An Introduction to Probability

Conditional Probability

Martin Summer

18 January, 2026

Conditional Probability

Conditional Probability

- **Conditional probability** is a basic tool of probability theory.
- Particularly relevant in **Finance** for analyzing dependencies and risk.
- Often obscured by complex terminology despite simple ideas.

Motivation: A Striking Scenario

Motivation: A Striking Scenario

- Imagine evaluating the safety of a **bond portfolio**:
  - Bonds are highly rated and diversified.
  - During a global recession:
    - Defaults occur, unraveling the portfolio.
    - Losses mount unexpectedly.
- Key lesson: **Underestimating event connections** leads to catastrophic risks.
- **Conditional probability** enables us to model dependencies effectively.

Why Conditional Probability Matters

Why Conditional Probability Matters

- Mastering conditional probability is crucial for:
    - Pricing financial instruments.
    - Assessing credit risk.
    - Making informed investment decisions.
- Neglecting it can lead to systemic failures:
    - Example: The **2007-2008 financial crisis**.

Case Study: The Financial Crisis (2007-2008)

Case Study: The Financial Crisis (2007-2008)

- Revealed the dangers of assuming independence between events.
- Highlighted failures in **structured finance**:
    - Underestimation of dependencies.
    - Misjudgment of risk profiles.
- Reference: For a comprehensive discussion, see [Tooze 2018].

Understanding Structured Finance

- **Bonds**:
  - Financial instruments with fixed payments and default risks.
  - Ratings by agencies like Moody's and Standard & Poor's:
    - High grade (AAA, AA) to speculative (BB, B) and default danger (CCC, C).

Rating Category	Moody's	Standard & Poor's
High grade	Aaa	AAA
	Aa	AA
Medium grade	A	A
	Baa	BBB
Speculative grade	Ba	BB
	B	B
Default danger	Caa	CCC
	Ca	CC
	C	C/D

Pooling and Tranching: An Innovation

Pooling and Tranching: An Innovation

- **Structured finance**:
  - Pools risky assets.
  - Divides cash flows into **tranches** with distinct risk profiles.
  - Enables creation of investment-grade securities from speculative-grade assets.
- Example products:
  - **Mortgage-backed securities (MBS)**.
  - Variants using similar financial engineering concepts.

Simplified Example

Simplified Example

- Based on Karl Schmedder's course on probability.
- Illustrates **structured finance** and its relation to probability.
- Develops an intuitive understanding of pooling, tranching, and conditional dependencies.

A Simple Event Tree for One Bond

A Simple Event Tree for One Bond

- Consider a single bond paying €1 at maturity in the future.
  - Default probability: 10% ($P(D) = 0.1$).
  - Non-default probability: 90% ($P(N) = 0.9$).
- Payoff structure:
  - No Default (N): €1.
  - Default (D): €0.

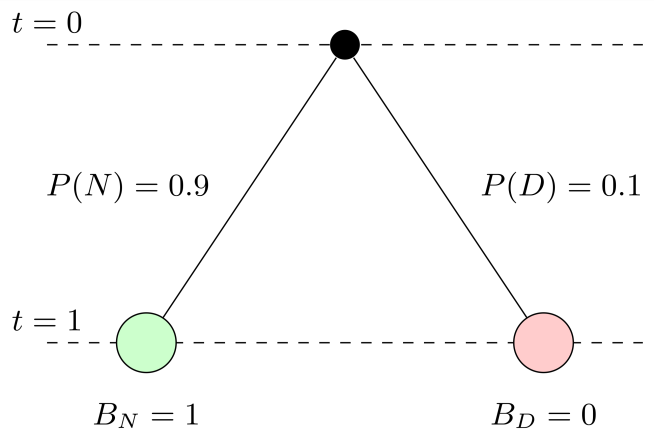


Figure 1

Understanding the Event Tree

Understanding the Event Tree

- Nodes represent states of the bond:
  - $t = 0$: Initial state.
  - $t = 1$: Outcomes (N or D).
- Probabilities:
  - $P(N) = 0.9$.
  - $P(D) = 0.1$.
- Analogy:
  - Coin toss with unequal probabilities:
    - Heads: 90% (No Default).
    - Tails: 10% (Default).
- Probabilistic interpretation:
  - Random experiment with outcomes N and D.

