Statistical Methods II: Week 1 Assignment (due 13 January, 2025)

1. Fifty measurements of four different features of a particular species of flower are recorded. It is planned that these measurements will be modelled as 50 realizations of a four-variate random vector. Describe the conventional form of the data matrix constructed for this purpose. According to the presumed model, do the elements of a particular row of this matrix have the same distribution? Explain.

2. Name two different plots for trivariate data (data on three variables) and explain their limitations.

3. What is the number of Chernoff's faces needed to represent a data matrix of order $n \times k$?

4. What is the number of piecewise linear curves in a parallel plot used to represent a data matrix of order $n \times k$? How many straight line segments does each curve have?

5. Write and execute R codes for evaluating the expressions

   (a) $\log_{10} 2 - \log_2 10$

   (b) $\tan \frac{\pi}{6}$

   (c) $\tan 60^\circ$

   (d) $\sin^{-1} \frac{\sqrt{3}}{2}$ in degrees

   (e) $\frac{1}{\sqrt{2\pi}} e^{-\frac{2^2}{2}}$

6. Without using R or RStudio, explain the outcome of the following activities.

   (a) You have saved a series of R codes in a script file, and left an RStudio session without saving the workspace image. In a new RStudio session, you have reopened the script file, the first line of which reads: `x <- 15`. After that, you typed `2*x` in the control window and hit the "Enter" key.

   (b) You have saved the following series of R codes in a script file:

   ```
   x <- 20
   y <- 15
   z <- x - y
   ```

   and left an RStudio session after saving the workspace image. In a new RStudio session, you have reopened the script file, and examined the content of the top right panel.

You may refer to the book *Introduction to Multivariate Data Analysis* written by Trevor F. Cox, available in the ISI library, for the multivariate plots discussed in the first week. (It may not be available in the shelf, as it has just been returned.)

1. Exploring the Data Matrix and Multivariate Distributions

To address the question, we'll break it down into its core components and explain all relevant concepts, including:

1. **Understanding the data matrix**
2. **Concepts of random vectors and multivariate distributions**
3. **Structure of data and assumptions**
4. **Answering the question based on these concepts**
5. **Related topics and further exploration**

1. Understanding the Data Matrix

A **data matrix** is a common structure used in statistics and machine learning to store multivariate data. When we record observations of multiple features (variables) for multiple entities (individuals, instances), the data can be organized as:

$$\mathbf{X} = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1p} \\ x_{21} & x_{22} & \cdots & x_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n1} & x_{n2} & \cdots & x_{np} \end{bmatrix}$$

Here:

- n: Number of observations (rows), which in this case is 50.
- p: Number of features (columns), which in this case is 4.
- x_{ij}: Value of the j^{th} feature for the i^{th} observation.

This is an $n \times p$ matrix, where each row corresponds to a single observation of all p features, and each column represents all observations of a single feature.

2. Concepts of Random Vectors and Multivariate Distributions

Random Vector

A **random vector** is a collection of random variables represented as a vector:

$$\mathbf{X} = (X_1, X_2, \ldots, X_p)^\top$$

Each component X_j is a random variable corresponding to one feature. The random vector models the joint behavior of all features.

Multivariate Distribution

The random vector $\mathbf{X}$ is described by its:

1. **Joint Probability Distribution**:
   - If $\mathbf{X}$ follows a multivariate normal distribution, we write:
     $$\mathbf{X} \sim \mathcal{N}_p(\boldsymbol{\mu}, \boldsymbol{\Sigma})$$
     where:
     - $\boldsymbol{\mu} = (\mu_1, \mu_2, \ldots, \mu_p)^\top$: Mean vector.
     - $\boldsymbol{\Sigma}$: Covariance matrix, capturing variances and covariances between variables.
2. **Marginal Distributions**:
   - The individual components $X_1, X_2, \ldots, X_p$ have marginal distributions derived from the joint distribution.
3. **Independence/Dependence**:
   - Features may be independent (no relationship) or dependent (correlated).

Realizations of the Random Vector

Each observation in the dataset corresponds to a **realization** of the random vector $\mathbf{X}$. For example:

$$\mathbf{x}_i = (x_{i1}, x_{i2}, \ldots, x_{ip})^\top$$

is the i^{th} realization of the random vector.

