

# **An Introduction to Probability**

Random Variables

---

Martin Summer

24 January, 2025

# Random Variables

---

# The Role of Random Variables in Finance

Random variables are the main mathematical tool that allows us to quantify and analyze uncertainty in Finance for complex portfolios and other financial contexts like risk management. In all of these contexts, random variables provide the foundation for probabilistic reasoning.

# Importance of Random Variables

- **Core of probability theory:** Link real-world phenomena to mathematical models.
- **Key tools:**
  - **Expected value**
  - **Variance**
  - **Covariance**
- **Applications in Finance:**
  - **Portfolio Management:** Estimating expected return and risk.
  - **Risk Assessment:** Modeling extreme events like market crashes.
  - **Asset Pricing:** Evolving prices using models like the binomial tree.

# Learning Objectives

By the end of this lecture, you will:

1. Understand the definition and properties of random variables.
2. Learn how to compute and interpret:
  - Expected value
  - Variance
  - Covariance
3. Apply R programming to:
  - Simulate random variables.
  - Construct a binomial tree to model asset price dynamics.

# Random Variables and Distributions

A **random variable** is a numerical outcome of a random phenomenon. Formally, a random variable is a function that assigns a real number to each outcome in the sample space of a random experiment. More explicitly:

## Definition: Random Variable

A **random variable**  $X$  is a function  $X : \mathcal{S} \rightarrow \mathbb{R}$  from the sample space (the set of all possible outcomes of the random experiment) to the real numbers.

# Random Variables and Distributions

A random variable is thus a **function** defined on the sample space of a random experiment. This formal definition allows us to generalize and analyze a wide variety of real-world scenarios. For instance:


- In the coin-flipping example from Lecture 2, the number of heads in 10 flips was a random variable.
- The number of multiple birthdays in a group of  $n$  people is also an example of a random variable.

By explicitly recognizing these as random variables, we can now apply a systematic framework to quantify their behavior and analyze them.

# Random Variables and Distributions

- Convention in probability theory: Use capital letters such as  $X$  and  $Y$  etc. as symbols of a random variable.
- Random variables come in two varieties, depending on the properties of the sample space  $S$ . If the sample space is a finite or countably finite set, the sample space is discrete and we talk of a **discrete random variables**.
- Sometimes it is natural to consider continuous sample spaces. For example when we consider the return of an asset over a year or the price of a stock at a specific time.
- In this case we call a random variable **continuous**. With continuous sample spaces we will need tools from calculus. (lecture 5)



 Definition: Probability Mass Function of a Discrete Random Variable

Let  $X$  be a random variable and  $x_1, x_2, \dots$  the values which it assumes. The aggregate of all sample points on which  $X$  assumes a fixed value  $x_i$  is form the event  $X = x_i$ . It's probability is denoted by  $P(X = x_i)$ . The function:

$$P(X = x_i) = p(x_i), \quad i = 1, 2, \dots$$

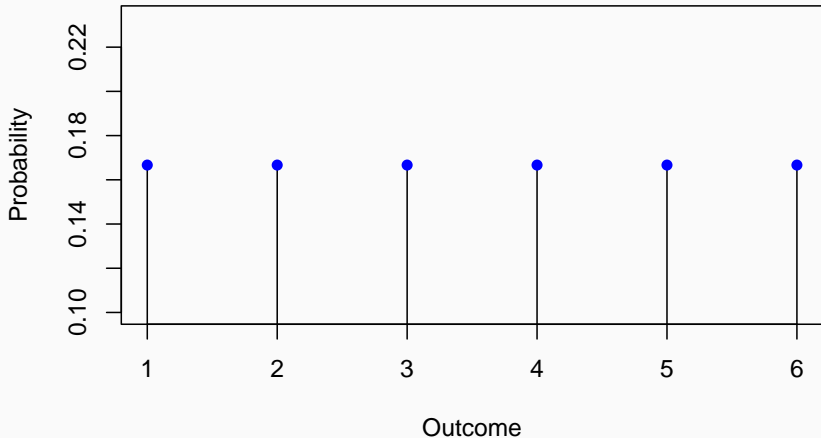
is called the **probability distribution** of the random variable  $X$ .

where  $p(x_i)$  satisfies:

- $0 \leq p(x_i) \leq 1$  for  $i = 1, 2, \dots$
- $\sum_i p(x_i) = 1$ .

# Visualization: Probability distribution six sided die

Probability Distribution



## Some very common confusions about random variables

1. **Random Variable vs. Outcome:** A random variable is not the same as an individual outcome. The confusion is partially created by the name. Maybe a better term would be a random mapping.
2. **Probability Distribution vs. Histogram:** A probability distribution represents theoretical probabilities. Don't mix this concept up with the concept of a histogram, known from statistics and data analysis, which shows frequencies of empirical data.
3. **Discrete vs. Continuous:** Discrete variables take specific values (e.g., dice outcomes), while continuous variables can take any value in a range. Dealing with continuous variables needs specific tools which we discuss in lecture 5.

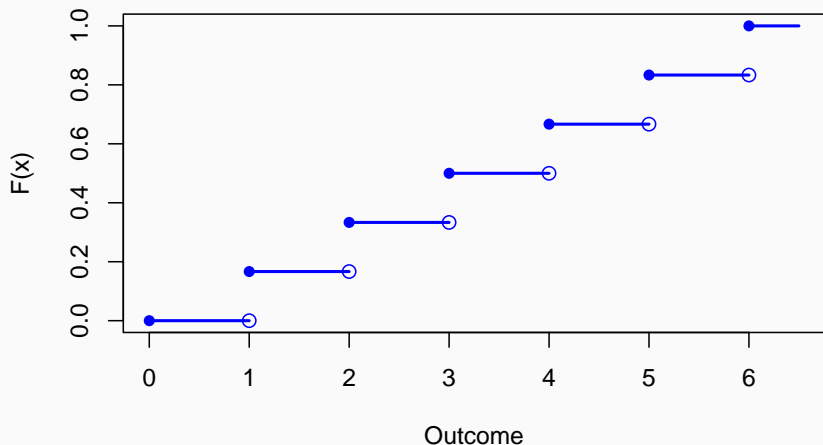
# Cumulative distribution function (CDF)

## Definition: Cumulative Distribution Function (CDF)

The **cumulative distribution function** - abbreviated CDF - shows the probability that a random variable  $X$  take a value less than or equal to a given value  $x_i$ . It is usually denoted as  $F(x_i) = P(X \leq x_i)$  where  $F$  is non-decreasing and  $0 \leq F(x_i) \leq 1$  for  $i = 1, 2, \dots$

## A visualization of the CDF for the six sided die

**Cumulative Distribution Function**



## A brief digression about building layered graphics in R

- The visualization of the CDF of the six sided die is an excellent opportunity for a brief digression into building graphs from layers in baseR.
- Let me walk you through the code of this visualization step by step.

