

while(money > 0 & money < 100){
  spin <- sample(wheel, size = 1)
  money <- ifelse(spin == "red", money + 1, money - 1)
}

money
```

```
[1] 0
```

- `while` loop: repeat the process until the condition is true

Step 5: repeat the process

```r
set.seed(279)

nsim <- 1000
results <- rep(NA, nsim)

for(i in 1:nsim){
  money <- 50 # starting money

  while(money > 0 & money < 100){
    spin <- sample(wheel, size = 1)
    money <- ifelse(spin == "red", money + 1, money - 1)
  }

  results[i] <- ...
}
```

- What should I check at each iteration?

Step 5: repeat the process

```r
set.seed(279)

nsim <- 1000
results <- rep(NA, nsim)

for(i in 1:nsim){
  money <- 50 # starting money

  while(money > 0 & money < 100){
    spin <- sample(wheel, size = 1)
    money <- ifelse(spin == "red", money + 1, money - 1)
  }

  results[i] <- money == 100
}

mean(results)
```

```
[1] 0.008
```

A new question

In STA 112, you learned about the simple linear regression model:

$$Y_i = \beta_0 + \beta_1 X_i + \varepsilon_i$$

Question: What assumptions does this model make?

A new question

In STA 112, you learned about the simple linear regression model:

$$Y_i = \beta_0 + \beta_1 X_i + \varepsilon_i$$

Question: How important is it that $\varepsilon_i \sim N(0, \sigma^2)$? Does it matter if the errors are *not* normal?

Activity

$$Y_i = \beta_0 + \beta_1 X_i + \varepsilon_i$$

Activity: With a neighbor, brainstorm how you could use simulation to assess the importance of the normality assumption (you do not need to write code!).

- How would you simulate data?
- What result would you measure for each run of the simulation?

Activity

$$Y_i = \beta_0 + \beta_1 X_i + \varepsilon_i$$

How would you study the importance of the normality assumption?

Simulating data

To start, simulate data for which the normality assumption holds:

```r
n <- 100 # sample size
beta0 <- 0.5 # intercept
beta1 <- 1 # slope

x <- runif(n, min=0, max=1)
noise <- rnorm(n, mean=0, sd=1)
y <- beta0 + beta1*x + noise
```

- `runif(n, min=0, ,max=1)` samples X_i uniformly between 0 and 1
- `rnorm(n, mean=0, sd=1)` samples $\varepsilon_i \sim N(0, 1)$

Fit a model

```r
n <- 100 # sample size
beta0 <- 0.5 # intercept
beta1 <- 1 # slope

x <- runif(n, min=0, max=1)
noise <- rnorm(n, mean=0, sd=1)
y <- beta0 + beta1*x + noise

lm_mod <- lm(y ~ x)
lm_mod
```

```
Call:
lm(formula = y ~ x)

Coefficients:
(Intercept)            x
      0.647        0.437
```

Calculate confidence interval

```r
lm_mod <- lm(y ~ x)

ci <- confint(lm_mod, "x", level = 0.95)
ci
```

```
       2.5 %   97.5 %
x -0.1957341 1.069806
```

- **Question:** How can we check whether the confidence interval contains the true β_1 ?

Calculate confidence interval

```r
lm_mod <- lm(y ~ x)

ci <- confint(lm_mod, "x", level = 0.95)
ci
```

```
       2.5 %   97.5 %
x -0.1957341 1.069806
```

- **Question:** How can we check whether the confidence interval contains the true β_1 ?

```r
ci[1] < 1 & ci[2] > 1
```

```
[1] TRUE
```

Repeat!

```r
nsim <- 1000
n <- 100 # sample size
beta0 <- 0.5 # intercept
beta1 <- 1 # slope
results <- rep(NA, nsim)

for(i in 1:nsim){
  x <- runif(n, min=0, max=1)
  noise <- rnorm(n, mean=0, sd=1)
  y <- beta0 + beta1*x + noise

  lm_mod <- lm(y ~ x)
  ci <- confint(lm_mod, "x", level = 0.95)

  results[i] <- ci[1] < 1 & ci[2] > 1
}
mean(results)
```

Repeat!

```r
nsim <- 1000
n <- 100 # sample size
beta0 <- 0.5 # intercept
beta1 <- 1 # slope
results <- rep(NA, nsim)

for(i in 1:nsim){
  x <- runif(n, min=0, max=1)
  noise <- rnorm(n, mean=0, sd=1)
  y <- beta0 + beta1*x + noise

  lm_mod <- lm(y ~ x)
  ci <- confint(lm_mod, "x", level = 0.95)

  results[i] <- ci[1] < 1 & ci[2] > 1
}
mean(results)
```

```
[1] 0.951
```

- What should we do next?

Course goals

- Develop computing skills to work with data and answer statistical questions
- Emphasize reproducibility and good coding practices
- Introduce other important computing tools for statistics and data science (Python, SQL, Git)

What this course isn't:

- An exhaustive list of R or Python functions
- A computer science course
- A deep dive into how R actually works

Tentative topics

- Simulation
- Intro to Python
- Data wrangling and manipulation
- Intro to SQL
- Version control and reproducibility
- Working with text data
- Time permitting: select advanced topics

Course components

Component	Weight
Homework	50%
Midterm exam	10%
Final exam	20%
Project	20%

Diversity and inclusion

In this class, we will embrace diversity of age, background, beliefs, ethnicity, gender, gender identity, gender expression, national origin, neurotype, race, religious affiliation, sexual orientation, and other visible and non-visible categories. The university and I do not tolerate discrimination.

- You deserve to be addressed in the manner you prefer. To guarantee that I address you properly, you are welcome to tell me your pronoun(s) and/or preferred name at any time, either in person or via email.