An Introduction to Probability

Probability: Basic Definitions and Rules

Martin Summer

13 January, 2026

Probability: Basic Definitions and Rules

Probability: Basic Definitions and Rules

This lecture explores the fundamentals of probability, including:

- Random experiments, sample spaces, and events
- Empirical probability and its foundational role
- The concept of independence and interactions between events

We will aim for a more formal discussion of probability concepts and expand on their practical applications using R, with stock market data as our leading example.

Probability Terminology

Random Experiment

Definition: Random Experiment

A process with a set of possible outcomes, where the specific outcome cannot be predicted with certainty beforehand.

Example: Observing whether a stock price rises or falls tomorrow.

Possible outcomes depend on the agreed scope:

- *Simplified:* `{rise, fall}`
- *Extended:* `{rise, fall, unchanged}`

Sample Space

Definition: Sample Space

The collection of all possible outcomes of a random experiment, denoted by the set S.

Example: For the stock price experiment:

$S = \{\text{rise}, \text{fall}\}$

Events and Simple Events

Definition: Basic Outcome, Event, Simple Event

- **Basic Outcome:** A single possible result of a random experiment.
- **Event:** A subset of the sample space representing one or more outcomes.
- **Simple Event:** An event containing exactly one basic outcome.

Example:

$S = \{\text{rise}, \text{fall}\}$

- Event `{rise}` is a simple event.
- Event `{rise, fall}` spans the entire sample space.

Probability as a Measure

Probability as a Measure

In the theory of probability:

- **Probability** is a function that assigns a likelihood to events.
- Properties of probability:
    1. $P(S) = 1$
    2. $0 \leq P(A) \leq 1$ for all events A
    3. For mutually exclusive events A and B:
       $P(A \cup B) = P(A) + P(B)$

Discrete and Non-Discrete Sample Spaces

Discrete and Non-Discrete Sample Spaces

- **Discrete Sample Spaces:** Finite or countable sequences of outcomes.

  Example: Coin tosses until the first Heads

  $S = \{H, TH, TTH, TTTH, ...\}$

- **Non-Discrete Sample Spaces:** Include uncountable sets.

  Example: Outcomes from continuous processes (to be discussed later).

Applications of Probability

Applications of Probability

In practice:

- Probabilities are expressed as numbers between 0 and 1.
- They are abstract measures of uncertainty in theory.
- From the perspective of the theory they are *given*.
- Compare to the concept of distance in geometry.
- Practical applications often require statistical methods to estimate probabilities.

Probability and the Language of Sets

Probability and the Language of Sets

Probability theory relies on the language of sets to describe relationships between events.
Understanding key set operations is essential for working with probabilities effectively.

Set Union

Definition: Set Union

The **union of two events** A and B represents all outcomes that belong to A, B, or both.

It is written $A \cup B$.

Example: Rolling a Die

Consider the experiment of rolling a die:

- Sample space: $S = \{1, 2, 3, 4, 5, 6\}$
- Event A: The outcome is 1, 2, or 3, written as $A = \{1, 2, 3\}$.
- Event B: The outcome is an even number, written as $B = \{2, 4, 6\}$.
- The union of A and B:

$$A \cup B = \{1, 2, 3, 4, 6\}$$

Implementing Set Operations in R

In R, we define the sets A and B using the assignment operator:

```r
A <- c(1,2,3)
B <- c(2,4,6)
```

- To compute the union of A and B, we use the `union()` function:
- `union(A, B)`
- This gives us the union of both sets, $A \cup B = \{1, 2, 3, 4, 6\}$.

Visualizing Set Union

To better understand the operation of set union, we can visualize the example:

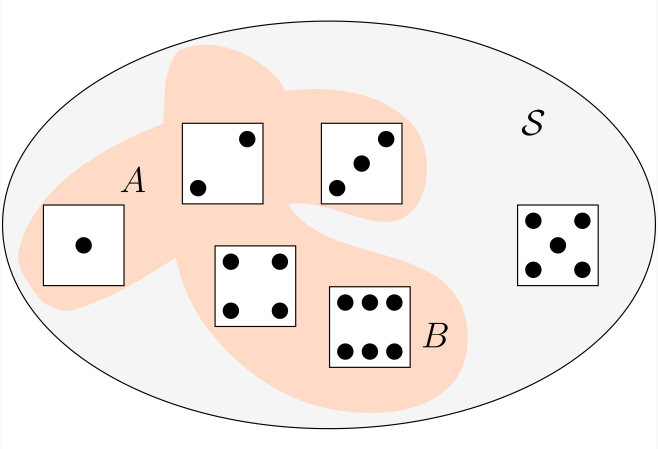


Figure 1: The meaning of set union

Set Intersection

Definition: Intersection

The **intersection of two events** includes all outcomes that are both in A and in B.

It is written as $A \cap B$.

Implementing Intersection in R

To compute the intersection of A and B in R, we use the `intersect()` function:

```r
intersect(A, B)
```

```
[1] 2
```

This returns the set $A \cap B$, containing the outcomes that belong to both A and B.