3. Structure of Data and Assumptions

Conventional Data Matrix Form

The data matrix stores realizations of a p-variate random vector as rows:

$$\mathbf{X} = \begin{bmatrix} \mathbf{x}_1^\top \\ \mathbf{x}_2^\top \\ \vdots \\ \mathbf{x}_n^\top \end{bmatrix} = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1p} \\ x_{21} & x_{22} & \cdots & x_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n1} & x_{n2} & \cdots & x_{np} \end{bmatrix}$$

where each row is a realization of the random vector, and each column represents all observations of a single feature.

Assumptions in the Model

1. **Independence Between Observations**:
   - The rows (realizations) are often assumed to be **independent** and identically distributed (i.i.d.).
2. **Same Distribution of Features Across Rows**:
   - Each row is a realization of the same random vector, implying that the joint distribution of features (columns) is the same for all rows.

4. Answering the Question

a. Conventional Form of Data Matrix

The data matrix is organized as a 50×4 matrix, where:

- Each row represents one realization of the four-variate random vector (one observation of all four features).
- Each column corresponds to all 50 measurements for a specific feature.

b. Do the Elements of a Particular Row Have the Same Distribution?

No, the elements of a particular row do **not** have the same distribution. Here's why:

- The elements in a row represent different features of the flower species, modeled as components of the random vector.
- These features may have distinct marginal distributions (e.g., feature 1 may be normally distributed with one mean and variance, while feature 2 may have a different distribution).

Thus, while the joint distribution of the random vector is the same across rows, the individual features in a row may have different distributions.

5. Related Topics and Further Exploration

a. Covariance Matrix

The covariance matrix $\mathbf{\Sigma}$ plays a crucial role in understanding the relationships between features:

$$\mathbf{\Sigma} = \begin{bmatrix} \text{Var}(X_1) & \text{Cov}(X_1, X_2) & \cdots & \text{Cov}(X_1, X_p) \\ \text{Cov}(X_2, X_1) & \text{Var}(X_2) & \cdots & \text{Cov}(X_2, X_p) \\ \vdots & \vdots & \ddots & \vdots \\ \text{Cov}(X_p, X_1) & \text{Cov}(X_p, X_2) & \cdots & \text{Var}(X_p) \end{bmatrix}$$

It describes:

- Variances of each feature (diagonal elements).
- Covariances between pairs of features (off-diagonal elements).

b. Principal Component Analysis (PCA)

PCA is a dimensionality reduction technique often applied to such data matrices to identify patterns and reduce dimensionality while retaining as much variance as possible.

c. Multivariate Statistical Tests

Tests like MANOVA (Multivariate Analysis of Variance) are used to analyze whether the mean vectors of multiple groups differ.

d. Questions for Practice

1. If the covariance matrix of the random vector is diagonal, what does that imply about the features?
2. Suppose you want to normalize the features (columns) of the data matrix. How would you do it, and why?
3. How does the assumption of independence affect the modeling of multivariate data?

2. Visualizing Trivariate Data

Breaking Down the Question

The goal is to explore:

1. **What trivariate data is**
2. **Two plots commonly used for visualizing trivariate data**
3. **Limitations of these plots**
4. **Related topics and further exploration**

1. What Is Trivariate Data?

Trivariate data involves observations on **three variables** simultaneously. It can be represented as a data matrix:

$$\mathbf{X} = \begin{bmatrix} x_{11} & x_{12} & x_{13} \\ x_{21} & x_{22} & x_{23} \\ \vdots & \vdots & \vdots \\ x_{n1} & x_{n2} & x_{n3} \end{bmatrix}$$

Here:

- n: Number of observations (rows).
- x_{ij}: The value of the j^{th} variable for the i^{th} observation.

Trivariate data often requires visualization methods that can represent relationships among all three variables simultaneously.

2. Two Common Plots for Trivariate Data

a. 3D Scatter Plot

A 3D scatter plot represents three variables by plotting points in three-dimensional space:

- **X-axis**: First variable.
- **Y-axis**: Second variable.
- **Z-axis**: Third variable.

Key Features

- Visualizes relationships among all three variables.
- Points in space represent individual observations.