## Start with an empty canvas setting the frame

```
x <- 1:6
cdf <- cumsum(rep(1/6, 6))

# Plot the empty plot frame
plot(0:6, c(0, cdf), type = "n",
     main = "Cumulative Distribution Function",
     xlab = "Outcome",
     ylab = "F(x)",
     xlim = c(0, 6.5), ylim = c(0, 1))
```

## Next draw bars and points

```
# Draw the first segment
```

```
segments(0, 0, 1, 0, col = "blue", lwd = 2)
```

```
# Draw a point with closed circle at (0,0)
```

```
points(0, 0, pch = 16, col = "blue")
```

```
# Draw a point with open circle at (1,)
```

```
points(1, 0, pch = 1, col = "blue", cex = 1.2)
```



## Draw the bars in a for loop

```
# Draw the stepwise CDF
for (i in 1:(length(x) - 1)) {
  # Draw the horizontal bar for each step
  segments(x[i], cdf[i], x[i + 1], cdf[i],
           col = "blue", lwd = 2)

  # Add the closed circle at the start of the segment
  points(x[i], cdf[i], pch = 16,
         col = "blue") # Filled circle

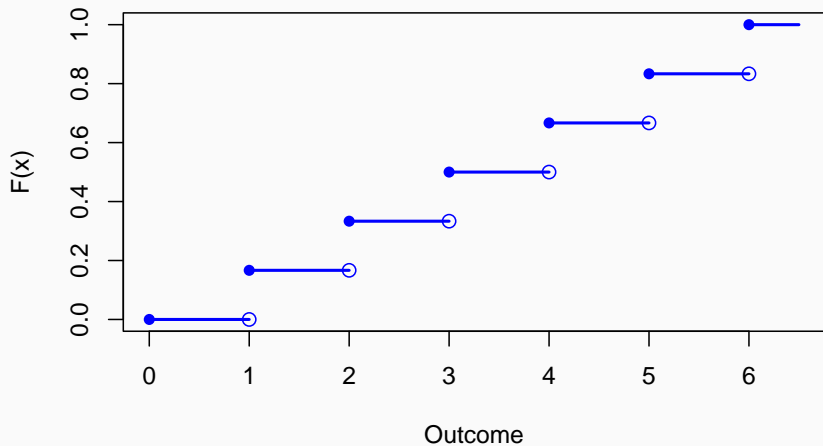
  # Add the open circle at the end of the segment
  points(x[i + 1], cdf[i], pch = 1,
         col = "blue", cex = 1.2) # Open circle
}
```

## Finishing bars and points

```
# Draw the last horizontal bar and closed circle at the end  
  
segments(6, cdf[6], 6.5, cdf[6], col = "blue", lwd = 2)  
points(6, cdf[6], pch = 16, col = "blue") # Closed circle
```

By executing these different layered steps in a sequence the graph is built in layer by layer to display the result I showed you before.

## Cumulative Distribution Function



## Key points about the CDF

- The CDF gives cumulative probabilities ( $P(X \leq x)$ ), not individual probabilities.
- The CDF is always non-decreasing and reaches 1 for the largest possible value of  $X$ .
- The CDF for discrete variables is a step-function with jumps at the values the random variable can take.

## Generalization: More than one random variable

### Joint Probability Distribution

Consider now two discrete random variables  $X$  and  $Y$  defined on the same sample space and let  $x_1, x_2, \dots$  and  $y_1, y_2, \dots$  the values which they assume. Let  $f(x_i)$  and  $g(y_j)$  be the corresponding distribution functions. The aggregate of points in which the two conditions  $X = x_i$  and  $Y = y_j$  are satisfied forms an event whose probability is denoted by

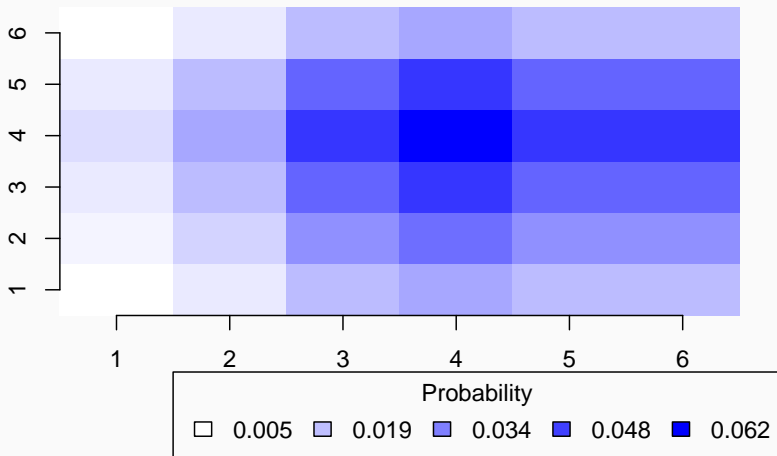
$$P(X = x_i, Y = y_j) = p(x_i, y_j), \quad i, j = 1, 2, \dots$$

is called the **joint probability distribution** of  $X$  and  $Y$ . where  $p(x_i, y_j)$  satisfies:

- $0 \leq p(x_i, y_j) \leq 1$  for  $i, j = 1, 2, \dots$ ,
- $\sum_{i,j} p(x_i, y_j) = 1$  for  $i, j = 1, 2, \dots$ .

# A visualization tool for two dimensions: Heat map for two (un-fair) dice

Heatmap of Joint PMF (Two Biased Dice)



# Conditional probability for discrete random variables

## Definition: Conditional Probability

The **conditional probability** of an event  $Y = y_j$ , given  $X = x_i$  with  $f(x_i) > 0$  is defined as

$$P(Y = y_j | X = x_i) = \frac{p(x_i, y_j)}{f(x_i)}$$

## Conditional distribution

In this way a number is associated with every value of  $X$  and so defines a function on  $X$ . This function is called

 Definition: Conditional Distribution

The **conditional distribution** of  $Y$  for given  $X$  and denoted by

$$P(Y = y_j | X)$$



## Important points about conditional distributions

- In general, the conditional probability distribution of  $Y$  given  $X = x_i$  (denoted  $p(y_j | x_i)$ ) differs from the marginal probability distribution  $g(y_j)$ .
- This means that except in special cases, knowing the value of  $X$  provides information about  $Y$ , which indicates stochastic dependence between  $X$  and  $Y$ .
- The strongest form of dependence occurs when  $Y$  is a deterministic function of  $X$ , i.e.,  $Y = h(X)$  for some function  $h$ . In this case,  $X$  completely determines  $Y$ , and their relationship is entirely predictable.

