Intermediate R Programming

POS6933: Computational Social Science

Jake S. Truscott, Ph.D

University of Florida

Spring 2026

Overview

- **Today's Goal**: Improve Effectiveness w/ R Programming
- Random Number Generation in R
- Loops and Iteration
- Visualizing Data and Relationships Using `ggplot::()`

Coin Flips

What is the probability that any independent coin flip will land on heads?

Coin Flips

What is the probability that any independent coin flip will land on heads?

Does this change if I flip 50 times?

Coin Flips

What is the probability that any independent coin flip will land on heads?

Does this change if I flip 50 times?

What about 100 times?

Coin Flips

What is the probability that any independent coin flip will land on heads?

Does this change if I flip 50 times?

What about 100 times?

What about 1000 times?

Coin Flips

What is the probability that any independent coin flip will land on heads?

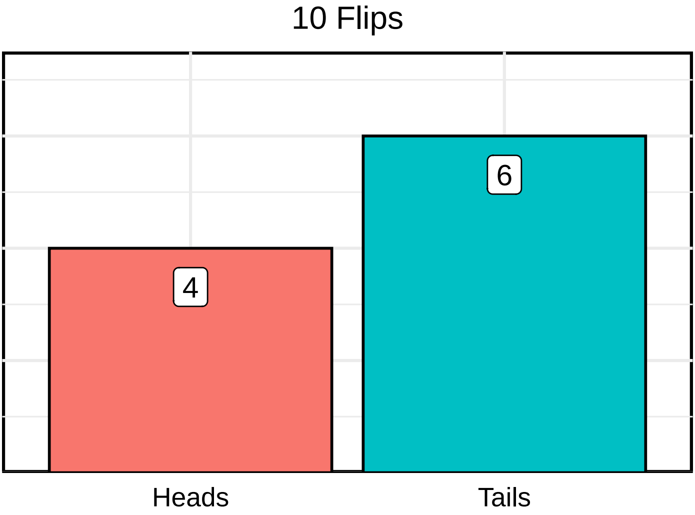
Does this change if I flip 50 times?

What about 100 times?

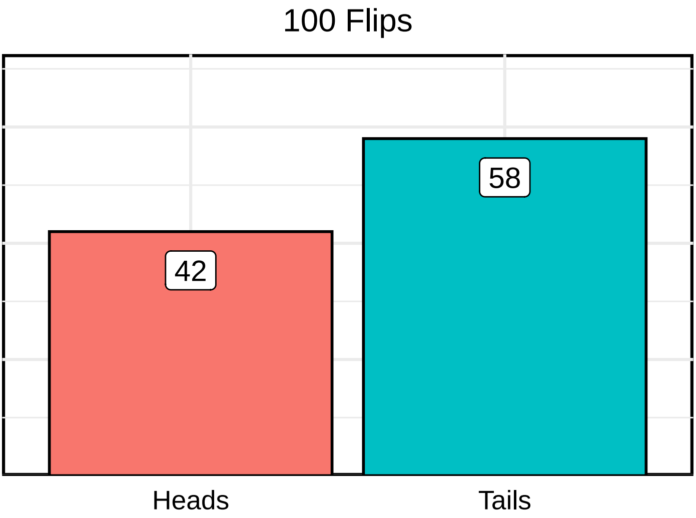
What about 1000 times?

What about 10000 times?

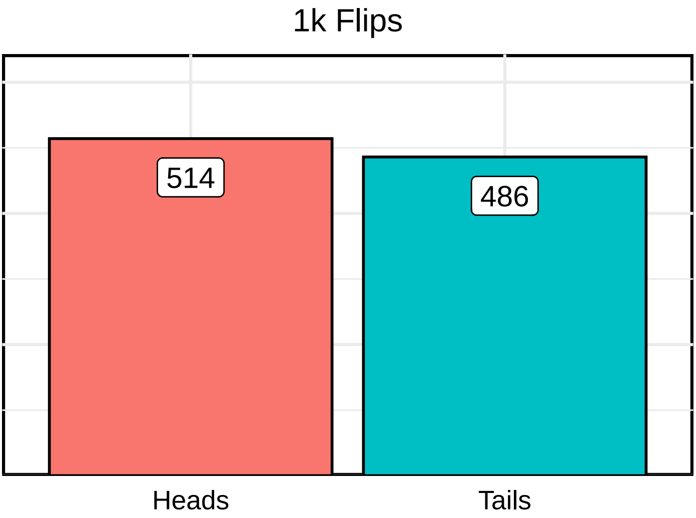
Coin Flips (Cont.)