Combining Two Bonds: Independence Assumption

Combining Two Bonds: Independence Assumption

- Portfolio of two bonds.
- Independence Assumption:
  - Defaults occur independently.
  - Default of one bond does not influence the other.
- Simplifies calculations:
  - Default probabilities remain uncorrelated.
  - Example: $P(D_1 \cap D_2) = P(D_1) \cdot P(D_2)$.
- Historically justified by:
  - Diversification.
  - Uncorrelated defaults under normal conditions.

Systemic Risks: Challenges to Independence

Systemic Risks: Challenges to Independence

- Systemic risks disrupt independence:
    - Defaults become correlated during crises.
    - Shared macroeconomic factors increase joint defaults.
- Example: 2008 financial crisis:
    - Rising mortgage defaults driven by economic downturn.
    - Increased correlation in bond defaults.
- Critical thinking reveals:
    - Diversification alone cannot guarantee safety.
    - Assumption of independence is fragile.

Event Tree for Two Bonds

Event Tree for Two Bonds

- Two independent bonds:
    - Combine event trees of individual bonds.
    - Visualized as a double event tree.

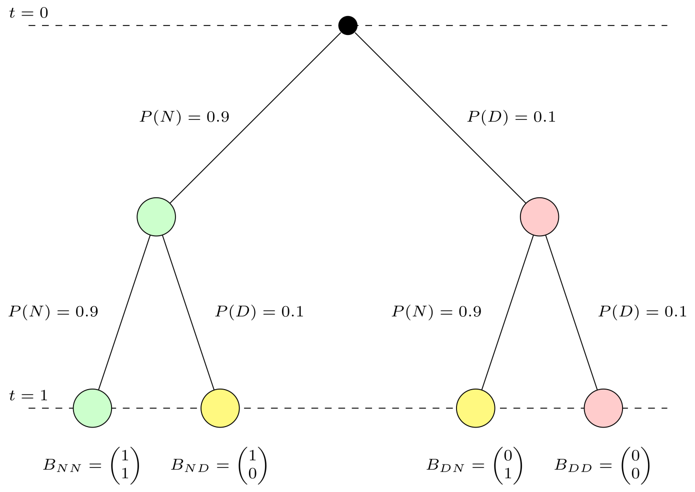

$t = 0$
$P(N) = 0.9$
$P(D) = 0.1$
$P(N) = 0.9$
$P(D) = 0.1$
$P(N) = 0.9$
$P(D) = 0.1$
$t = 1$
$B_{NN} = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$
$B_{ND} = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$
$B_{DN} = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$
$B_{DD} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$

Figure 2

- Outcomes:
  - Each path represents a combination of defaults or no defaults.
  - Example: $B_{NN} = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$ (both bonds pay €1).

Pooling and Tranching: Independent Risks

Pooling and Tranching: Independent Risks

- Re-engineering risk profiles:
  - Pool payoffs of two bonds.
  - Create two new securities:
    1. Pays €1 except when both bonds default.
    2. Pays €0 except when both bonds do not default.
- Probabilities:
  - Both bonds default: $P(D_1 \cap D_2) = 0.1 \cdot 0.1 = 0.01$.
  - Both bonds do not default: $P(N_1 \cap N_2) = 0.9 \cdot 0.9 = 0.81$.