## Important points about conditional distributions

- If the joint probability distribution  $p(x_i, y_j)$  factorizes as:

$$p(x_i, y_j) = f(x_i)g(y_j) \quad \text{for all pairs } (x_i, y_j),$$

then  $X$  and  $Y$  are **independent**.

- This implies that the occurrence of one event (e.g.,  $X = x_i$ ) has no influence on the probability of the other (e.g.,  $Y = y_j$ ).
- The joint distribution in this case takes the form of a multiplication table, where the probabilities are products of the marginal probabilities. When  $X$  and  $Y$  are independent, their interaction is minimal, and no inference can be drawn about one variable from the other.

## Important points about the conditional distribution

- The joint distribution  $p(x_i, y_j)$  uniquely determines the marginal distributions  $f(x_i)$  and  $g(y_j)$ , as these can be computed by summing over the appropriate dimensions:

$$f(x_i) = \sum_j p(x_i, y_j), \quad g(y_j) = \sum_i p(x_i, y_j).$$

- However, the reverse is not true: the marginal distributions  $f(x_i)$  and  $g(y_j)$  alone do not determine the joint distribution  $p(x_i, y_j)$ . For example, different joint distributions can have the same marginals but encode different types of dependence or independence between  $X$  and  $Y$ .

## Important points about the conditional distribution

Two random variables  $X$  and  $Y$  can have the same marginal distribution  $f(x_i) = g(y_j)$  but may or may not be independent.

For instance:

- If  $X$  and  $Y$  are independent, their joint distribution will factorize as described earlier.
- If  $X$  and  $Y$  are dependent, their joint distribution will include nontrivial interactions between the variables.

## Example from Lecture 3: Conditional probability in structured finance

- this stage it might be a good idea to recall the example on structured finance from lecture 3.
- The defaultable bonds in this example were modeled as random variables.
- Bond-1 :  $\{D, N\} \mapsto \{0, 1\}$  and Bond-2 :  $\{D, N\} \mapsto \{0, 1\}$

## Example from lecture 3: The case of independent default risks

	Default (D)	No Default (N)	Total
Bond 1: Default (D)	0.01	0.09	0.1
Bond 1: No Default (N)	0.09	0.81	0.9
Total	0.10	0.90	1.0

Contingency Table of Joint and Marginal Probabilities: Independence

## Example from lecture 3: The case of dependent default risks

	Y: Default (d)	Y: No Default (n)	Total
X: Default (d)	0.06	0.04	0.1
X: No Default (n)	0.04	0.86	0.9
Total	0.10	0.90	1.0

Contingency Table of Joint and Marginal Probabilities: Dependence

Keep in mind:

- **Joint Distributions** describe how two random variables behave together.
- **Conditional Distributions** refine our understanding of one variable based on information about another.
- Joint PMFs naturally generalize to describe both **independent** and **dependent** random variables. For dependent variables, the joint PMF captures the interaction and dependencies between the variables, often requiring **conditional probabilities** to explain the relationships.
- These concepts are foundational for understanding dependencies in financial modeling, such as asset correlations or portfolio risk.



# **The Binomial Distribution: A Fundamental Model of Chance**

---

# The Binomial Distribution: A Fundamental Model of Chance

- The **binomial distribution** is one of the most important distributions for discrete random variables.
- It arises whenever we repeat a simple experiment, known as a Bernoulli trial, multiple times under the same conditions. Each trial has two possible outcomes, often labeled as “success” and “failure.”
- By counting the number of successes in a fixed number of trials, the binomial distribution provides a complete picture of the probabilities associated with all possible outcomes.

## Why is the binomial distribution so important?

- It is ubiquitous in applications. From predicting election outcomes to evaluating the reliability of systems and understanding financial risks, the binomial model underpins countless real-world phenomena.
- More importantly, it serves as a foundational tool for understanding the behavior of discrete random variables, providing a framework to compute probabilities, analyze expectations, and quantify variability.

## A building block for the binomial distribution: Bernoulli variables

Consider a single flip of a fair coin. Let's define a random variable  $X$ , where:

- $X = 1$  if the coin lands heads (success),
- $X = 0$  if the coin lands tails (failure).

This is an example of a **Bernoulli random variable**, the simplest discrete random variable.

# Expected value of a discrete random variable

## Definition: Expected value

Let  $X$  be a random variable assuming the values  $x_1, x_2, x_3, \dots$  with probabilities  $p(x_1), p(x_2), p(x_3), \dots$ . The **expected value** of  $X$  is defined by

$$\mathbb{E}[X] = \sum_{x_i} x_i \cdot p(x_i)$$

provided the series converges absolutely. In this case, we say that  $X$  has finite expectation. If  $\sum |x_i|p(x_i)$  diverges, we say that  $X$  has no finite expectation.

## Intuitive interpretation of expected value

- Let the experiment be repeated  $n$  times under identical conditions and denote by  $X_1, X_2, \dots, X_n$  the values of  $X$  that were actually observed, then for large  $n$  the average of these values should be close to  $\mathbb{E}[X]$ .
- While the terms *mean*, *average* and *expectation* are synonymous, expectation is usually used in relation to random variables, whereas mean and average is used in relation to empirical data.

## Example: Expected value for Bernoulli random variable

Let us compute the expected value of the Bernoulli random variable  $X$ :

$$\mathbb{E}[X] = 1 \cdot p + 0 \cdot (1 - p) = p.$$

Thus, the expected value of  $X$  is  $p$ . For a fair coin ( $p = 0.5$ ),  $\mathbb{E}[X] = 0.5$ . This means that in the long run, half of the flips are expected to result in heads.



## Definition: Expected value

Let  $X$  be a random variable assuming the values  $x_1, x_2, x_3, \dots$  with probabilities  $p(x_1), p(x_2), p(x_3), \dots$ . The **variance** of  $X$  is defined by

$$\text{Var}(X) = \mathbb{E}[(X - \mathbb{E}[X])^2].$$



## Example: Bernoulli random variable

For discrete random variables, the variance can be written as:

$$\text{Var}(X) = \sum_{x_i} (x_i - \mathbb{E}[X])^2 \cdot p(x_i).$$

For our Bernoulli random variable  $X$ , substituting  $\mathbb{E}[X] = p$ :

$$\text{Var}(X) = (1 - p)^2 \cdot p + (0 - p)^2 \cdot (1 - p) = p(1 - p).$$

For a fair coin ( $p = 0.5$ ):

$$\text{Var}(X) = 0.5 \cdot 0.5 = 0.25.$$

The **standard error** (SE) is the square root of the variance:

$$SE(X) = \sqrt{\text{Var}(X)}.$$

For our fair coin:

$$SE(X) = \sqrt{0.25} = 0.5.$$

## Why standard error?

- The standard error is often more convenient than the variance because it is expressed in the same units as the expected value. It thus gives as a clear sense how much the random variable is spread around the mean in the units of the mean.
- Suppose a stock's daily return,  $X$ , is modeled as a random variable with an expected return of 0.002 (0.2%) and a variance of  $\text{Var}(X) = 0.0004$  (0.04%).
- The variance is expressed in squared units of the return.
- The standard error is 0.02 (2%) is expressed in the same units as the return. This tells us that daily returns typically deviate by about 2% from the expected return of 0.2%.