Limitations

1. **Difficult to Interpret in 2D Medium**: When viewed on paper or a screen, the 3D perspective may cause points to overlap, making it challenging to interpret the relationships.
2. **Limited to Moderate Dataset Sizes**: For large datasets, the density of points can obscure patterns.
3. **No Clear Statistical Summaries**: Only visual patterns can be inferred; no direct statistical summaries are included in the plot.

b. Ternary Plot

A ternary plot (or triangle plot) is a two-dimensional plot used for trivariate data where the three variables represent proportions that sum to a constant (e.g., 1 or 100%).

Key Features

- Each vertex of the triangle corresponds to 100% of one variable, and points inside the triangle represent the relative proportions of all three variables.
- Commonly used in chemistry (e.g., compositions of mixtures) and geology.

Limitations

1. **Requires Proportional Data**: Ternary plots are meaningful only when the variables represent proportions summing to a constant. They cannot visualize general trivariate data.
2. **Difficult for Large Data**: With many data points, the plot becomes cluttered and difficult to read.
3. **No Direct Quantitative Analysis**: Interpretation relies on relative positions in the triangle; exact values may not be obvious.

3. Comparing the Two Plots

Aspect	**3D Scatter Plot**	**Ternary Plot**
Type of Data	$General trivariate data$	$Proportional data (sum to constant)$
Ease of Use	$Moderate$	$Requires specific data formatting$
Interpretation	$May be hard due to perspective issues$	$Relative values, not exact$
Applications	$General relationships (e.g., statistics)$	$Proportion-related studies (e.g., chemistr$

4. Related Topics and Further Exploration

a. Alternative Plots for Trivariate Data

1. **Bubble Plot**: A 2D scatter plot where the third variable is represented by the size (or color) of the points.
    - **Limitation**: Size/color interpretation can be subjective.
2. **Parallel Coordinate Plot**: Plots each observation as a line connecting three parallel axes for the three variables.
    - **Limitation**: Can become cluttered with large datasets.

b. Dimensionality Reduction Techniques

When visualizing higher-dimensional data:

- **Principal Component Analysis (PCA)**: Reduces three variables to two principal components for easier visualization.
- **t-SNE**: A nonlinear dimensionality reduction technique often used for high-dimensional datasets.

c. Questions for Practice

1. How would you interpret a cluster of points in a 3D scatter plot?
2. What preprocessing steps are required before constructing a ternary plot?
3. Can you think of real-world datasets that might require trivariate visualization?

3. Chernoff's Faces and Multivariate Data

Breaking Down the Question

We aim to answer: **What is the number of Chernoff's faces needed to represent a data matrix of order $n \times k$?**

To do so, we'll cover:

1. **Understanding Chernoff's Faces**
2. **Understanding the Data Matrix ($n \times k$)**
3. **Relating Chernoff's Faces to the Data Matrix**
4. **Answering the Question**
5. **Limitations of Chernoff's Faces**
6. **Related Concepts and Further Exploration**

1. Understanding Chernoff's Faces

a. What Are Chernoff's Faces?

Chernoff's faces are a graphical representation method that uses cartoon-like human faces to visualize multivariate data. Each variable (feature) in the dataset corresponds to a distinct facial characteristic, such as:

- Eye shape, size, or position
- Mouth curvature
- Face width or height
- Nose size, etc.

The idea is that humans can easily perceive differences in facial expressions, making it easier to identify patterns or outliers in multivariate data.

b. Key Characteristics

- Each face represents one **observation** (row in the data matrix).
- Features of the face represent values of the **variables** (columns in the data matrix).
- Chernoff's faces are mainly used for exploratory data analysis (EDA) and are not suited for quantitative analysis.

2. Understanding the Data Matrix ($n \times k$)

A data matrix of order $n \times k$ consists of:

- n: Number of observations (rows).
- k: Number of variables (columns).

Each observation corresponds to one instance in the dataset, and each variable represents a feature measured for all observations.

Example:

$$\mathbf{X} = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1k} \\ x_{21} & x_{22} & \cdots & x_{2k} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n1} & x_{n2} & \cdots & x_{nk} \end{bmatrix}$$

3. Relating Chernoff's Faces to the Data Matrix

a. How Many Faces Are Needed?