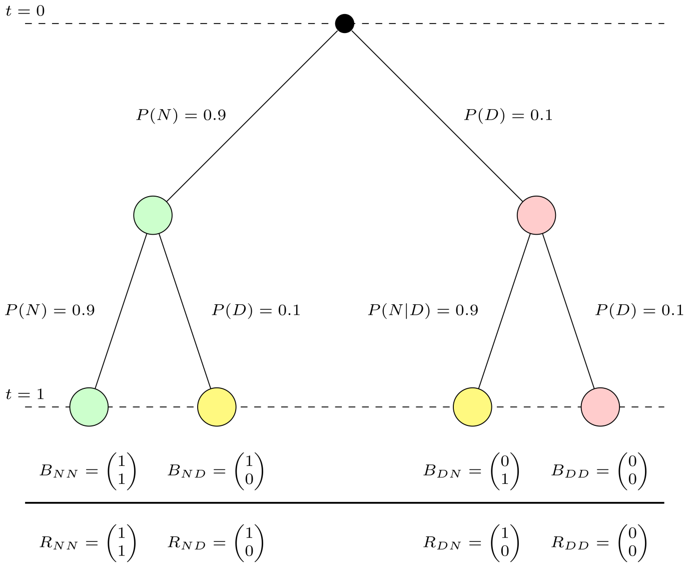
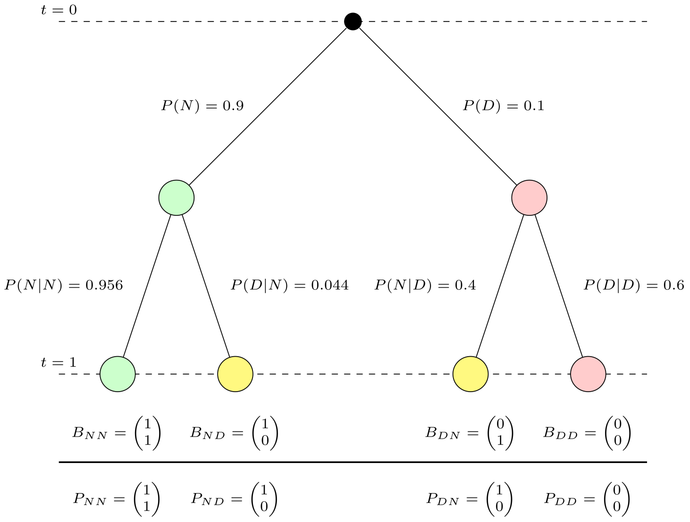

$t = 0$
$P(N) = 0.9$
$P(D) = 0.1$
$P(N) = 0.9$
$P(D) = 0.1$
$P(N|D) = 0.9$
$P(D) = 0.1$
$t = 1$
$B_{NN} = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$ $B_{ND} = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$ $B_{DN} = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$ $B_{DD} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$
$R_{NN} = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$ $R_{ND} = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$ $R_{DN} = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$ $R_{DD} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$

Figure 3

Pooling and Tranching: Dependent Risks

Pooling and Tranching: Dependent Risks

- What happens if independence does not hold?
  - Default probabilities change:
    - $P(D_2 \mid D_1) = 0.6$.
    - $P(D_2 \mid N_1) = 0.044$.
- Increased correlation during systemic events:
  - Shared risks drive joint defaults.



$t = 0$
$P(N) = 0.9$
$P(D) = 0.1$
$P(N|N) = 0.956$
$P(D|N) = 0.044$
$P(N|D) = 0.4$
$P(D|D) = 0.6$
$t = 1$
$B_{NN} = \begin{pmatrix}1\\1\end{pmatrix}$
$B_{ND} = \begin{pmatrix}1\\0\end{pmatrix}$
$B_{DN} = \begin{pmatrix}0\\1\end{pmatrix}$
$B_{DD} = \begin{pmatrix}0\\0\end{pmatrix}$
$P_{NN} = \begin{pmatrix}1\\1\end{pmatrix}$
$P_{ND} = \begin{pmatrix}1\\0\end{pmatrix}$
$P_{DN} = \begin{pmatrix}1\\0\end{pmatrix}$
$P_{DD} = \begin{pmatrix}0\\0\end{pmatrix}$

Figure 4

Impact of Dependence on Risk Profiles

Impact of Dependence on Risk Profiles

- Dependent risks change probabilities:
    - Both bonds default: $P(D_1) \cdot P(D_2 \mid D_1) = 0.1 \cdot 0.6 = 0.06$.
    - Six times higher than under independence.
- Structured finance products fail:
    - Investment-grade tranches lose their safety.
    - Junk plus junk remains junk.

Lessons from Structured Finance

Lessons from Structured Finance

1. **Diversification**:
    - Assets must be from independent sectors.
2. **Macroeconomic Stability**:
    - Low systemic risk is crucial.
3. **Transparent Modeling**:
    - Dependencies must be accounted for.

- Neglecting these led to flawed models and systemic failures during the 2008 crisis.

Conditional Probability

Conditional Probability

- Conditional probability formalizes how the probability of one event changes when another event is known to occur.
- It provides a framework for understanding dependencies quantitatively.