# Binomial random variable

Suppose we flip the coin  $n$  times and count the total number of heads. This total is a **binomial random variable**  $X$ , with parameters: -  $n$ : number of trials, -  $p$ : probability of heads.

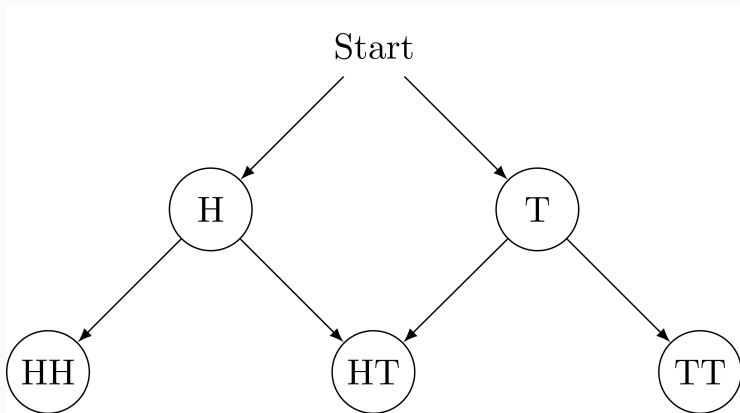
## Definition: Binomial Random Variable

A **binomial random variable** is a discrete random variable with the  $X$  with probability distribution

$$P(X = k) = \binom{n}{k} p^k (1 - p)^{n-k}, \quad k = 0, 1, \dots, n,$$

where  $\binom{n}{k}$  is the number of ways to choose  $k$  successes in  $n$  trials.

## Visualize as a binomial lattice



**Figure 1:** A binomial lattice

## Expected value and variance of a binomial random variable

- Let  $X$  be a binomial random variable with probability function

$$P(X = k) = \binom{n}{k} p^k (1 - p)^{n-k}, \quad k = 0, 1, 2, \dots, n.$$

- Its expected value is then:

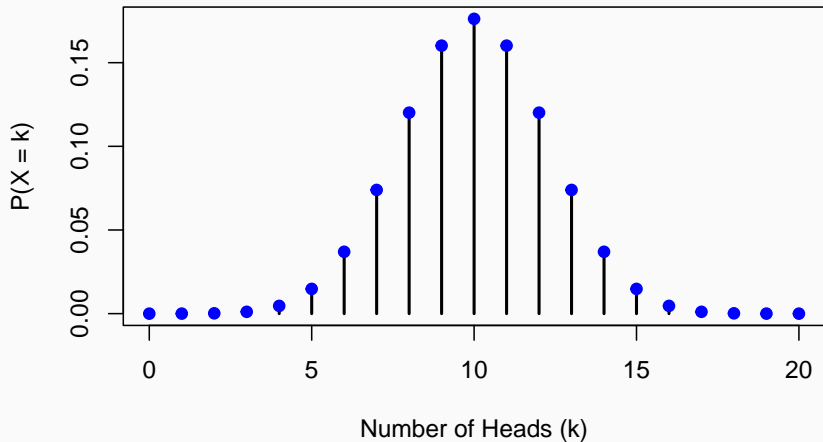
$$\mathbb{E}[X] = \sum_{k=0}^n k \cdot P(X = k) = p n.$$

- Its variance is

$$\text{Var}(X) = \sum_{i=1}^n \text{Var}(X_i) = n \cdot p(1 - p).$$

## Visualization for $n = 20$ and $p = 0.5$

PMF of Binomial(20, 0.5)



# The Power of the binomial distribution

The **binomial distribution** shows the power and versatility of the basic coin model we introduced right at the beginning of this course. It is the basic building block of this distribution and is very powerful in modeling simple experiments and deriving key properties:

- **Expected value** provides the long-run average.
- **Variance** and **standard error** measure variability.
- The PMF describes the probabilities of all possible outcomes.



## Definition: Covariance

The covariance of two random variables  $X$  and  $Y$  is defined by

$$\text{Cov}(X, Y) = \mathbb{E}(X - \mathbb{E}[X])(Y - \mathbb{E}[Y])$$

Alternatively we can write

$$\text{Cov}(X, Y) = \mathbb{E}[XY] - \mathbb{E}[X]\mathbb{E}[Y]$$

This definition is meaningful whenever  $X$  and  $Y$  have finite variance.

## Key points about Covariance

Note the following key points about covariance:

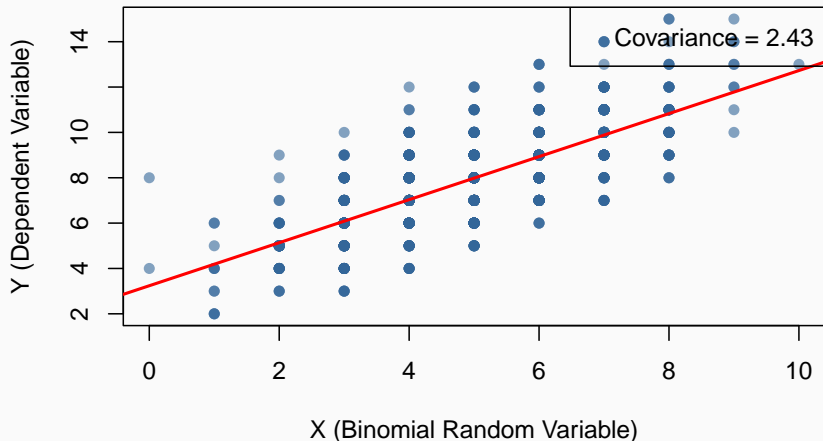
- If  $\text{Cov}(X, Y) > 0$ :  $X$  and  $Y$  tend to increase together (positive relationship).
- If  $\text{Cov}(X, Y) < 0$ :  $X$  and  $Y$  tend to move in opposite directions (negative relationship).
- If  $\text{Cov}(X, Y) = 0$ :  $X$  and  $Y$  are linearly uncorrelated, though they may still have a nonlinear relationship.

## Covariance measures only linear dependence

- Covariance captures only **linear relationships** because it measures the degree to which  $X$  and  $Y$  co-vary in a straight-line manner.
- If  $X$  and  $Y$  are related nonlinearly (e.g., quadratic, exponential), their covariance might still be zero even though a relationship exists.