Visualizing Set Intersection

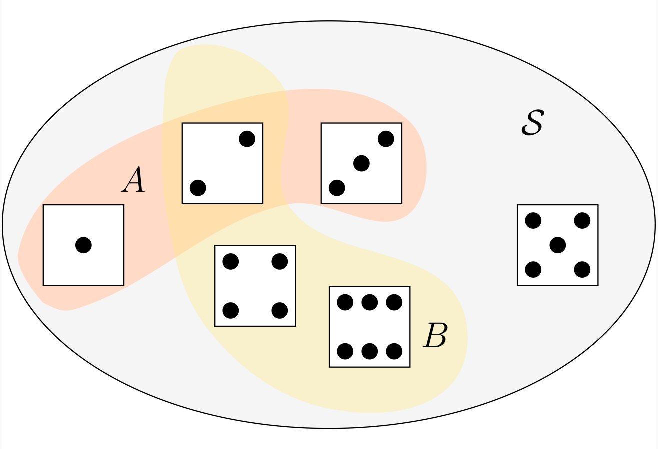


Figure 2: The meaning of set intersection

Complement

Definition: Complement

The **complement** of an event A within the sample space S is the set of all outcomes that are in S but not in A.
It is written as $S \setminus A$.

Example: Complement in R

To compute the complement of $A \cup B$ with respect to the sample space S, we use the `setdiff()` function in R.

R Code Example:

```r
S <- c(1,2,3,4,5,6)
setdiff(S, union(A, B))
```

```
[1] 5
```

This computes the set difference of S and $A \cup B$ in the context of the die-rolling example.

Visualizing the Complement

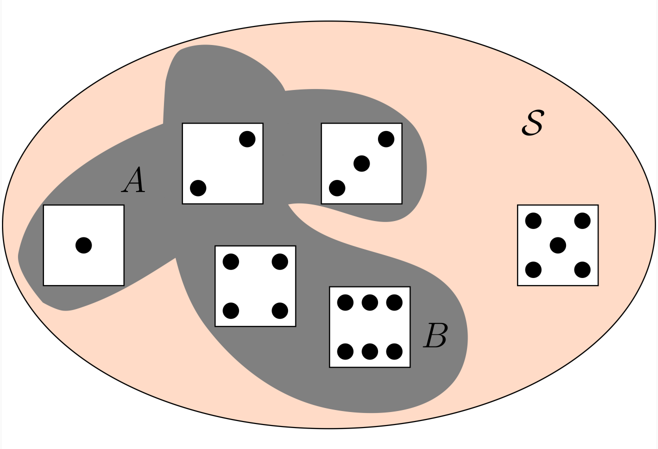


Figure 3: The meaning of complement

Mutually Exclusive Events

Definition: Mutually Exclusive

Two events A and B are **mutually exclusive** if they cannot occur simultaneously.
This means $A \cap B = \emptyset$, their intersection is empty.

Example: Mutually Exclusive Events in R

Consider the sets of even and odd outcomes in a die roll:
- $B = \{2, 4, 6\}$ (even outcomes)
- $C = \{1, 3, 5\}$ (odd outcomes)

Example: Mutually Exclusive Events in R Continued

To find their intersection in R:

```r
B <- c(2,4,6)
C <- c(1,3,5)

intersect(B, C)
```

```
numeric(0)
```

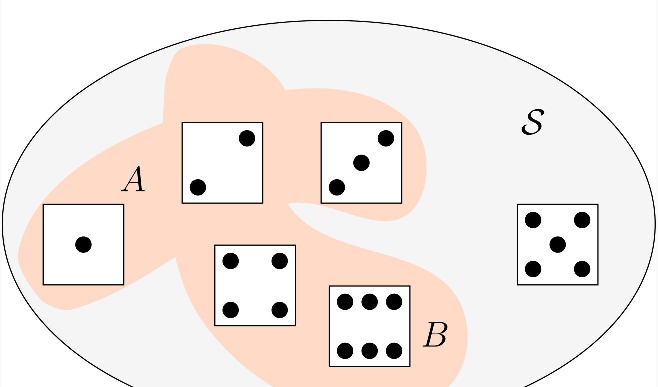
The result is `numeric(0)`, indicating that the intersection is empty. This means B and C are mutually exclusive.

Probability of the Union of Two Events

Let's explore the probability of the union of two events A and B in the context of our visual examples:

- $A = \{1, 2, 3\}$
- $B = \{2, 4, 6\}$

Figure 4 visualizes this:



Avoiding Double Counting

To compute $P(A \cup B)$, we must avoid double-counting outcomes in $A \cap B$:

- $P(A \cup B) = P(A) + P(B) - P(A \cap B)$
- If A and B are mutually exclusive, $A \cap B = \emptyset$, so:
- $P(A \cup B) = P(A) + P(B)$

Mutual exclusivity ensures no double counting, allowing probabilities to be added directly.

Using an LLM to Deepen Your Understanding of Set Theory in Probability

Using an LLM to Deepen Your Understanding of Set Theory in Probability

An LLM can be a powerful tool to explore concepts and solidify your understanding.

Here are some ways to use it effectively.

Ask for Clarifications

If a concept isn't clear, ask the LLM for explanations or examples.