Definition: Conditional Probability

Definition: Conditional Probability

Let B be an event with positive probability. For an arbitrary event A, the **conditional probability** of A given B is:

$$P(A \mid B) = \frac{P(A \cap B)}{P(B)}, \quad \text{provided } P(B) \neq 0$$

Undefined Conditional Probabilities

Undefined Conditional Probabilities

- Conditional probabilities are **undefined** when the conditioning event B has $P(B) = 0$.
- This distinction is:
  - Irrelevant for **discrete sample spaces**.
  - Crucial in the **general theory**.

Clarifying Conditional Probabilities

Clarifying Conditional Probabilities

- Conditional probabilities represent a **notation** change:
  - Probabilities adjust to reflect known conditions.
- Example: Revisit the financial crisis scenario:
  - Highlighted how dependencies can amplify systemic risk.

The Probability Tree and Conditional Probabilities

The Probability Tree and Conditional Probabilities

- A **probability tree** is labeled with **edge probabilities**:
  - Representing marginal and conditional probabilities at each level.
  - At $t = 0$
    - $P(B_1 = N) = 0.9$, $P(B_1 = D) = 0.1$
  - At $t = 1$
    - $P(B_2 = N \mid B_1 = N) = 0.956$,
      $P(B_2 = D \mid B_1 = N) = 0.044$
    - $P(B_2 = N \mid B_1 = D) = 0.4$, $P(B_2 = D \mid B_1 = D) = 0.6$.

Defining Probabilities in R

Defining Probabilities in R

- Probabilities defined from the **probability tree**:

```
# Define the probabilities

# Marginal probabilities for B_1
P_N <- 0.9  # Probability that B_1 does not default
P_D <- 0.1  # Probability that B_1 defaults

# Conditional probabilities for B_2 given B_1
P_N_given_N <- 0.8604/0.9
# B_2 does not default given B_1 does not default
P_D_given_N <- 0.0396/0.9
# B_2 defaults given B_1 does not default
P_N_given_D <- 0.4
# B_2 does not default given B_1 defaults
P_D_given_D <- 0.6
# B_2 defaults given B_1 defaults
```

Computing Joint Probabilities

Computing Joint Probabilities

- Joint probabilities, $P(A \cap B)$, are calculated using the **multiplication rule**:

$$P(A \cap B) = P(A \,|\, B) \cdot P(B).$$

```
# Calculate joint probabilities

P_NN <- P_N * P_N_given_N  # Both bonds do not default
P_ND <- P_N * P_D_given_N  # B_1 does not default, B_2 defaults
P_DN <- P_D * P_N_given_D  # B_1 defaults, B_2 does not default
P_DD <- P_D * P_D_given_D  # Both bonds default
```

Simulating Bond Portfolio Defaults

Goal:

- Simulate a bond portfolio with two types of bonds (`B1` and `B2`).
- Reproduce a portfolio where default probabilities align with the given contingency table.

Key Concepts:

- Joint probabilities from the contingency table.
- Unconditional and conditional probabilities.