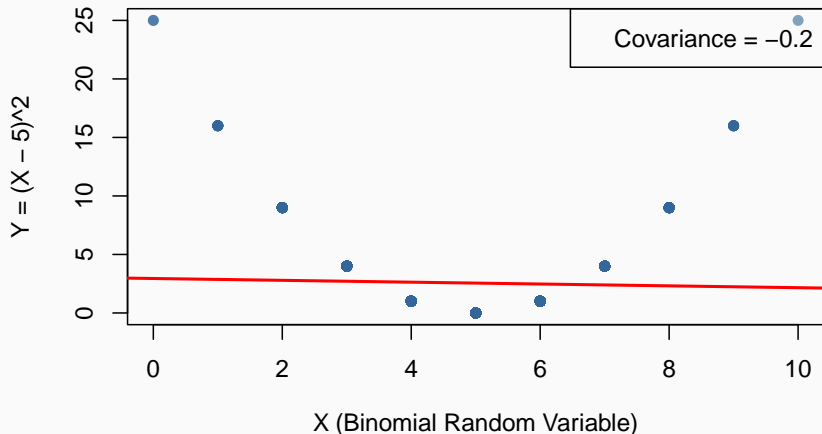
## Visualization: Two binomial random variables linear dependence

Scatterplot of X and Y with Regression Line



## Visualization: Two binomial random variables non-linear dependence

Scatterplot of  $Y = (X - 5)^2$



## Variance of a sum of two random variables

- The variance of the sum of two random variables,  $X$  and  $Y$ , is a natural place where covariance arises. Let's explore why this happens.
- For two random variables  $X$  and  $Y$ , the variance of their sum is:

$$\text{Var}(X + Y) = \text{Var}(X) + \text{Var}(Y) + 2 \cdot \text{Cov}(X, Y).$$

## Why does Covariance appear here?

When adding  $X$  and  $Y$ , the variance accounts not only for the individual variances of  $X$  and  $Y$ , but also for how they interact.

The term  $2 \cdot \text{Cov}(X, Y)$  reflects this interaction: - If  $X$  and  $Y$  are positively correlated, the variability of their sum increases. - If  $X$  and  $Y$  are negatively correlated, it decreases.

## A basic principle of insurance

- This property reflects a basic principle of insurance: Diversification reduces risk. In an insurance pool, risks (e.g., claims or losses) that are negatively correlated—or at least uncorrelated—reduce the overall variability of total claims.
- When risks are positively correlated (e.g., claims rise simultaneously due to shared external factors like natural disasters), the total risk increases, making diversification less effective. By managing correlation, insurers aim to stabilize payouts and maintain predictability.



## Independence as a special case

Consider independent variables as a special case: If  $X$  and  $Y$  are independent,  $\text{Cov}(X, Y) = 0$ , and the formula simplifies:

$$\text{Var}(X + Y) = \text{Var}(X) + \text{Var}(Y).$$

## Generalization to more than two outcomes

These insights can be extended to more than two random variables. For a sum of  $n$  random variables  $X_1, X_2, \dots, X_n$ , the variance becomes:

$$\text{Var} \left( \sum_{i=1}^n X_i \right) = \sum_{i=1}^n \text{Var}(X_i) + \sum_{i \neq j} \text{Cov}(X_i, X_j).$$

This shows how the covariances between pairs of random variables collectively influence the overall variance.

## **Case study: The CAPM formula and the role of covariance in comparing stocks**

---

## Capital asset pricing model

The Capital asset pricing model (CAPM) is a simple market mode, that relates individual stock returns to the market average. The CAPM formula is given by:

$$r_{jt} = \alpha_j + \beta_j m_t + \epsilon_{jt}$$

## Explanation of terms in the CAPM

- The **left-hand side**,  $r_{jt}$ , is the **excess return** of equity  $j$  at time  $t$ . This measures the return of the asset minus a measure of the risk-free rate.
- $m_t$  is the **market return**, which captures the aggregate return on the market at time  $t$ . A common choice when looking at US data is the S&P 500, which weights companies according to their market capitalization (the total value of their stock).
- $\epsilon_{jt}$  is an **error term**, a random variable that captures the part of the excess return not explained by the market return. It has the following properties:
  1.  $\mathbb{E}[\epsilon_{jt}] = 0$ , meaning it has no systematic bias.
  2.  $\mathbb{E}[m_t \epsilon_{jt}] = 0$ , meaning it is uncorrelated with the market return.

## Beta: $\beta$

The term  $\beta_j$  in the CAPM formula reflects the **sensitivity of the excess return of equity  $j$  to the market return  $m_t$** . It is calculated using the **covariance** between the returns of the asset and the market. Specifically:

$$\beta_j = \frac{\text{Cov}(r_j, m)}{\text{Var}(m)}$$

Where:

- $\text{Cov}(r_j, m)$  is the covariance between the excess return of equity  $j$  and the market return.
- $\text{Var}(m)$  is the variance of the market return.

## A simple tool to turn raw data into actionable insight

This simple model allows for a meaningful comparison between stocks based on their insurance properties, as well as whether they create value  $\alpha_j > 0$ , because the stock is providing value above what the model predicts or destroy value  $\alpha_j < 0$  because the stock provides less value than what the model predicts based on the performance of the market.

## Look at 20 biggest US stocks with tidyquant

I look specifically at:

- NVIDIA Corporation”, Apple Inc., Microsoft Corporation, Amazon.com, Inc.
- Alphabet Inc., Meta Platforms, Inc., Tesla, Inc.
- UnitedHealth Group Incorporated, Johnson & Johnson, Visa Inc.
- Exxon Mobil Corporation, JPMorgan Chase & Co
- Walmart Inc., Mastercard Incorporated, “Procter & Gamble Company
- The Home Depot, Inc., “Bank of America Corporation, Chevron Corporation
- Eli Lilly and Company, S&P 500 Index



## Loading the data

```
library(tidyquant)

tickers <- c("NVDA", "AAPL", "MSFT", "AMZN",
             "GOOGL", "META", "TSLA",
             "UNH", "JNJ", "V", "XOM",
             "JPM", "WMT", "MA", "PG", "HD",
             "BAC", "CVX", "LLY")

# Add the S&P 500 index ticker
tickers <- c(tickers, "^GSPC")

# Get historical data starting from 2015
price_data <- tq_get(
  tickers,
  get = "stock.prices",
  from = "2015-01-01"
)
```

## Do some data filtering first

Now let's do some definitions and data filtering first:

```
# Define the market ticker for the S&P 500
```

```
market_ticker <- "^GSPC"
```

```
# Filter the market and stock data from  
# the previously downloaded dataset
```

```
market_data <- price_data[price_data$symbol ==  
                           market_ticker, ]  
stocks_data <- price_data[price_data$symbol !=  
                           market_ticker, ]
```