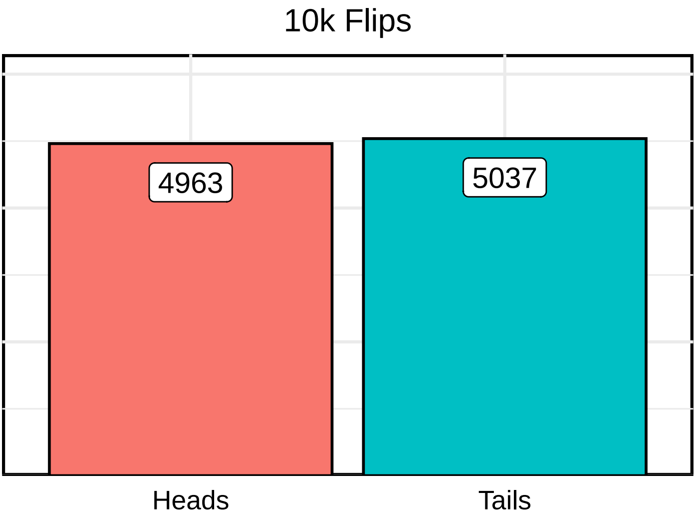
Coin Flips (Cont.)



Coin Flips (Cont.)



Coin Flips (Cont.)



Coin Flips (Cont.)

- We can use `sample()` to randomly select elements from a vector
- In this case, a coin flip where $p(heads) = p(tails) = 0.5$

```
sides <- c("Heads", "Tails")  # Flip Options
single_flip <- sample(sides, size = 1)  # Single Draw
print(single_flip)
```

```
[1] "Tails"
```

6-Sided Die

- We can use the same approach to “roll” a six-sided die.

```
sides <- c(1:6)  # 1, 2, 3, 4, 5, 6
single_roll <- sample(sides, size = 1)  # Single Roll
message("Result of Single Roll: ", single_roll)
```

```
Result of Single Roll: 2
```

Poker Hands

- We can even use it to do more complex operations like simulate a random draw from 5-card Poker

```
cards <- as.character(c(2:10, "J", "Q", "K", "A"))
# All Card Values
suits <- c("Hearts", "Diamonds", "Spades", "Clubs")
# Suits

deck <- expand.grid(value = cards, suit = suits) %>%
    mutate(card = paste(value, "of", suit)) %>%
    pull(card)  # Create a Full Deck

random_draw <- sample(deck, size = 5, replace = F)
# Random 5-Card Draw w/out Replacement
```

Poker Hands (Cont.)

```
Hand:
3 of Diamonds
5 of Hearts
10 of Diamonds
2 of Diamonds
Q of Diamonds
```

Generating Distributions

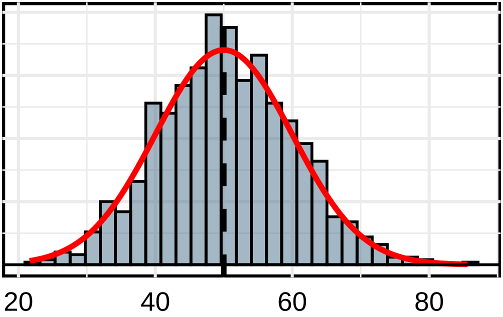
- What if we wanted to move beyond random selection where each draw or iteration exists with equal probability or within a uniform distribution?

- R is very flexible and capable of illustrating sampling distributions against expected outcomes

Generating Distributions (Standard Normal)

- Let's start with 1000 samples from a standard normal distribution where $\mu = 50$ and $\sigma = 10$

```r
normal <- rnorm(1000, mean = 50, sd = 10)
```

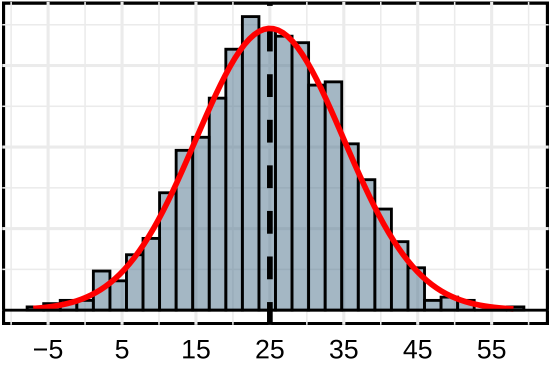


Generating Distributions (Standard Normal)

- **Your Turn**: Generate 1000 draws from a standard normal distribution where $\mu = 25$ and $\sigma = 10$.