Set up simulation parameters

Set up simulation parameters

```r
N <- 5000  # Total number of bonds
P_DD <- 0.06  # P(X = D, Y = D)
P_DN <- 0.04  # P(X = D, Y = N)
P_ND <- 0.04  # P(X = N, Y = D)
P_NN <- 0.86  # P(X = N, Y = N)

# Verify probabilities sum to 1
stopifnot(abs(P_DD + P_DN + P_ND + P_NN - 1) < 1e-6)
```

```r
# Simulate joint outcomes based on the contingency table
simulate_defaults <- function(N, probs) {
  sample(
    c("DD", "DN", "ND", "NN"),
    size = N,
    replace = TRUE,
    prob = probs
  )
}
```

```r
# Generate portfolio data
portfolio <- data.frame(
  BondID = 1:N,
  BondType = sample(c("B1", "B2"), N,
                    replace = TRUE, prob = c(0.5, 0.5))
)

# Assign joint default outcomes
portfolio$JointOutcome <-
  simulate_defaults(N, c(P_DD, P_DN, P_ND, P_NN))
portfolio$X_Defaulted <-
  portfolio$JointOutcome %in% c("DD", "DN")
portfolio$Y_Defaulted <-
  portfolio$JointOutcome %in% c("DD", "ND")
```

```r
# Compute unconditional probabilities
P_X_D <- mean(portfolio$X_Defaulted)  # P(X = D)
P_Y_D <- mean(portfolio$Y_Defaulted)  # P(Y = D)

# Compute conditional probabilities
P_X_given_Y_D <-
  mean(portfolio$X_Defaulted[portfolio$Y_Defaulted])
# P(X = D | Y = D)
P_Y_given_X_D <-
  mean(portfolio$Y_Defaulted[portfolio$X_Defaulted])
# P(Y = D | X = D)
```

```r
# Display results
cat("Unconditional Probabilities:\n")
cat("P(X = D):", round(P_X_D, 4), "\n")
cat("P(Y = D):", round(P_Y_D, 4), "\n\n")

cat("Conditional Probabilities:\n")
cat("P(X = D | Y = D):", round(P_X_given_Y_D, 4), "\n")
cat("P(Y = D | X = D):", round(P_Y_given_X_D, 4), "\n")
```

```r
# Verify calibration matches input probabilities
calibration_check <- table(portfolio$JointOutcome) / N
expected_probs <- c(P_DD, P_DN, P_ND, P_NN)

calibration_result <- data.frame(
  JointOutcome = names(calibration_check),
  Frequency = as.numeric(calibration_check),
  Expected = expected_probs
)

print(calibration_result)
```

Key Takeaways

- **Conditional Probability**:
  - Computed as the relative frequency in a subset of rows where the condition holds.
  - E.g., $P(X = D \,|\, Y = D)$ focuses only on rows where `Y_Defaulted` is `TRUE`.
- **Calibration**:
  - Simulated frequencies align closely with expected probabilities from the contingency table.
- **Practical Application**:
  - Demonstrates how dependency structures in default risks are modeled.

Advanced R Concepts

In this section, we explore advanced R programming concepts:

1. **Environments**: How R evaluates and stores variables.
2. **Scoping Rules**: How R resolves variable names.
3. **Closures**: Functions that retain the environment where they were created.

Environments

Environments

- An **environment** in R stores objects (variables, functions, etc.).
- The **global environment** stores user-created objects.
- Local variables can override global ones in specific functions.

Example: Global and Local Variables

```r
# Global interest rate
interest_rate <- 0.05

# Function to calculate interest
calculate_interest <- function(principal, rate = interest_rate) {
  interest <- principal * rate
  return(interest)
}

# Global calculation
global_interest <- calculate_interest(1000)

# Local override
local_interest <- calculate_interest(1000, rate = 0.07)

cat("Global Interest:", global_interest, "\n")
```

```
Global Interest: 50
```

```r
cat("Local Interest:", local_interest, "\n")
```

Scoping Rules

- R uses **lexical scoping** to find variables:
  - Searches the closest environment first.
  - Moves outward to enclosing environments.

Example: Nested Functions

```r
# Global default rates
default_rates <- c(AAA = 0.01, BBB = 0.02, Junk = 0.05)

# Function for conditional default
conditional_default <- function(rating) {
  local_default_rates <- c(
    AAA = unname(default_rates["AAA"]),
    BBB = unname(default_rates["BBB"]),
    Junk = unname(default_rates["Junk"])
  )
  return(local_default_rates[rating])
}
```

Lookup Tables

Lookup Tables

- **Lookup tables** map inputs to outputs.
- Example: Default probabilities for credit ratings.

Benefits:

1. Centralizes data for easy updates.
2. Avoids repetitive conditional statements.

Closures

Closures

- **Closures** are functions that retain their creation environment.
- Used to create dynamic and reusable functions.

Example: Probability Calculator Factory

```r
# Function factory
probability_calculator_factory <- function(event_probability) {
  function(conditional_probability) {
    joint_probability <- event_probability * conditional_probability
    return(joint_probability)
  }
}

# Create calculators
junk_calculator <- probability_calculator_factory(0.05)
bbb_calculator <- probability_calculator_factory(0.02)

# Calculate joint probabilities
junk_joint <- junk_calculator(0.1)
bbb_joint <- bbb_calculator(0.2)

cat("Joint probability for Junk bonds:", junk_joint, "\n")
```

```
Joint probability for Junk bonds: 0.005
```

Analyzing Closures

Analyzing Closures

1. Function Factory:
    - Takes `event_probability` as input.
    - Returns a function for calculating joint probabilities.
2. Reusable Calculators:
    - `junk_calculator` for Junk bonds.
    - `bbb_calculator` for BBB bonds.
3. Encapsulation:
    - Parameters are “locked in” during function creation.