## Do some data preparation

```
# Ensure the date column is properly formatted

price_data$date <- as.Date(price_data$date)

# Calculate daily returns for market and stocks

returns_data <- do.call(rbind,
  lapply(split(price_data, price_data$symbol),
    function(df) {
      # Convert to xts object for dailyReturn
      xts_data <- xts::xts(df$adjusted, order.by = df$date)
      df$daily_return <-
        quantmod::dailyReturn(xts_data, type = "log")
      return(df)
    })
  )
```

## Do some more data preparation

```
# Separate market and stock returns

market_returns <- returns_data[returns_data$symbol ==
                               market_ticker,
                               c("date", "daily_return")]
stock_returns <- returns_data[returns_data$symbol !=
                               market_ticker, ]

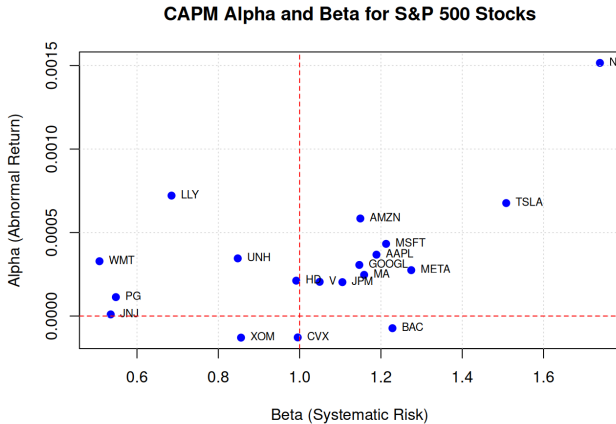
# Merge market returns with each stock's returns

merged_data <- merge(stock_returns,
                     market_returns, by = "date",
                     suffixes = c("", "_market"))
```

## Fit CAPM to each stock

```
# Fit CAPM for each stock
capm_results <- do.call(rbind,
  lapply(split(merged_data, merged_data$symbol),
    function(df) {
      model <- lm(daily_return ~ daily_return_market,
        data = df)
      data.frame(
        symbol = unique(df$symbol),
        alpha = coef(model)["(Intercept)"],
        beta = coef(model)["daily_return_market"]
      )
    })
  )))
```

# Visualize in $\alpha$ - $\beta$ -space



## Working with the binomial distribution in R

R provides built-in functions for working with the binomial distribution. These functions follow a **generic syntax** that is consistent across many other distributions, making them powerful tools for statistical analysis.

## d is for the probability density function

The probability distribution gives the probability of observing a specific number of successes in a binomial experiment: `dbinom(x, size, prob)` with

- `x`: Number of successes.
- `size`: Number of trials ( $n$ ).
- `prob`: Probability of success ( $p$ ).



## p is for the cumulative distribution function

The CDF gives the probability of observing up to a certain number of successes: `pbinom(x, size, prob)` with

- `x`: Upper limit of the number of successes.
- `size`: Number of trials ( $n$ ).
- `prob`: Probability of success ( $p$ ).

## q is for the quantile function

The quantile function computes the smallest number of successes for which the cumulative probability exceeds a specified value:

`qbinom(p, size, prob)` with

- `p`: Cumulative probability.
- `size`: Number of trials ( $n$ ).
- `prob`: Probability of success ( $p$ ).

## r is for random number generation

This function generates random samples from the binomial distribution: `rbinom(n, size, prob)` with

- `n`: Number of random values to generate.
- `size`: Number of trials ( $n$ ).
- `prob`: Probability of success ( $p$ ).



# Programing in R: The binomial lattice model

---

# Introduction to the Binomial Lattice Model

The **binomial lattice model** is a simple yet powerful framework for simulating asset price dynamics under uncertainty. It is widely used in financial modeling, particularly in pricing derivative securities like options.

- Captures probabilistic price movements in discrete time.
- At each time step, an asset can move up or down by a specified factor.
- A great application case for discrete random variables and more complex R programming.

# Objectives of the Lecture

In this session, we will:

- Apply the **binomial lattice model** to simulate the dynamics of the **S&P 500 index**.
- Reinforce R programming concepts while exploring this financial modeling tool.

## Learning Goals:

- Write more complex R programs, including functions for creating and exploring the lattice.
- Use control structures (for, while, repeat) and conditional statements (if, else).
- Leverage R's **list structure** for organizing and manipulating data.
- Apply modular programming principles to write reusable, maintainable code.

# The Binomial Lattice Model: Basics

## Key Features:

- The model assumes a basic time period (e.g., one day, one week).
- The asset price can either:
  - Move **up** by a factor  $u$ , or
  - Move **down** by a factor  $d$ .
- Probabilities:
  - $P(\text{up}) = p$ ,
  - $P(\text{down}) = 1 - p$ .

## Recursive Dynamics:

- If the initial price is  $S_0$ , the price at the next time step can be:
  - $uS_0$  or  $dS_0$ ,
  - Repeated over  $n$  periods.



## Connection to the Binomial Distribution

The price after  $n$  periods is:

$$S_n = S_0 \times u^k \times d^{n-k}$$

Where: -  $k$ : Number of upward movements. -  $n - k$ : Number of downward movements.

The distribution of  $k$  follows the **binomial distribution**:

$$P(k) = \binom{n}{k} p^k (1-p)^{n-k}, \quad k = 0, 1, 2, \dots, n$$

# Approximating Realistic Price Dynamics

## Insights:

- The binomial lattice model generates a discrete range of prices consistent with the binomial distribution.
- As the number of periods  $n$  increases, the model becomes more flexible and realistic.

## Advantages:

- Non-negative prices due to the multiplicative structure.
- The model approximates more complex dynamics as  $\Delta t$  (time step size) decreases.

To handle multiplicative price dynamics, we use logarithmic transformations:

- **Expected annual growth rate:**

$$\nu = \mathbb{E}[\ln(S_T/S_0)]$$

- **Variance of returns:**

$$\sigma^2 = \text{var}[\ln(S_T/S_0)]$$

## Parameter Selection for Small Time Steps

For small  $\Delta t$  (e.g., daily or weekly periods), the parameters of the binomial lattice are:

$$p = \frac{1}{2} + \frac{1}{2} \left( \frac{\nu}{\sigma} \right) \sqrt{\Delta t}$$

$$u = \exp(\sigma\sqrt{\Delta t}), \quad d = \exp(-\sigma\sqrt{\Delta t})$$

### Example:

- If  $\Delta t = 1$  week, the scaling factor is  $\sqrt{1/52}$ .
- If  $\Delta t = 1$  day, the scaling factor is  $\sqrt{1/252}$ , assuming 252 trading days in a year.

By the end of this lecture, you will have:

- Gained a deeper understanding of the **binomial lattice model**.
- Learned how to apply R programming techniques to simulate and explore price dynamics.
- Understood how the model connects to real-world financial scenarios.