Generating Distributions (Standard Normal – Ex)

```r
normal <- rnorm(1000, mean = 25, sd = 10)
```

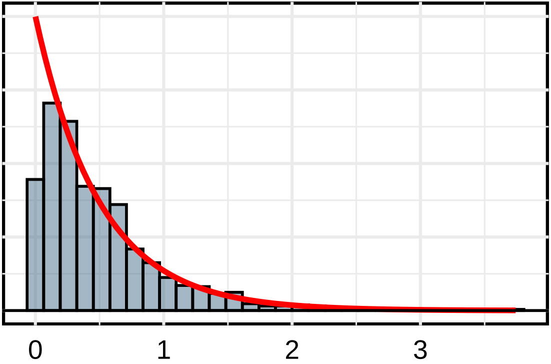


Generating Distributions (Exponential – Ex)

- **Your Turn**: Generate 1000 draws from an exponential distribution where `rate` $= 2$

Generating Distributions (Exponential – Ex)

```
exp <- rexp(1000, rate = 2)
```



Functions (Basics)

- Functions are reusable blocks of code that perform a specific task when called, helping avoid repetition.
- They can take arguments (inputs) and return values (outputs), making them flexible and generalizable.
- They can also be combined, nested, and used within other functions to build complex workflows in a clear, organized way.

Functions (Basic Syntax)

```
function_name <- function(input_1, input_2) {

    # Code to Assume Input_1 and Input_2

    # return(Return Output Value or Object)

}
```

Functions (Example)

```r
add_numbers <- function(x, y) {
    result <- x + y
    return(result)
}

add_numbers(5, 3)
```

Functions (Basics, Cont.)

- Take some time to try your own!
- Try:
    - Random Number Generation
    - Easy Task Completion (e.g., addition, subtraction, etc.)

Loops (Basics)

- In R, a for loop is a control structure used to repeat a block of code a fixed number of times, iterating over a sequence of values. The basic syntax is:

```
for (variable in sequence) {
    # Repeating Code Routine return(Result Value
    # or Object)
}
```

Loops (Basics, Cont.)

- For example, we can complete basics rolls of six-sided dice:

```
rolls <- c()

for (i in 1:10) {

    temp_roll <- sample(1:6, 1, replace = TRUE, prob = rep(1/6,
        6))

    rolls <- c(rolls, temp_roll)
}

rolls  # Print
```

```
 [1] 4 6 3 1 2 2 3 2 5 5
```

Loops (Basics, Cont.)

- We can also conditionally iterate through different values from the `functions` example

```
add_numbers <- function(x, y) {
    result <- x + y
    return(result)
}

available_values <- c(1:10)
sums <- c()

for (pair in seq(1:10)) {
    temp_pair <- sample(available_values, 2)
    sums <- c(sums, add_numbers(temp_pair[1], temp_pair[2]))
}

sums
```

```
 [1]  9  7  9  9 12 15 19 15  8 13
```

Loops (Basics, Cont.)

- I can also deal 5 hands from a standard 52-card deck for a game of Texas Hold 'Em
- Here's the setup – What's Next?

```r
set.seed(1234)  # Seed
cards <- as.character(c(2:10, "J", "Q", "K", "A"))
suits <- c("Hearts", "Diamonds", "Spades", "Clubs")
deck <- expand.grid(value = cards, suit = suits) |>
    mutate(card = paste(value, "of", suit)) |>
    pull(card)  # Create a Full Deck

hands <- lapply(1:5, function(x) x)
```

Loops (Basics, Cont.)

```
for (card in 1:2) {
    for (player in 1:5) {
        temp_player_card <- sample(deck, 1, replace = F)
        deck <- deck[!deck %in% temp_player_card]
        hands[[player]][card] <- temp_player_card
    }  # For All 5 Players
}  # For Both Cards

do.call(cbind, hands)
```

```
     [,1]          [,2]            [,3]
[1,] "3 of Spades" "4 of Diamonds" "J of Diamonds"
[2,] "A of Clubs"  "10 of Hearts"  "6 of Hearts"
     [,4]         [,5]
[1,] "2 of Clubs" "10 of Clubs"
[2,] "6 of Clubs" "7 of Diamonds"
```

Games of Chance: Blackjack

What are the basic rules of Blackjack?

Rules of Blackjack:

- Objective: Beat the dealer by getting closer to 21 without going over
- Card values:
  - Number Cards = Face Value
  - Face Cards = 10 (Aces = 1 *or* 11)
- Dealer Rules: Dealer reveals cards after players act and must hit until *at least* 17
- Gameplay:
  - Go Over 21 = **BUST** (Loss)
  - Tie w/ Dealer = Push (No Win/Loss)
  - Standard Win = **1:1** (Win Bet x2)
  - Blackjack (Ace + 10-Value Card = **3:2**

Blackjack Exercise

Write an R routine to play a round of Blackjack. I will do the same.