Example Prompt

"What is the difference between the union and intersection of sets in probability? Can you give examples?"

Follow-Up Prompt

"Can you compare this to a real-life scenario, like rolling a die or flipping a coin?"

Generate Additional Examples

Use the LLM to create new examples similar to those in the lecture.

💡 Example Prompt

"Give me an example of mutually exclusive events involving sports outcomes."

💡 Example Prompt

"Can you show a sample space and events for tossing two coins?"

Simulate Visualizations and Code Interpretation

While the LLM doesn't directly create visuals, you can ask it to describe diagrams or R outputs.

Example Prompt

"Describe what a Venn diagram looks like for $A \cup B$, $A \cap B$, and $A \setminus B$."

Example Prompt

"What does the R function `union(A, B)` compute? How is it related to $A \cup B$?"

Practice Applying Definitions

Test your understanding by quizzing yourself using the LLM.

Example Prompt

“Ask me questions about the definitions of sample spaces, union, intersection, and complement.”

Example Prompt

“Give me a scenario and ask which set operation applies.”

Explore Real-World Applications

Learn how set theory applies beyond the lecture.

Example Prompt

"How is the concept of set intersection used in data science or finance?"

Example Prompt

"Explain how mutually exclusive events are important in designing experiments."

Learn R Through Step-by-Step Guidance

If you're new to R, ask the LLM to guide you through coding step by step.

Example Prompt

"Explain how to use `setdiff()` in R with an example involving dice rolls."

Follow-Up Prompt

"How does this output relate to the complement of a set?"

Dive Deeper into Probability Rule 3

Clarify how probabilities relate to set operations.

Follow-Up Prompt

“Explain why $P(A \cup B) = P(A) + P(B) - P(A \cap B)$.”

Follow-Up Prompt

“Can you provide a numerical example to illustrate this rule?”

Simulate Discussions

Ask the LLM to take the role of a peer or instructor for a simulated conversation.

Example Prompt

"Pretend you are my study partner. Let's discuss the complement of events and its significance in probability."

Engage with the Material

By actively engaging with the LLM through these kinds of prompts, you can: - Practice concepts - Explore applications - Deepen your understanding of the material

Try it alone or with your group!

Probability and Frequency

Probability and Frequency

The **frequency interpretation of probability** connects observed frequencies with theoretical probabilities:

$$P(A) = \frac{\text{Number of times } A \text{ occurs in repeated trials}}{\text{Total number of trials}}$$

This practical approach is intuitive and widely used but raises questions about the relationship between observed frequencies and theoretical probabilities.

Historical Context: Frequency and Probability

- 17th-century philosophers like Leibniz and Bernoulli explored connections between frequency and probability.
- Jacob Bernoulli's Weak Law of Large Numbers (WLLN) formalized this connection and became a cornerstone of probability theory.

The Weak Law of Large Numbers

Weak Law of Large Numbers

As the number of independent and identically distributed (i.i.d.) trials increases, the relative frequency of an event converges to its true probability with high probability.

Understanding the Weak Law

What the WLLN says:

1. Frequencies approximate probabilities over many trials.
2. Convergence occurs with high likelihood as trials increase.

What the WLLN does **not** say:

1. Frequencies **are not** probabilities.
2. Exact convergence is not guaranteed in finite samples—it describes long-run behavior.

Demonstrating the Weak Law: Coin Toss Example

Consider tossing a fair coin where $P(\text{Heads}) = 0.5$.

- Few tosses (e.g., 10): Frequencies may deviate significantly (e.g., 60% Heads).
- Many tosses (e.g., 10,000): Frequencies approach 50%.

Visualizing the Weak Law with R

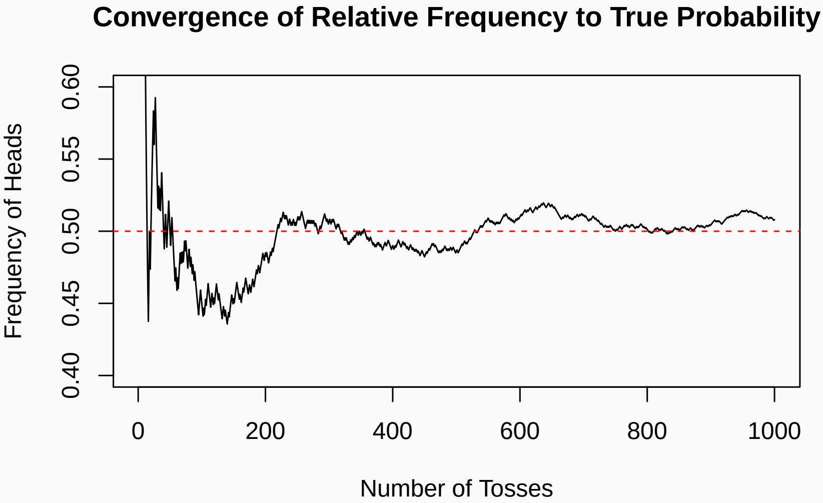
```r
# Define the coin
coin <- 0:1

# Toss the coin n times
n <- 1000
results <- replicate(n, sample(coin, size = 1))

# Calculate cumulative frequency of Heads
heads_freq <- cumsum(results == 1) / (1:n)

# Plot the convergence
plot(1:n, heads_freq, type = "l", ylim = c(0.4, 0.6),
     xlab = "Number of Tosses", ylab = "Frequency of Heads",
     main = "Convergence of Relative Frequency to True
     ↪  Probability")
abline(h = 0.5, col = "red", lty = 2)
```

Visualizing the Weak Law with R



Independence

The WLLN assumes **independence**, where the occurrence of one event does not affect the probability of another.