# Simulating the dynamic of the SP500

---

## Calibrating the parameters from data

```
# Retrieve S&P 500 data (adjust ticker as needed)
sp500_data <- tq_get("SPY",
                     from = "2000-01-01", to = "2025-01-01",
                     get = "stock.prices")

# Extract adjusted prices and dates
dates <- sp500_data$date
adjusted_prices <- sp500_data$adjusted
```

## Extract annual prices

```
# Find the first trading day of each year
# Extract unique years
years <- unique(format(dates, "%Y"))
# Get the first trading day of the year
annual_indices <- sapply(years, function(year) {
  which(format(dates, "%Y") == year)[1]
})
# Extract prices and dates for the first
# trading day of each year
annual_prices <- adjusted_prices[annual_indices]
annual_dates <- dates[annual_indices]
```



## Compute annual log returns

```
# Compute annual log returns
# Compute  $\log(S_T / S_0)$ 
log_returns <- diff(log(annual_prices))

# Combine into a data frame
# Corresponding years for log returns
annual_log_returns <- data.frame(
  year = format(annual_dates[-1], "%Y"),
  log_return = log_returns
)
```

## Set parameters for weekly time steps

```
# Compute annual log returns
# Compute log(S_T / S_0)
log_returns <- diff(log(annual_prices))

# Combine into a data frame
# Corresponding years for log returns
annual_log_returns <- data.frame(
  year = format(annual_dates[-1], "%Y"),
  log_return = log_returns
)
```

## Set parameters

```
# Parameters for weekly time step (1/52 years)
# Weekly time step
delta_t <- 1 / 52
#Mean log return
nu <- mean(annual_log_returns$log_return, na.rm = TRUE)
# Variance
sigma_squared <- var(annual_log_returns$log_return, na.rm = TRUE)
# Standard deviation
sigma <- sqrt(sigma_squared)
# Calibrate binomial model parameters

p <- 0.5 + 0.5 * (nu / sigma) * sqrt(delta_t)
u <- exp(sigma * sqrt(delta_t))
d <- exp(-sigma * sqrt(delta_t))
```

## Building the lattice: For loops and lookup tables.

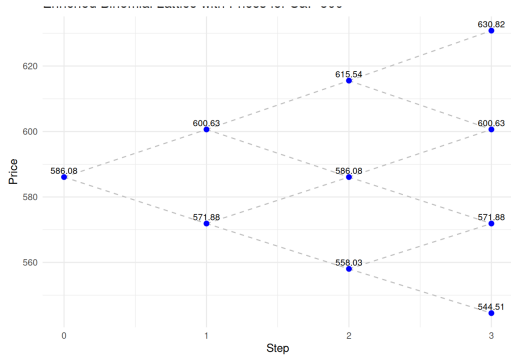
```
# Set the step number
n <- 3

# Set the starting price S0 (choose most recent price)
S0 <- adjusted_prices[length(adjusted_prices)]

# Initialize the lattice as a list
lattice <- vector("list", n + 1)

# Build the lattice
for (step in 0:n) {
  prices <- numeric(step + 1)
  for (i in 0:step) {
    prices[i + 1] <- S0 * u^i * d^(step - i)
  }
  lattice[[step + 1]] <- prices
}
```

# Visualize



## ggplot2 as a visualization tool

The `ggplot2` package is one of the most popular tools in R for creating visualizations. It uses a **layered grammar of graphics**, which means you build a plot step-by-step by adding layers. Each layer corresponds to a specific aspect of the visualization, such as:

1. **The Data:** Define the dataset you want to visualize.
2. **Aesthetic Mappings (`aes`):** Specify how variables in your data map to visual properties (e.g., x/y positions, colors, sizes).
3. **Geometries (`geom_`):** Add geometric objects like points, lines, or bars to represent the data visually.
4. **Themes and Labels:** Customize the look of the plot (theme) and annotate it with titles, axis labels, etc.

## Exploring the Lattice: Flow control

Finding the maximal price with a while-loop.

```
# Traverse lattice to find maximum price
max_price <- -Inf
step <- 1

while (step <= length(lattice)) {
  max_price <- max(max_price, max(lattice[[step]]))
  step <- step + 1
}

max_price
```

## Does the price exceed a threshold?: Conditional statements to annotate whether a price exceeds a certain threshold.

```
# Annotate prices exceeding 4100
threshold <- 4100

for (step in 0:n) {
  for (i in seq_along(lattice[[step + 1]])) {
    if (lattice[[step + 1]][i] > threshold) {
      print(paste("Price exceeds threshold at step", step,
    })
  }
}
```



# Modularizing your code

---

# Writing Modular Code: Reusable Functions for the Binomial Lattice

In programming, modularity refers to the practice of dividing your code into

- smaller
- reusable, and
- independent

pieces—typically functions—that can be composed to solve complex problems. Modularity enhances readability, maintainability, and reusability, making your code more professional and easier to debug or extend.

# Why should we write our programs in a modular way? Here are four

strong reasons:

1. **Clarity:** Breaking code into smaller, well-named functions makes the logic easier to follow.
2. **Reusability:** Functions can be reused across different projects or scenarios without rewriting the code.
3. **Debugging:** Errors can be isolated to specific functions, simplifying the debugging process.
4. **Testing:** Individual functions can be tested independently, ensuring correctness of each part.

## In our example

In our example the following modularization suggests itself:

1. `calibrate_parameters`: Calibrate the binomial lattice parameters from data.
2. `build_lattice`: Construct the binomial lattice using calibrated parameters.
3. `plot_lattice`: Visualize the lattice using `ggplot2`.
4. `explore_lattice`: Analyze properties of the lattice, such as maximum price or threshold crossings.

## Calibrate parameters

```
calibrate_parameters <- function(annual_prices, delta_t) {  
  log_returns <- diff(log(annual_prices))  
  
  # Calculate mean and variance  
  nu <- mean(log_returns, na.rm = TRUE)  
  sigma_squared <- var(log_returns, na.rm = TRUE)  
  sigma <- sqrt(sigma_squared)  
  
  # Calibrate parameters  
  p <- 0.5 + 0.5 * (nu / sigma) * sqrt(delta_t)  
  u <- exp(sigma * sqrt(delta_t))  
  d <- exp(-sigma * sqrt(delta_t))  
  
  list(nu = nu, sigma = sigma, p = p, u = u, d = d)  
}
```

## Building the lattice

```
build_lattice <- function(S0, n, u, d) {  
  lattice <- vector("list", n + 1)  
  
  for (step in 0:n) {  
    prices <- numeric(step + 1)  
    for (i in 0:step) {  
      prices[i + 1] <- S0 * u^i * d^(step - i)  
    }  
    lattice[[step + 1]] <- prices  
  }  
  
  lattice  
}
```