- Each **row** in the matrix corresponds to an **observation**.
- Each observation is represented by **one Chernoff's face**.

Thus, the **number of Chernoff's faces needed** is equal to the number of rows (n) in the data matrix.

b. How Many Facial Features Can Be Used?

- Each **column** (k) corresponds to a variable or feature.
- The number of facial characteristics used in the visualization should ideally match the number of variables k, but:
  - There's a practical limit to how many facial features can be meaningfully distinguished by human perception.
  - Typically, around 6–8 facial features are used, even if $k > 8$.

4. Answering the Question

For a data matrix of order $n \times k$:

- The number of Chernoff's faces needed is n, one for each observation (row).
- The number of facial features corresponds to k, but in practice, k is capped at a perceptually manageable number (6–8 features).

5. Limitations of Chernoff's Faces

1. **Limited Variables (Features):** Human perception cannot effectively interpret more than 6–8 facial features simultaneously.
2. **Subjectivity:** Interpretation of facial differences is qualitative and subjective, making it unsuitable for precise analysis.
3. **Overemphasis on Appearance:** Small variations in certain facial features may be overemphasized, leading to misinterpretation.
4. **Scalability:** Difficult to interpret for large datasets (n) due to the sheer number of faces.

6. Related Concepts and Further Exploration

a. Alternative Visualization Techniques

1. **Star Plots (Radar Charts):** Represent each observation as a star-shaped polygon where each axis corresponds to a variable.
    - **Limitation**: Challenging to interpret for high-dimensional data.
2. **Parallel Coordinate Plots:** Represent observations as lines across parallel axes, each representing a variable.
    - **Limitation**: Becomes cluttered with many observations.
3. **Heatmaps:** Use color intensity to represent values for a matrix.
    - **Limitation**: Doesn't convey patterns as intuitively as faces.

b. Practical Use Cases

Chernoff's faces are typically used for:

- Exploratory data analysis (EDA).
- Detecting clusters, patterns, and outliers in datasets.

c. Questions for Practice

1. What preprocessing steps (e.g., normalization) are required before using Chernoff's faces for visualization?
2. How would you modify Chernoff's faces to represent more than 8 variables?
3. Compare Chernoff's faces with star plots for multivariate data representation.

4. Parallel Coordinate Plots and Multivariate Data

Breaking Down the Question

We aim to address:

1. **What is the number of piecewise linear curves in a parallel plot used to represent a data matrix of order $n \times k$?**
2. **How many straight line segments does each curve have?**

To do so, we'll cover:

1. **Understanding Parallel Coordinate Plots**
2. **Structure of the Data Matrix ($n \times k$)**
3. **Relating Piecewise Linear Curves to the Data Matrix**
4. **Answering the Question**
5. **Limitations of Parallel Coordinate Plots**
6. **Related Topics and Further Exploration**

1. Understanding Parallel Coordinate Plots

a. What Are Parallel Coordinate Plots?

A **parallel coordinate plot** is a method for visualizing multivariate data. Each observation (row) in the dataset is represented by a piecewise linear curve, and each variable (column) corresponds to a parallel vertical axis.

Key Features:

- The number of axes equals the number of variables (k).
- Each curve represents one observation and connects the values of the variables across the axes.

b. How It Works

1. Normalize the variables so their values fall within a common range (e.g., [0, 1]).
2. Plot each observation as a line passing through the corresponding values on the parallel axes.

Example:

For a data matrix with $n = 5$ and $k = 3$:

- There are 3 vertical axes (one for each variable).
- Each observation is represented by a curve connecting the values on the 3 axes.

2. Structure of the Data Matrix ($n \times k$)

A data matrix has:

- n: Number of observations (rows).
- k: Number of variables (columns).

The structure is:

$$\mathbf{X} = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1k} \\ x_{21} & x_{22} & \cdots & x_{2k} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n1} & x_{n2} & \cdots & x_{nk} \end{bmatrix}$$

- Each row represents an observation.
- Each column represents a feature.