Intuition Behind Independence

Two events are **independent** if: - Knowing one event occurs provides no information about the likelihood of the other.

Example: Rolling a die twice:
- The first roll does not influence the outcome of the second roll.

Worked Example: Rolling a Die

Calculate the probability of rolling a 5 on the first roll and a 6 on the second roll:

$$P(5 \cap 6) = P(5) \times P(6) = \frac{1}{6} \times \frac{1}{6} = \frac{1}{36}$$

This uses the **multiplication rule for independent events**.

Formal Definition of Independence

Independence

Two events A and B are **independent** if:

$$P(A \cap B) = P(A) \times P(B)$$

Key Insights About Independence

1. Independence allows the use of the multiplication rule:

$$P(A \cap B) = P(A) \times P(B)$$

2. Independence must be verified—it cannot be assumed just because probabilities multiply.
3. Independence underpins the WLLN and many other probabilistic frameworks.

Some More Concepts from R

Some More Concepts from R

In this section, we'll explore important R concepts and apply them to analyze stock price data:

- Reading data
- R objects and atomic vectors
- Simulating stock price movements

Reading Data in R

Data is central to financial analysis. Let’s load and inspect a dataset of Apple stock prices using R.

Reading CSV Files

To load a CSV file, use the `read.csv()` function:

```r
aapl_prices <- read.csv("../data/aapl_prices.csv")
```

Ensure the file path is correct. Use `getwd()` to check your working directory:

```r
getwd()
```

```
[1] "/home/martinsummer/Code/R/Probability_Introduction/slide
```

Inspecting the Data

Once loaded, inspect the dataset using:

```r
head(aapl_prices, n = 10)
```

```
    X symbol       date     open     high      low    close
1   1   aapl 2000-01-03 0.936384 1.004464 0.907924 0.999442
2   2   aapl 2000-01-04 0.966518 0.987723 0.903460 0.915179
3   3   aapl 2000-01-05 0.926339 0.987165 0.919643 0.928571
4   4   aapl 2000-01-06 0.947545 0.955357 0.848214 0.848214
5   5   aapl 2000-01-07 0.861607 0.901786 0.852679 0.888393
6   6   aapl 2000-01-10 0.910714 0.912946 0.845982 0.872768
7   7   aapl 2000-01-11 0.856585 0.887277 0.808036 0.828125
8   8   aapl 2000-01-12 0.848214 0.852679 0.772321 0.778460
9   9   aapl 2000-01-13 0.843610 0.881696 0.825893 0.863839 1
10 10   aapl 2000-01-14 0.892857 0.912946 0.887277 0.896763
    adjusted
```

R Objects and Stock Price Movements

In R, most data structures are built from **atomic vectors**, the simplest R objects.

Atomic Vectors: Stock Price Changes

Daily stock price changes can be represented as an atomic vector:

```r
price_changes <- c(-1, 0, 1)
```

Check if it’s an atomic vector:

```r
is.vector(price_changes)
```

```
[1] TRUE
```

Properties of Atomic Vectors

1. **Length:** Use `length()` to determine the number of elements:

```
length(price_changes)
```

```
[1] 3
```

2. **Data Type:** Use `typeof()` to check the type of data:

```
typeof(price_changes)
```

```
[1] "double"
```

Simulating Stock Price Movements

Simulate a week of stock price movements with `sample()`:

```r
week_movements <- sample(price_changes,
                         size = 7, replace = TRUE)
week_movements
```

```
[1]  1 -1  0  0  1 -1  0
```

Weighted Simulations

Add probabilities to simulate scenarios with unequal likelihoods:

```r
week_movements_weighted <- sample(price_changes,
                                  size = 7,
                                  replace = TRUE,
                                  prob = c(0.3, 0.4, 0.3))
week_movements_weighted
```

```
[1] -1  0  0  1 -1  1 -1
```

Here, probabilities represent the likelihood of a decrease, no change, or an increase.

More about Atomic Vectors

R supports six types of atomic vectors:

1. **Double**: Numeric data with decimal precision.
2. **Integer**: Whole numbers.
3. **Character**: Text strings.
4. **Logical**: Boolean values (`TRUE`, `FALSE`).
5. **Complex**: Numbers with imaginary components (not covered here).
6. **Raw**: Binary data (not covered here).

Logical Vectors: Stock Analysis

Logical vectors store TRUE or FALSE values.

Example: Identify days when the closing price was higher than the opening price:

```r
aapl_prices$up_day <- aapl_prices$close > aapl_prices$open
head(aapl_prices$up_day, n = 10)
```

```
 [1]  TRUE FALSE  TRUE FALSE  TRUE FALSE FALSE FALSE  TRUE  T
```

Factors: Categorical Data in R

Factors represent categorical data.