- *Hint*: Sample from all 52 cards without replacement...

Blackjack Exercise (Cont.)

1. What if we play with a four-deck shoe?

Blackjack Exercise (Cont.)

1. What if we play with a four-deck shoe?
2. What if I wanted to repeat this process 1,000 times?

Hint: Use a loop!

Blackjack Exercise (Cont.)

1. Assume I begin with $1000 every day and bet $100 each game (though I'll only play 10 hands. . .). Over 100 days, approximately how much money am I left with? *Note*: If I run out of money on a given day, I'm done – also, each day restarts with $1000 but previous day's leftover sum is added to aggregate winnings.
2. What if I start with $1000 but don't replace the money every day. . . How much will I have after 10 days? 50 days?
3. Take some time then play around with blackjack_simulation.R

Roulette Exercise

- Head over to Course GitHub (Intermediate Programming in `R`)
- Bottom of `Number Generation & Loops`

Data Visualization

- `ggplot()` is an incredibly flexible visualization tool
- There's a balance between professional & *noisy* visualizations
- Some journals & reviewers are more critical than others
- **Goal**: *Publication-ready* visualizations capable of relaying inferential value on its own

My Default ggplot() Aesthetics

```
default_ggplot_theme  <- theme_minimal(base_size = 12) +
  theme(
    plot.title = element_text(hjust = 0.5, size = 12),

    axis.title = element_text(size = 12, colour = 'black'),

    axis.text = element_text(size = 10, colour = 'black'),

    panel.background = element_rect(linewidth = 1, colour = 'black', fill = NA),

    legend.position = 'bottom',

    legend.background = element_rect(linewidth = 1, colour = 'black', fill = NA),
  )
```

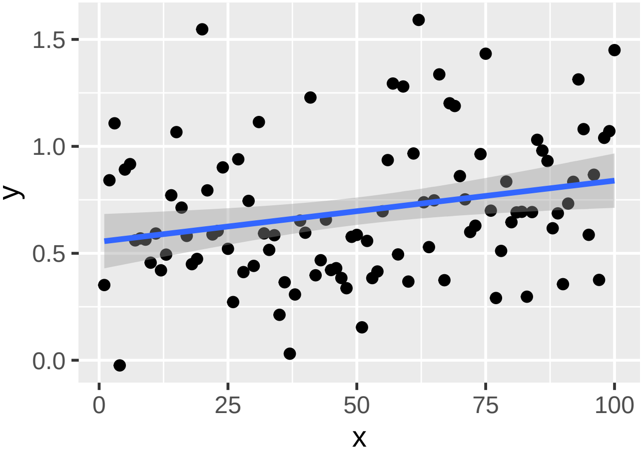
My Default ggplot() Aesthetics

```r
set.seed(1234)

sample_data <- tibble(x = c(1:100), y = rnorm(100,
    mean = 0.75, sd = 0.33))

summary(sample_data)
```

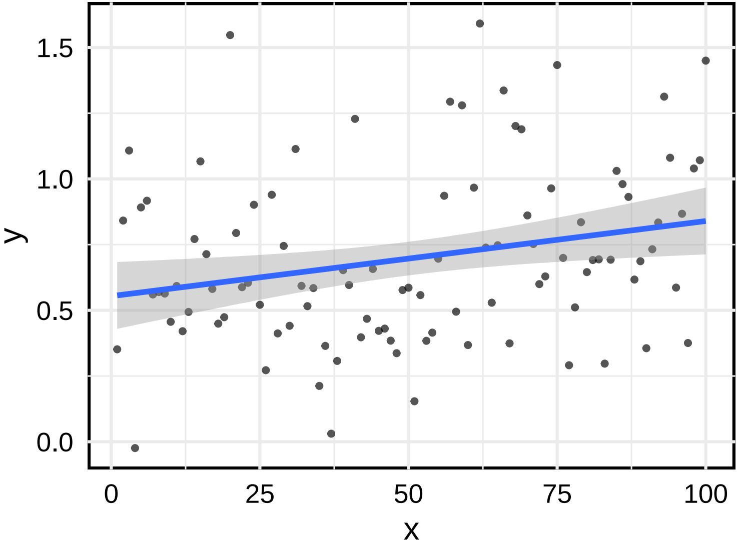
```
       x                y
 Min.   :  1.00   Min.   :-0.02408
 1st Qu.: 25.75   1st Qu.: 0.45454
 Median : 50.50   Median : 0.62307
 Mean   : 50.50   Mean   : 0.69827
 3rd Qu.: 75.25   3rd Qu.: 0.90550
 Max.   :100.00   Max.   : 1.59117
```