3. Relating Piecewise Linear Curves to the Data Matrix

a. Number of Piecewise Linear Curves

- Each **observation (row)** in the data matrix corresponds to **one piecewise linear curve** in the parallel plot.
- Therefore, the number of piecewise linear curves is equal to the number of rows (n) in the matrix.

b. Number of Line Segments per Curve

- Each curve connects the values of the k variables (columns) across the k parallel axes.
- To connect k axes, there are $k - 1$ straight line segments per curve.

Example:

- If $k = 3$ (3 variables), each curve will have 2 line segments.
- If $k = 5$ (5 variables), each curve will have 4 line segments.

4. Answering the Question

1. **Number of Piecewise Linear Curves:** The number of curves is **n**, the number of rows in the data matrix.
2. **Number of Straight Line Segments per Curve:** Each curve has $\mathbf{k-1}$ straight line segments, where k is the number of columns (variables).

5. Limitations of Parallel Coordinate Plots

1. **Overplotting:** When n (number of observations) is large, the plot becomes cluttered and difficult to interpret.
2. **Limited Axes Readability:** When k (number of variables) is large, the axes may overlap or the relationships between variables become hard to discern.
3. **Lack of Quantitative Information:** While patterns and clusters can be identified visually, precise relationships or correlations are not apparent.
4. **Scaling Issues:** Variables must be scaled to a common range, and poor scaling can obscure important patterns.

6. Related Topics and Further Exploration

a. Applications of Parallel Coordinate Plots

- Identifying clusters or outliers in multivariate datasets.
- Exploring relationships between variables.

b. Enhancements to Parallel Coordinate Plots

1. **Color Coding:** Use color to represent a categorical or numerical variable, enhancing interpretability.
2. **Interactive Tools:** Interactive versions allow users to highlight subsets of data and filter out irrelevant observations.

c. Alternative Visualizations for Multivariate Data

- **Chernoff's Faces:** Represent each observation as a face where variables correspond to facial features.
- **Star Plots:** Represent each observation as a star-shaped polygon with one axis per variable.

d. Questions for Practice

1. If you have a data matrix with $n = 100$ and $k = 5$, how many straight-line segments will be in the parallel plot? **Answer:** $n \times (k - 1) = 100 \times 4 = 400$.

2. How would you handle overplotting in a parallel coordinate plot with a large dataset?

3. What preprocessing steps are essential before creating a parallel coordinate plot?

6. RStudio Sessions, Scripts, and Workspace Images

Breaking Down the Question

We aim to address the outcomes of two scenarios involving RStudio sessions, scripts, and the workspace image. Here's how we'll approach this:

1. **Understanding Key Concepts:**
   - What is a script file?
   - What is the workspace image in R?
2. **Explaining the Outcomes of Each Scenario:**
   - **Scenario (a):** New session, script reopened, and code execution.
   - **Scenario (b):** New session with a saved workspace image and script reopening.
3. **Exploring Related Topics.**

1. Key Concepts

a. Script File in R

- A script file is a plain text file containing a series of R commands.
- When reopened, the script file does not execute automatically; you must manually run the code or specific lines of interest.

b. Workspace Image in R

- The workspace image contains the current state of all objects (e.g., variables, data frames) in memory.
- When saved, it is stored as a file named `.RData` in the working directory.
- On reopening RStudio, if the workspace image is loaded, all objects from the previous session will be restored.

c. RStudio Panels Overview

- **Top Right Panel (Environment Tab):** Displays all objects currently loaded into memory, such as variables, functions, and datasets.

2. Explaining the Outcomes

(a) New Session Without Workspace Image

1. **What Happened:**

   - You saved a script file with R codes and exited the session without saving the workspace image.
   - In a new session, you reopened the script file, which contains the first line: `x <- 15`.
   - You did not run the script but manually typed `2*x` in the console and hit "Enter."

2. **Outcome:**

   - Since the script file does not execute automatically, the assignment `x <- 15` was **not run**, and the variable `x` does not exist in the environment.
   - Typing `2*x` in the console will result in an **error:**

     ```
     Error: object 'x' not found
     ```

(b) New Session with Saved Workspace Image

1. **What Happened:**

   - You saved a script file containing the following code:

     ```
     x <- 20
     y <- 15
     z <- x - y
     ```

   - You exited the session **after saving the workspace image**.
   - In a new session, you reopened the script file but did not run it. You examined the Environment tab in the top right panel.