Example: Representing weekdays:

```
days <- factor(c("Monday", "Tuesday",
                 "Wednesday", "Thursday", "Friday"))
days
```

```
[1] Monday    Tuesday   Wednesday Thursday  Friday
Levels: Friday Monday Thursday Tuesday Wednesday
```

Factors can also be ordered:

```
ordered_days <- factor(days, levels =
  c("Monday", "Tuesday", "Wednesday",
    "Thursday", "Friday"), ordered = TRUE)
ordered_days
```

```
[1] Monday    Tuesday   Wednesday Thursday  Friday
```

Factors in Stock Analysis: Adding Weekdays and Price Changes

To analyze Apple stock data, we can add weekday and price change factors:

```r
aapl_prices$date <- as.Date(aapl_prices$date)
aapl_prices$weekday <- factor(weekdays(aapl_prices$date),
  levels = c("Monday", "Tuesday",
  "Wednesday", "Thursday", "Friday"))

# Calculate daily price changes
price_diff <- aapl_prices$close - aapl_prices$open
aapl_prices$price_change <-
  factor(ifelse(price_diff > 0, "up",
         ifelse(price_diff < 0, "down", "unchanged")))
```

Tabulating Data with Factors

Factors make it easy to tabulate categorical data:

```
tabulated_data <- table(aapl_prices$weekday,
                        aapl_prices$price_change)
tabulated_data
```

```
            down unchanged  up
  Monday     549         4 676
  Tuesday    644         6 693
  Wednesday  632         2 709
  Thursday   650         1 666
  Friday     678         4 632
```

Example: Find the number of down moves on Mondays:

```
down_on_mondays <- tabulated_data["Monday", "down"]
down_on_mondays
```

Data Frames: Organizing Data

A **data frame** organizes data into rows and columns.

Example: Apple stock price dataset:

```r
head(aapl_prices)
```

```
  X symbol       date     open     high      low    close
1 1   aapl 2000-01-03 0.936384 1.004464 0.907924 0.999442 535
2 2   aapl 2000-01-04 0.966518 0.987723 0.903460 0.915179 512
3 3   aapl 2000-01-05 0.926339 0.987165 0.919643 0.928571 778
4 4   aapl 2000-01-06 0.947545 0.955357 0.848214 0.848214 767
5 5   aapl 2000-01-07 0.861607 0.901786 0.852679 0.888393 460
6 6   aapl 2000-01-10 0.910714 0.912946 0.845982 0.872768 505
  up_day   weekday price_change
1   TRUE    Monday           up
2  FALSE   Tuesday         down
3   TRUE Wednesday           up
```

Example: Adding a Logical Column to a Data Frame

We can add a logical column to indicate days when the stock closed higher:

```r
aapl_prices$up_day <-
  aapl_prices$close > aapl_prices$open
head(aapl_prices[c("date", "close", "up_day")], n = 5)
```

```
        date    close up_day
1 2000-01-03 0.999442   TRUE
2 2000-01-04 0.915179  FALSE
3 2000-01-05 0.928571   TRUE
4 2000-01-06 0.848214  FALSE
5 2000-01-07 0.888393   TRUE
```

Subsetting Data Frames

You can filter rows or subset a data frame.

Example: Extract rows where the stock closed higher than it opened:

```
higher_close <- aapl_prices[aapl_prices$up_day == TRUE, ]
head(higher_close)
```

```
    X symbol       date     open     high      low    close
1   1   aapl 2000-01-03 0.936384 1.004464 0.907924 0.999442
3   3   aapl 2000-01-05 0.926339 0.987165 0.919643 0.928571
5   5   aapl 2000-01-07 0.861607 0.901786 0.852679 0.888393
9   9   aapl 2000-01-13 0.843610 0.881696 0.825893 0.863839 1
10 10   aapl 2000-01-14 0.892857 0.912946 0.887277 0.896763
11 11   aapl 2000-01-18 0.901786 0.946429 0.896763 0.928013
    adjusted up_day   weekday price_change
1  0.8392805   TRUE    Monday           up
3  0.7797668   TRUE Wednesday           up
```

Lists: Combining Multiple Data Types

Lists: Combining Multiple Data Types

Lists in R allow grouping different types of objects and structures, making them ideal for handling heterogeneous data.

Example: Creating a List

Summarize key information about Apple's stock prices:

```r
stock_summary <- list(
  ticker = "AAPL",
  price_summary = summary(aapl_prices$close),
  highest_price = max(aapl_prices$high, na.rm = TRUE),
  date_range = range(aapl_prices$date, na.rm = TRUE)
)
stock_summary
```

```
$ticker
[1] "AAPL"


$price_summary
```

Nested Lists

Lists can also contain nested lists or data frames:

```r
nested_list <- list(
  summary = stock_summary,
  recent_data = head(aapl_prices, n = 5)
)
nested_list
```

```
$summary
$summary$ticker
[1] "AAPL"

$summary$price_summary
    Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
  0.2343   2.5631  18.5564  50.2000  54.0862 286.1900

$summary$highest_price
```

Factors, Data Frames, and Lists in Practice

Factors, Data Frames, and Lists in Practice

We'll summarize Apple's weekly stock price movements using a combination of:

- **Factors**: Categorize days of the week.
- **Data Frames**: Organize daily price changes.
- **Lists**: Store structured summaries.