My Default ggplot() Aesthetics

```
sample_data %>%
    ggplot(aes(x = x, y = y)) + geom_point() + geom_smooth(method = "lm",
    formula = "y~x")
```



Adding Default Aes



Stargazer

```r
library(stargazer)
temp_lm <- lm(Sepal.Length ~ Sepal.Width + Petal.Length +
    Petal.Width, data = iris)

stargazer::stargazer(temp_lm, type = "text", omit.stat = c("ser",
    "f", "adj.rsq"), dep.var.caption = "")
```

Stargazer – Text Example

```
========================================
                    Sepal.Length
----------------------------------------
Sepal.Width           0.651***
                       (0.067)

Petal.Length          0.709***
                       (0.057)

Petal.Width          -0.556***
                       (0.128)

Constant              1.856***
                       (0.251)

----------------------------------------
Observations             150
R2                      0.859
========================================
Note:        *p<0.1; **p<0.05; ***p<0.01
```

Stargazer – Latex Example (type = 'latex')

```
% Table created by stargazer v.5.2.3 by Marek Hlavac, Social Policy Institute. E-mail: marek.hlavac at gmail.co
% Date and time: Fri, Jan 23, 2026 - 10:11:48 AM
\begin{table}[!htbp] \centering
  \caption{}
  \label{}
\begin{tabular}{@{\extracolsep{5pt}}lc}
\\[-1.8ex]\hline
\hline \\[-1.8ex]
\\[-1.8ex] & Sepal.Length \\
\hline \\[-1.8ex]
 Sepal.Width & 0.651$^{***}$ \\
  & (0.067) \\
  & \\
 Petal.Length & 0.709$^{***}$ \\
  & (0.057) \\
  & \\
 Petal.Width & $-$0.556$^{***}$ \\
  & (0.128) \\
  & \\
 Constant & 1.856$^{***}$ \\
  & (0.251) \\
  & \\
\hline \\[-1.8ex]
Observations & 150 \\
R$^{2}$ & 0.859 \\
\hline
\hline \\[-1.8ex]
\textit{Note:}  & \multicolumn{1}{r}{$^{*}$p$<$0.1; $^{**}$p$<$0.05; $^{***}$p$<$0.01} \\
\end{tabular}
\end{table}
```

Stargazer – Latex Example Rendered

Table 1: Rendered `stargazer` Table

	Sepal.Length
Sepal.Width	0.651^{***}
	(0.067)
Petal.Length	0.709^{***}
	(0.057)
Petal.Width	-0.556^{***}
	(0.128)
Constant	1.856^{***}
	(0.251)
Observations	150
R^2	0.859
Note:	*p<0.1; **p<0.05; ***p<0.01

Stargazer – Multiple Models

Table 2: Rendered `stargazer` Table with 2 Models

	Sepal.Length	
	(1)	(2)
Sepal.Width	0.651***	
	(0.067)	
Petal.Length	0.709***	
	(0.057)	
Petal.Width	−0.556***	−0.311***
	(0.128)	(0.114)
Sepal.Width:Petal.Length		0.185***
		(0.017)
Constant	1.856***	4.150***
	(0.251)	(0.078)
Observations	150	150
R^2	0.859	0.821
Note:	*p<0.1; **p<0.05; ***p<0.01	

Stargazer – Summary Data

```r
stargazer(iris, type = "text", summary = TRUE, title = "Summary of Iris Dat
```

```
Summary of Iris Dataset
===========================================
Statistic     N   Mean  St. Dev.  Min   Max
-------------------------------------------
Sepal.Length 150 5.843  0.828   4.300 7.900
Sepal.Width  150 3.057  0.436   2.000 4.400
Petal.Length 150 3.758  1.765   1.000 6.900
Petal.Width  150 1.199  0.762   0.100 2.500
-------------------------------------------
```

Visualization Example

- Using the `mtcars` dataset – `library(mtcars)` – complete the following:

  1. Using `mpg` as the dependent variable, compile two models using `cyl`, `disp`, `hp`, and `wt` – the second should have an interaction between `disp` and `wt`.
  2. Produce a table using `stargazer` of the resulting models.
  3. Use `ggplot` to illustrate the distribution of each of the variables listed in (1).

Looking Forward

- Homework: Problem Set (Class 2)
- Next Class: Parallel Computing