2. **Outcome:**

   - Since the workspace image was saved, the objects `x`, `y`, and `z` were stored in the `.RData` file.
   - Upon reopening RStudio, the workspace image is automatically loaded (unless explicitly disabled in settings).
   - The Environment tab will display:

     ```
     x   20
     y   15
     z    5
     ```

   - The script file is displayed in the editor but is not executed automatically.

3. Exploring Related Topics

a. Behavior of Workspace Images

- **Automatic Loading:** By default, RStudio loads the saved workspace image from the last session unless disabled.
- **Manual Loading:** If not automatically loaded, you can load the `.RData` file manually using `load("path/to/file.RData")`.

b. Advantages of Not Saving the Workspace

- Encourages clean and reproducible coding practices.
- Forces users to rerun scripts, ensuring the code captures all necessary steps.

c. Common Errors in Such Scenarios

1. **Object Not Found Error:**
   - Occurs when a variable referenced in the console was not initialized.
2. **Overwriting Variables:**
   - Saving a workspace image can lead to confusion if old objects conflict with new ones.

d. Questions for Practice

1. How would the outcome in Scenario (a) change if you had run the script before typing `2*x`?
2. How can you prevent RStudio from automatically loading a saved workspace image in a new session?
3. Explain the advantages of writing reproducible scripts over relying on workspace images.

Statistical Methods II: Week 2 Assignment (due 20 January, 2025)

1. Show an example of a computation in R, where the accuracy of the computation depends on the order of computation (be prepared to give your own example, in case another student is asked to answer the same question before you are called).

2. Write an R code to simulate 10 rolls of a fair die.

3. Write an R code to simulate 10 tosses of a biased coin.

4. Write an R code to draw a random sample of 2 roll numbers from the list BS2445, BS2447, BS2448, BS2450, BS2452, for answering the next question in this list. How should one handle the 'Repeat' argument?

5. Continuing with the above problem, the student with roll number BS2450 was called in the previous week. There is a plan to assign him half the probability of being selected, compared to probability allocated to the four other students. How should one adjust the R code?

6. Write an R code to draw a sample of size 1 from the Poisson distribution with mean 3.

7. Write an R code to draw a sample of size 1 from the Exponential distribution with mean 3.

8. Generate a sample of size 100 from the normal distribution with mean 2 and standard deviation 1. Is the mean of this sample close to two? What happens when you repeat the exercise? What happens when the sample size is greater or smaller?

9. A tosses a fair coin until he gets a Head. B tosses a coin as many times as A took to get the first Head. Write an R code to get an idea of the expected number of Heads obtained by B in her tosses. Explain why this number should be less than 1.

Numerical Accuracy in R: Order of Computation

Example Problem

Compute the value of $f(x) = \frac{1-\cos(x)}{x^2}$ for $x = 10^{-8}$.

Direct Computation (Prone to Error)

Using the formula $\frac{1-\cos(x)}{x^2}$, substituting $x = 10^{-8}$ directly in R can result in a loss of accuracy due to floating-point precision limitations.

Accurate Computation

Using the trigonometric identity:

$$1 - \cos(x) = 2\sin^2\left(\frac{x}{2}\right),$$

we rewrite the formula as:

$$f(x) = \frac{2\sin^2\left(\frac{x}{2}\right)}{x^2}.$$

This avoids the subtraction of nearly equal numbers, yielding a more accurate result.

R Code

```
# Direct computation
x <- 1e-8
f_direct <- (1 - cos(x)) / x^2

# Accurate computation using trigonometric identity
f_accurate <- (2 * sin(x / 2)^2) / x^2

# Display results
cat("Direct computation:", f_direct, "\n")
cat("Accurate computation:", f_accurate, "\n")
```

Related Concepts

- **Floating-Point Precision:** Computers represent numbers with finite precision, leading to errors in certain operations.
- **Catastrophic Cancellation:** Occurs when subtracting two nearly equal numbers.
- **Numerical Stability:** Algorithms that minimize errors due to floating-point arithmetic are considered numerically stable.

Related Questions

1. Why does catastrophic cancellation occur in floating-point arithmetic?

   - **Answer:** It occurs because floating-point numbers have limited precision, and subtraction of nearly equal numbers can amplify rounding errors.