Weekly Summary Example

```r
# Extract the first five rows of data
weekly_data <- head(aapl_prices, 5)
# Add a factor for days of the week
weekly_data$day <- factor(
  c("Monday", "Tuesday", "Wednesday",
    "Thursday", "Friday"),
  levels = c("Monday", "Tuesday",
             "Wednesday", "Thursday", "Friday"),
  ordered = TRUE
)
# Simulate price changes
set.seed(42)
weekly_data$price_change <- sample(
  c(-1, 0, 1),
  size = nrow(weekly_data),
  replace = TRUE,
  prob = c(0.3, 0.4, 0.3)
)
```

Creating a Summary List

Summarize the data in a list:

```
summary_list <- list(
  week_data = weekly_data,
  positive_days = sum(weekly_data$price_change > 0),
  total_change = sum(weekly_data$price_change)
)
summary_list
```

```
$week_data
  X symbol       date     open     high      low    close
1 1   aapl 2000-01-03 0.936384 1.004464 0.907924 0.999442 535
2 2   aapl 2000-01-04 0.966518 0.987723 0.903460 0.915179 512
3 3   aapl 2000-01-05 0.926339 0.987165 0.919643 0.928571 778
4 4   aapl 2000-01-06 0.947545 0.955357 0.848214 0.848214 767
5 5   aapl 2000-01-07 0.861607 0.901786 0.852679 0.888393 460
  up_day   weekday price_change       day
```

Explanation of Steps

1. **Extract Weekly Data**: Use `head` to simulate one week of trading data.
2. **Add Days as Factors**: Represent days of the week with ordered factors.
3. **Simulate Price Movements**: Use `sample()` to generate daily price changes with specified probabilities.
4. **Create a Summary List**:
    - `week_data`: Holds the weekly data frame.
    - `positive_days`: Counts days with upward movements.
    - `total_change`: Sums net price changes.

Combining Structures for Financial Analysis

Combining Structures for Financial Analysis

By integrating factors, data frames, and lists, we can:

1. Categorize Data: Use factors to group and analyze.
2. Organize Tabular Data: Store structured datasets in data frames.
3. Integrate Diverse Data: Combine heterogeneous data in lists.

These tools are invaluable for real-world financial analysis.

Back to probability: Will Apple’s Stock Price Move Up or Down?

Back to probability: Will Apple’s Stock Price Move Up or Down?

Revisit the dataset:

```r
head(aapl_prices, n = 10)
```

```
    X symbol       date     open     high      low    close
1   1   aapl 2000-01-03 0.936384 1.004464 0.907924 0.999442
2   2   aapl 2000-01-04 0.966518 0.987723 0.903460 0.915179
3   3   aapl 2000-01-05 0.926339 0.987165 0.919643 0.928571
4   4   aapl 2000-01-06 0.947545 0.955357 0.848214 0.848214
5   5   aapl 2000-01-07 0.861607 0.901786 0.852679 0.888393
6   6   aapl 2000-01-10 0.910714 0.912946 0.845982 0.872768
7   7   aapl 2000-01-11 0.856585 0.887277 0.808036 0.828125
8   8   aapl 2000-01-12 0.848214 0.852679 0.772321 0.778460
9   9   aapl 2000-01-13 0.843610 0.881696 0.825893 0.863839 1
10 10   aapl 2000-01-14 0.892857 0.912946 0.887277 0.896763
    adjusted up_day   weekday price_change
```

Understanding the Structure of the Data

Confirm the type and structure of `aapl_prices`:

```r
typeof(aapl_prices)
```

```
[1] "list"
```

```r
class(aapl_prices)
```

```
[1] "data.frame"
```

```r
dim(aapl_prices)
```

```
[1] 6546   12
```

Key Insights:

- `aapl_prices` is a `data.frame` (class: list).
- Contains 6546 trading days.

Subsetting Data: Accessing Specific Elements

Subsetting Data: Accessing Specific Elements

To analyze the dataset, we extract specific values or subsets of data using:

```
aapl_prices[row_indices, column_indices]
```

Subsetting Methods in R

1. Positive Integers

   - Select the closing price on the first trading day:

```
aapl_prices[1, "close"]
```

```
[1] 0.999442
```

   - Select the first 5 closing prices:

```
aapl_prices[1:5, "close"]
```

```
[1] 0.999442 0.915179 0.928571 0.848214 0.888393
```

2. Negative Integers

- Exclude the first observation:

```r
head(aapl_prices[-1, "close"], 3)
```

```
[1] 0.915179 0.928571 0.848214
```

3. Zero

- Creates an empty object:

```
aapl_prices[0, 0]
```

```
data frame with 0 columns and 0 rows
```

4. Blank Spaces

- Select all values in a dimension:

```r
sel <- aapl_prices[, "close"]
length(sel)
```

```
[1] 6546
```