Bayes' Rule: A Cornerstone of Probability

Bayes' Rule: A Cornerstone of Probability

- Bayes' Rule provides a systematic method for updating probabilities based on new evidence.
- Formula:

$$P(B|A) = \frac{P(A|B)P(B)}{P(A)}$$

- Explanation:
  - A: Observed evidence.
  - B: Hypothesis or prior belief.

Deriving Bayes’ Rule

Deriving Bayes' Rule

- From the Multiplication Rule:
    1. $P(B|A)P(A) = P(A \cap B)$
    2. $P(A|B)P(B) = P(A \cap B)$
    - Equating and dividing by $P(A)$:

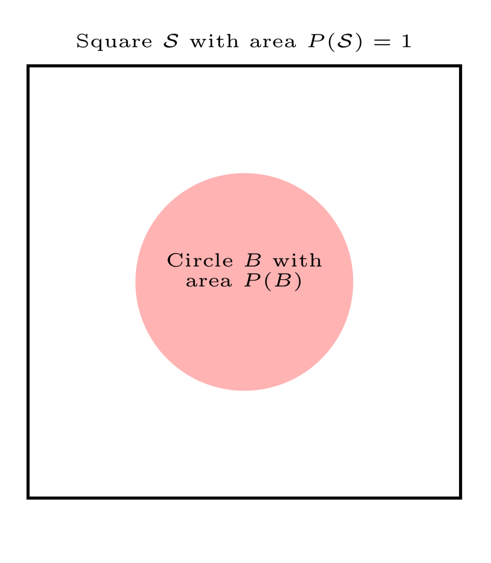
$$P(B|A) = \frac{P(A|B)P(B)}{P(A)}$$

- Significance: Ties prior beliefs to evidence using conditional probabilities.

Speck of Sand: An Intuitive Illustration

Speck of Sand: An Intuitive Illustration

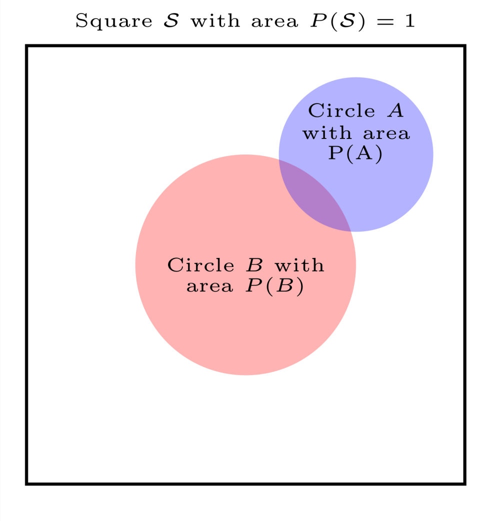
- A square of area 1 represents the sample space.
- Circle B represents the event with area $P(B)$.
- A speck of sand falls randomly in the square.


Square $\mathcal{S}$ with area $P(\mathcal{S}) = 1$
Circle B with area $P(B)$

Updating Beliefs

Updating Beliefs

- New Information:
    - The speck is known to land in another circle A inside the square.
- Question:
    - What is $P(B|A)$, the probability that the speck is in B, given it is inside A?


Square $\mathcal{S}$ with area $P(\mathcal{S}) = 1$
Circle A with area P(A)
Circle B with area $P(B)$