## Finding the maximum price

```
find_max_price <- function(lattice) {  
  max_price <- -Inf  
  for (step in seq_along(lattice)) {  
    max_price <- max(max_price, max(lattice[[step]]))  
  }  
  max_price  
}
```

## Check threshold

```
check_threshold <- function(lattice, threshold) {  
  for (step in seq_along(lattice)) {  
    for (price in lattice[[step]]) {  
      if (price > threshold) {  
        print(paste("Price exceeds threshold at step", step))  
      }  
    }  
  }  
}
```



# Assemble

```
# Calibrate parameters
parameters <- calibrate_parameters(annual_prices, delta_t =

# Build the lattice
lattice <- build_lattice(S0 = adjusted_prices[length(adjusted
                        n = 5,
                        u = parameters$u,
                        d = parameters$d)

# Plot the lattice
plot_lattice(lattice)

# Explore the lattice
max_price <- find_max_price(lattice)
cat("Maximum price in the lattice:", max_price, "\n")
```

As you progress in programming, consider **organizing your functions into an R package**. This makes them reusable and shareable.

## **Benefits of an R Package:**

- Structured organization of functions and documentation.
- Easier collaboration and consistent workflows.
- Reusability of tools for common tasks.

# Components of a Binomial Lattice Package

Your package for the **binomial lattice model** could include:

- Functions for **parameter calibration** (`calibrate_parameters`),
- Functions for **lattice construction** (`build_lattice`),
- Functions for **visualization** (`plot_lattice`), and
- Analytical tools (`find_max_price`, `check_threshold`).

## Why Create a Package?

- Streamline your workflow.
- Reinforce modular programming principles.
- Gain experience in R software development.

# Getting Started with R Packages

To dive deeper into creating R packages, refer to:

## Key Resource:

- **R Packages** by @WickhamBryan2023.

## Topics Covered:

- Setting up the package structure.
- Writing documentation with `roxygen2`.
- Adding tests to validate your functions.
- Publishing your package on GitHub or CRAN.

Focus first on **clean, modular functions**—these are the building blocks of any great R package.

# Using LLMs to Assist in Programming

Leverage **large language models (LLMs)** like ChatGPT for more complex tasks such as developing the binomial lattice model or creating an R package.

## Benefits:

- Code suggestions and explanations.
- Debugging help.
- Insights into best practices.

# How to Use LLMs Effectively

## 1. Break Down Your Problem

- Clearly define the task.
- Example prompts:
  - *“How do I implement a while loop in R?”*
  - *“What’s the best way to structure a parameter calibration function?”*

## 2. Use for Code Suggestions

- Request code snippets for specific tasks.
- Example: *“Write a function to visualize the lattice with ggplot2.”*

## 3. Seek Debugging Help

- Ask for explanations of error messages.
- Request fixes for common issues.

## 4. Learn Best Practices

- Modular and reusable code.
- Clear naming conventions.
- Example prompt: *“What are best practices for writing a reusable R function?”*

## 5. Turn Interactions into Learning Opportunities

- Example: *“Explain how `seq_along` is used in this loop.”*

## 6. Iterate with Feedback

- Start with a basic question and refine.
- Example: *“Can you add error handling to this function for invalid inputs?”*



# Be Mindful of LLM Limitations

While LLMs are powerful, they have limitations:

- They may generate syntactically correct but contextually incorrect code.
- Broader context might be misunderstood.
- Solutions might be outdated or suboptimal.

**Always review, test, and validate the code.**

## Example Use Case: Binomial Lattice Model

### Using LLMs in This Lecture:

- Write a function to construct the lattice.
- Explain how the binomial distribution relates to the model.
- Visualize the lattice effectively with `ggplot2`.

By combining your problem-solving skills with LLM capabilities, you can enhance your development process while learning.

- Creating an R package is a natural next step for reusable tools like the binomial lattice model.
- Leverage LLMs as a collaborative partner for:
  - Code generation,
  - Debugging, and
  - Learning programming best practices.

Focus on **building clean, modular functions**—the foundation for both packages and efficient programming.

# **Portfolio Simulation and Analysis with Discrete Random Variables**

---

In this project, you will:

- Simulate and analyze a portfolio composed of three assets under uncertain economic conditions.
- Model asset returns using discrete random variables.
- Compute portfolio metrics like expected return and variance.
- Understand portfolio diversification and its role in risk reduction.

This project reinforces: - Modular functions, - Control structures, - R's list structure.

# Project Objectives

By the end of this project, you will:

1. Apply random variables to model asset returns under uncertainty.
2. Simulate portfolio performance over multiple periods.
3. Compute and interpret key metrics: expected return, variance, and covariance.
4. Use R programming concepts to write efficient and reusable code.
5. Explore the trade-offs between risk and return in portfolio theory.

# Defining the Portfolio

1. **Economic States:** "High", "Medium", and "Low"
  - Probabilities: 30%, 50%, and 20%.
2. **Asset Returns** under each state:
  - **Asset 1:**  $c(0.08, 0.04, -0.02)$
  - **Asset 2:**  $c(0.12, 0.03, -0.05)$
  - **Asset 3:**  $c(0.05, 0.02, 0.01)$
3. **Portfolio Weights:**
  - 50% for Asset 1,
  - 30% for Asset 2,
  - 20% for Asset 3.

## Modular Functions:

1. Simulate a random economic state based on probabilities.
2. Retrieve asset returns for the simulated state using a list structure.
3. Compute portfolio return as the weighted sum of asset returns.



## Simulating Portfolio Performance (Cont'd)

### Steps:

- Simulate portfolio performance over **1,000 periods**.
- Store results in a list:
  - Simulated state.
  - Returns for each asset.
  - Portfolio return.

# Analyzing Portfolio Performance

1. Compute summary statistics for portfolio returns:
  - **Mean return.**
  - **Variance** of returns.
2. Compare simulated mean return with the **theoretical expected return**:

$$E(R_{\text{portfolio}}) = \sum_{i=1}^n w^i \cdot E(r^i)$$

3. (Optional) Compute the **theoretical variance**:

$$\text{Var}(R_{\text{portfolio}}) = \sum_{i=1}^n (w^i)^2 \cdot \text{Var}(r^i) + \sum_{i \neq j} w^i \cdot w^j \cdot \text{Cov}(r^i, r^j)$$

## 1. **Histogram:**

- Simulated portfolio returns.

## 2. **Frequency Table:**

- Display the frequency of each economic state.
- Verify implementation of probabilities.

1. Reflect on **diversification**:

- How do assets with different return profiles reduce portfolio variance?

2. Extensions:

- Add a fourth asset and observe its impact on portfolio metrics.
- Explore how changing portfolio weights affects expected return and variance.