5. Logical Values

- Use a logical vector to select specific columns:

```r
aapl_prices[1, c(FALSE, FALSE, FALSE, TRUE, FALSE, TRUE, FALSE,
 ↪  FALSE)]
```

```
      open      low price_change
1 0.936384 0.907924           up
```

6. Names

- Select using column names:

```r
aapl_prices[1, "close"]
```

```
[1] 0.999442
```

Calculating Daily Price Differences

Calculating Daily Price Differences

Manual Calculation

Use indexing to compute differences:

```r
aux_1 <- aapl_prices[2:8044, "close"]
aux_2 <- aapl_prices[1:8043, "close"]
diff_close <- aux_1 - aux_2
head(diff_close, n = 10)
```

```
 [1] -0.08426297  0.01339197 -0.08035702  0.04017901 -
0.01562500 -0.04464298
 [7] -0.04966497  0.08537894  0.03292406  0.03125000
```

Using the `diff()` Function

Simplify calculations with the built-in **`diff()`** function:

```
aapl_prices$diff <- c(NA, diff(aapl_prices$close))
head(aapl_prices, n = 5)
```

```
  X symbol       date     open     high      low    close
1 1   aapl 2000-01-03 0.936384 1.004464 0.907924 0.999442 535
2 2   aapl 2000-01-04 0.966518 0.987723 0.903460 0.915179 512
3 3   aapl 2000-01-05 0.926339 0.987165 0.919643 0.928571 778
4 4   aapl 2000-01-06 0.947545 0.955357 0.848214 0.848214 767
5 5   aapl 2000-01-07 0.861607 0.901786 0.852679 0.888393 460
  up_day   weekday price_change        diff
1   TRUE    Monday           up          NA
2  FALSE   Tuesday         down -0.08426297
3   TRUE Wednesday           up  0.01339197
4  FALSE  Thursday         down -0.08035702
5   TRUE    Friday           up  0.04017901
```

Frequency-Based Probability of Upward Moves

Frequency-Based Probability of Upward Moves

Create a Logical Column

Indicate whether the price difference is positive:

```r
aapl_prices$diff_pos <- aapl_prices$diff > 0
head(aapl_prices, n = 5)
```

```
  X symbol       date     open     high      low    close
1 1   aapl 2000-01-03 0.936384 1.004464 0.907924 0.999442 535
2 2   aapl 2000-01-04 0.966518 0.987723 0.903460 0.915179 512
3 3   aapl 2000-01-05 0.926339 0.987165 0.919643 0.928571 778
4 4   aapl 2000-01-06 0.947545 0.955357 0.848214 0.848214 767
5 5   aapl 2000-01-07 0.861607 0.901786 0.852679 0.888393 460
  up_day   weekday price_change        diff diff_pos
1   TRUE    Monday           up          NA       NA
2  FALSE   Tuesday         down -0.08426297    FALSE
3   TRUE Wednesday           up  0.01339197     TRUE
```

Calculate the Probability

Use relative frequency to compute the probability of an upward move:

```r
mean(aapl_prices$diff_pos, na.rm = TRUE)
```

```
[1] 0.5220779
```

The probability is approximately 52%.

Understanding the Calculation

Understanding the Calculation

Key Points:

1. **Type Coercion**: Logical values are converted to numerical values:
    - `TRUE` → `1`
    - `FALSE` → `0`
2. **Vectorized Operations**:
    - `mean()` calculates the proportion of `TRUE` values.
    - Equivalent to:

```r
sum(aapl_prices$diff_pos, na.rm = TRUE) /
sum(!is.na(aapl_prices$diff_pos))
```

```
[1] 0.5220779
```

3. **Flexibility**:
    - `mean()` directly computes meaningful results without loops or additional transformations.

This showcases the power of R's vectorized operations for analyzing

Applying Probability Concepts

Applying Probability Concepts

1. Probability of Consecutive Increases

What is the probability that the stock price increases every day over a week (5 trading days)?

$$P(U \cap U \cap U \cap U \cap U) = P(U)^5 = 0.51^5 = 0.035$$

2. Probability of One Decrease and Four Increases

The probability of one decrease and four increases is:

$$P(D \cap U \cap U \cap U \cap U) = 0.49 \cdot 0.51^4 = 0.033$$

Since there are 5 such scenarios, the total probability is:

$$5 \cdot 0.033 = 0.132$$

Reflecting on Assumptions

Reflecting on Assumptions

Are up and down moves truly independent?

Historical Context:

- **Louis Bachelier (1870–1946)**: Pioneered the study of stock price randomness.
- **Paul Samuelson (1965)**: Proposed that randomness arises from traders' rational behavior.

Random Walk Hypothesis:

- Stock prices behave like a random walk.
- If true, the probability of an up or down move is $\frac{1}{2}$, with frequencies converging to this value over time.

Challenges to the Random Walk Hypothesis

Research (e.g., Lo and MacKinlay (2019)) suggests: - Stock prices are **not completely random**. - Predictability exists to some degree.

Takeaway: Probability models are tools, not absolute truths. Always analyze assumptions critically and understand the context.