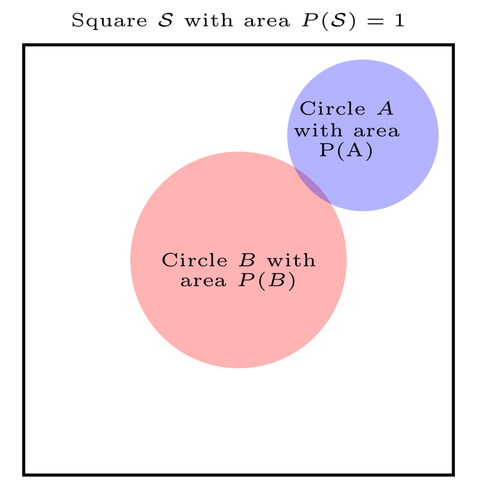
Overlap Between A and B

- The updated probability $P(B|A)$ depends on the overlap of B and A:

$$P(B|A) = \frac{\text{Area of } A \cap B}{\text{Area of } A} = \frac{P(A \cap B)}{P(A)}$$

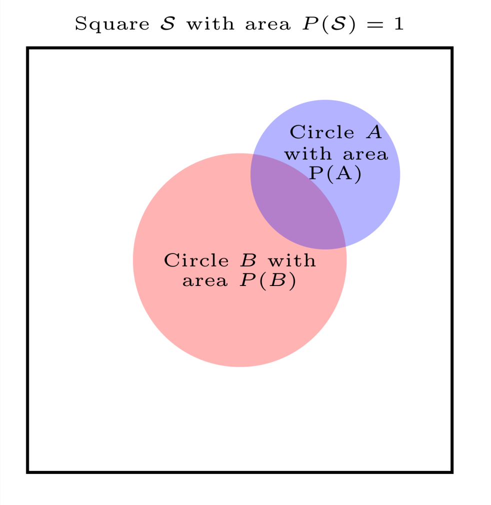
Small Overlap: Low Probability

Small Overlap: Low Probability



Large Overlap: High Probability

Large Overlap: High Probability



Bayesian Interpretation

Bayesian Interpretation

- Terms:
  - $P(B)$: Prior probability.
  - $P(A|B)$: Likelihood.
  - $P(A)$: Normalizing constant.
- Process:
  1. Start with prior $P(B)$.
  2. Add evidence $P(A|B)$.
  3. Compute updated belief $P(B|A)$.

Summary

- Conditional Probability: $P(A|B) = \frac{P(A \cap B)}{P(B)}$
    - Probability of A given that B has occurred
- Independence vs. Dependence
    - Independent: $P(A|B) = P(A)$
    - Dependent: Knowing B changes the probability of A
- Bayes' Rule: $P(B|A) = \frac{P(A|B) \cdot P(B)}{P(A)}$
    - Updating beliefs based on new evidence

Summary: Finance Concepts

- Credit Risk and Bond Ratings
    - Default probabilities vary by credit quality
    - Ratings: AAA (safest) to C/D (default)
- Structured Finance: Pooling and Tranching
    - Combine risky assets into pools
    - Create tranches with different risk profiles
- The Independence Trap
    - Assuming independence can drastically underestimate risk
    - Systemic events create correlated defaults
    - Lesson from 2007-2008: dependencies matter!

Summary: R Concepts

- Environments and Scoping
  - Global vs. local variables
  - Lexical scoping: R searches from inner to outer environments
- Lookup Tables
  - Named vectors for mapping inputs to outputs
  - Cleaner than nested if-else statements
- Closures (Function Factories)
  - Functions that "remember" their creation environment
  - Useful for creating specialized calculators

Motivation

- An investor in 2011 believed corporate profit margins would revert to mean
- Year after year, profits stayed elevated while the investor waited
- The mistake: **failing to update beliefs** as disconfirming evidence accumulated
- Winning isn't about having correct priors—it's about **efficiently updating** them

The Game Setup

Two types of coins:

- **Green coins**: 70% Heads, 30% Tails
- **Red coins**: 30% Heads, 70% Tails

Rules:

- Referee secretly draws a coin from a bucket
- You can “buy” each flip at some price
- Heads pays €2.00, Tails pays €0.00

The Key Insight

Worth of a flip depends on the coin:

- Green coin: $0.70 \times 2.00 + 0.30 \times 0.00 = 1.40$ (in €)
- Red coin: $0.30 \times 2.00 + 0.70 \times 0.00 = 0.60$ (in €)

Your task: Estimate which coin is being used to know what to pay

The Challenge

You can't see the coin's color directly. However:

1. Before each round, examine the bucket to estimate the proportion of green vs. red coins (your **prior**)
2. As flips occur, observe the results (your **data**)
3. **Update** your belief about which coin is being used

This is exactly what Bayes' Rule does!