Benford’s Law and Trading Volumes

Benford’s Law and Trading Volumes

Leading Digits in Real-World Data

The leading digit of a number is its first non-zero digit. Examples: - $7829 \to 7$ - $0.00453 \to 4$ - $10892 \to 1$

Benford’s Law predicts:

$$P(d) = \log_{10}\left(1 + \frac{1}{d}\right),\ d \in \{1, 2, \ldots, 9\}$$

Smaller digits like 1 occur more frequently than larger ones like 9.

Applying Benford’s Law to Trading Volumes

Extract Leading Digits

Filter valid trading volumes and extract leading digits:

```r
volumes <- aapl_prices$volume
valid_volumes <- volumes[volumes > 0 & !is.na(volumes)]
leading_digits <- as.numeric(
  substr(as.character(valid_volumes), 1, 1))
```

Empirical vs. Theoretical Frequencies

Compute empirical frequencies and compare to Benford's Law:

```r
emp_freq <- table(leading_digits) / length(leading_digits)
benford <- data.frame(
  Digit = 1:9,
  Empirical_Freq = as.numeric(emp_freq[1:9]),
  Benford_Prob = log10(1 + 1 / (1:9))
)
knitr::kable(benford)
```

Digit	Empirical_Freq	Benford_Prob
1	0.2598533	0.3010300
2	0.1374885	0.1760913
3	0.1232814	0.1249387
4	0.1208372	0.0969100
5	0.0898258	0.0791812
6	0.0803544	0.0669468

Interpretation and Broader Applications

Insights: - The empirical frequencies align closely with Benford’s Law. - Benford’s Law applies to datasets spanning multiple magnitudes.

Applications: - Detecting anomalies in tax filings and financial records. - Validating the authenticity of datasets.

Summary

Summary

Probability Concepts

- Definitions of sample space, basic outcomes, and events.
- The Weak Law of Large Numbers: Connecting empirical frequencies to theoretical probabilities.
- Independence: $P(A \cap B) = P(A) \cdot P(B)$.

R Concepts

- Subsetting data:
    - Positive/negative indices, logical values, and names.
- Computing differences and probabilities.
- Analyzing empirical distributions with Benford's Law.

Applications

- Simulating random experiments with R.
- Analyzing stock price movements.
- Validating datasets with Benford's Law.

Probability theory and R tools provide a powerful framework for analyzing uncertainty and real-world data. Always combine theory with critical thinking to draw meaningful conclusions.

Project 2: Financial Data Forensics – Investigating Financial Reports Using Benford’s Law

Project 2: Financial Data Forensics – Investigating Financial Reports Using Benford's Law

Overview

In this project, you will:

- Analyze financial data for conformity to **Benford's Law**.
- Detect and interpret anomalies in revenues and expenditures.
- Reinforce your understanding of probabilities and empirical analysis.

Objectives

By the end of this project, you will:

1. Analyze the distribution of leading digits in revenues and expenditures.
2. Compare empirical distributions to Benford's Law.
3. Identify deviations and hypothesize their causes.
4. Reflect on implications for financial forensics.

Steps to Complete the Project

Step 1: Understand the Research Question

Your main tasks: 1. Determine if the leading digits in revenues and expenditures follow Benford's Law. 2. Interpret deviations in expenditure data—possible signs of fraud or manipulation.

Step 2: Obtain and Inspect the Dataset

1. **Download the Dataset**:
   - File: `company_financials.csv`
   - Contains simulated data for revenues and expenditures of 200 companies, with subtle anomalies.
2. **Inspect the Data**:
   - Load the dataset in R.
   - Use functions like `head()`, `summary()`, and `str()` to understand:
     - `CompanyID`: Unique identifier for each company.
     - `Revenue`: Revenue of the company (in dollars).
     - `Expenditure`: Expenditure of the company (in dollars).

Step 3: Prepare the Data

1. **Filter Valid Data**:
    - Exclude:
        - Non-positive values (e.g., 0 or negative numbers).
        - Missing values (`NA`).
2. **Extract Leading Digits**:
    - Use string manipulation in R to extract the first digit from each valid value.

Step 4: Analyze the Data

1. **Compute Empirical Frequencies**:
   - Tabulate the leading digits for revenues and expenditures.
2. **Compare with Benford's Law**:
   - Create data frames showing:
     - Empirical frequencies.
     - Theoretical probabilities from Benford's Law.
3. **Visualize the Results**:
   - Plot bar charts comparing distributions.

Step 5: Interpret the Results

1. **Evaluate Conformity**:
   - Does the revenue data follow Benford's Law?
   - Do expenditures deviate significantly?
2. **Hypothesize Causes**:
   - Rounded or artificial values?
   - Fraud or anomalies in expenditure data?
3. **Relate to Probabilities**:
   - Discuss how large sample sizes enhance reliability.

Ready to Detect Anomalies?

Ready to Detect Anomalies?

Dive into the project, and let Benford's Law guide your forensic investigation of financial data!

Lo, Andrew, and Craig MacKinlay. 2019. *A Non-Random Walk down Wallstreet.* Princeton University Press.