Part 1: Expected Value (Question 1.1)

Suppose you believe there's a 90% chance the coin is green.

(a) Calculate the expected worth of a single flip.

(b) Write an R function `calculate_worth()`:

```r
# Starter code
calculate_worth <- function(prob_green) {
  # Green coin worth: €1.40
  # Red coin worth: €0.60
  # Your code here
}

# Test: should return €1.32 for prob_green = 0.9
calculate_worth(0.9)
```

Part 2: Bayesian Updating Mathematics

Let G = "coin is green", D_n = data from n flips with k heads.

$$P(G|D_n) = \frac{P(D_n|G) \cdot P(G)}{P(D_n)}$$

The likelihood follows a binomial distribution:

- $P(k \text{ heads in } n \text{ flips}|G) = \binom{n}{k}(0.7)^k(0.3)^{n-k}$
- $P(k \text{ heads in } n \text{ flips}|R) = \binom{n}{k}(0.3)^k(0.7)^{n-k}$

Part 2: Implementing Bayes' Rule (Question 2.1)

```r
update_belief <- function(prior_green, n_heads, n_flips) {
  p_heads_green <- 0.7
  p_heads_red <- 0.3

  # Calculate likelihood (hint: use dbinom())
  likelihood_green <- # Your code
  likelihood_red <- # Your code

  # Apply Bayes' Rule
  posterior_green <- # Your code

  return(posterior_green)
}

# Test: After 10 flips with 2 heads, prior 0.9
update_belief(0.9, n_heads = 2, n_flips = 10)
```

Part 2: Sequential Updating (Question 2.2)

Write `simulate_round()` that simulates 50 flips and updates beliefs:

```r
simulate_round <- function(true_color, prior_green, n_flips = 50) {
  p_heads <- ifelse(true_color == "green", 0.7, 0.3)

  results <- data.frame(
    flip_number = 1:n_flips,
    result = character(n_flips),
    cumulative_heads = integer(n_flips),
    posterior_green = numeric(n_flips),
    worth_estimate = numeric(n_flips)
  )
  # Your code: simulate flips and update beliefs
  return(results)
}
```

Part 3: Visualizing Overconfidence (Question 3.1)

True coin is **red**, but you start with different priors:

- Prior 0.500: “I have no idea”
- Prior 0.900: “Probably green”
- Prior 0.990: “Almost certainly green”
- Prior 0.999: “Virtually guaranteed green”

```r
set.seed(2011)  # The year of our hypothetical investor!
priors <- c(0.5, 0.9, 0.99, 0.999)
# Run simulations and plot worth estimates over 50 flips
```

Part 3: The Asymmetry (Question 3.2)

Compare two scenarios:

- When the coin is **red** but you thought it was green
- When the coin is **green** and you thought it was green

Question: Why is overconfidence asymmetrically costly?

(Hint: What do you gain vs. what do you lose?)

Part 4: Profit Calculation (Question 4.1)

You bid your worth estimate on each flip:

- Pay: your worth estimate
- Receive: €2.00 if Heads, €0.00 if Tails

```
calculate_profits <- function(simulation_results) {
  # For each flip:
  # - You pay the worth_estimate
  # - You receive 2.00 if Heads, 0.00 if Tails
  # - profit = payout - worth_estimate
  # Return data frame with cumulative_profit added
}
```

Part 5: Real-World Connection (Question 5.1)

In the coin-flipping game analogy:

- What corresponds to “profit margins are elevated”?
- What corresponds to “profit margins revert to the mean”?
- What corresponds to “assigning a 99.9% prior”?

Key insight: The investor’s mistake wasn’t having wrong initial beliefs, but failing to update efficiently.

Summary: What You'll Learn

1. **Bayesian updating** provides a principled way to revise beliefs based on evidence
2. **Overconfidence is asymmetrically costly**: being wrong with high confidence takes longer to correct
3. **Evidence required for strong beliefs should be proportional to confidence**
4. **Efficient updating** can compensate for initially incorrect beliefs

Reference

The coin-flipping game analogy is adapted from “Profit Margins, Bayes’ Theorem, and the Dangers of Overconfidence” from the blog *Philosophical Economics* (September 2017).