2. How can numerical stability be improved in computations?

   - **Answer:** Use alternative mathematical formulations (e.g., identities, approximations) that avoid operations prone to error.

3. Write an R program to demonstrate loss of precision in adding and subtracting very large and small numbers.

   - **Answer:**

```
# Loss of precision example
a <- 1e10
b <- 1e-10
result <- (a + b) - a
cat("Result:", result, "\n")
```

Simulation of 10 Rolls of a Fair Die in R

Problem Statement

Simulate 10 rolls of a fair six-sided die using R.

R Code

```
# Simulate 10 rolls of a fair die
set.seed(123) # Set seed for reproducibility
rolls <- sample(1:6, size = 10, replace = TRUE)

# Display the results
cat("Results of 10 rolls:", rolls, "\n")
```

Explanation

- **Simulation Process:** The die has six faces numbered $1, 2, 3, 4, 5, 6$. We perform 10 independent rolls with replacement.
- **Key Functions:**
  - `sample(1:6, size = 10, replace = TRUE)`: Generates 10 random numbers between 1 and 6 with replacement.
  - `set.seed(123)`: Ensures reproducibility of the random sequence.
  - `cat()`: Prints the result of the simulation.

Related Concepts

- **Uniform Distribution:** A fair die corresponds to a uniform discrete distribution where each outcome (1–6) has an equal probability of $\frac{1}{6}$.
- **Reproducibility in Simulations:** Using a random seed ensures that results can be reproduced in subsequent runs.
- **Random Sampling with Replacement:** Replacement allows the same number to appear multiple times in the simulation, mimicking the behavior of a real die.

Related Questions

1. What is the expected frequency of each number in 10 rolls of a fair die?
   - **Answer:** Each number 1 through 6 is expected to appear approximately $\frac{10}{6} \approx 1.67$ times.
2. How would the code change if the die is biased?

- **Answer:** Modify the `prob` parameter in the `sample()` function to reflect the biased probabilities, e.g.,

```
sample(1:6, size = 10, replace = TRUE, prob = c(0.1, 0.2, 0.3, 0.1, 0.2
```

3. How can you simulate rolls for multiple dice at once?

- **Answer:** Use a matrix to store results, e.g.,

```
rolls <- replicate(3, sample(1:6, size = 10, replace = TRUE))
```

Simulation of 10 Tosses of a Biased Coin in R

Problem Statement

Simulate 10 tosses of a biased coin in R, where the probability of heads (H) is 0.7, and the probability of tails (T) is 0.3.

R Code

```
# Simulate 10 tosses of a biased coin
set.seed(123) # Set seed for reproducibility
coin_tosses <- sample(c("H", "T"), size = 10, replace = TRUE, prob = c(0.7, 0.3))

# Display the results
cat("Results of 10 tosses:", coin_tosses, "\n")
```

Explanation

- **Simulation Process:** The coin has two outcomes: Heads (H) and Tails (T).
- **Key Functions:**
  - `sample(c("H", "T"), size = 10, replace = TRUE, prob = c(0.7, 0.3))`: Simulates 10 tosses of the coin, accounting for the bias.
  - `set.seed(123)`: Ensures reproducibility.
  - `cat()`: Prints the result of the simulation.

Related Concepts

- **Bernoulli Distribution:** A single toss of the biased coin follows a Bernoulli distribution with $P(H) = 0.7$ and $P(T) = 0.3$.
- **Reproducibility:** The random seed ensures that the same sequence of outcomes is generated in subsequent runs.
- **Random Sampling with Replacement:** Ensures that each toss is independent of the others.

Related Questions

1. How would you simulate tosses for multiple biased coins at once?
   - **Answer:** Use a matrix to store the results of multiple coins, e.g.,

     ```
     tosses <- replicate(3, sample(c("H", "T"), size = 10, replace = TRUE, pr
     ```

2. How can you calculate the proportion of heads in the simulation?

- **Answer:**

```
mean(coin_tosses == "H")
```

3. How would you simulate a fair coin?
   - **Answer:** Set $P(H) = 0.5$ and $P(T) = 0.5$:

```
sample(c("H", "T"), size = 10, replace = TRUE, prob = c(0.5, 0.5))
```

Drawing a Random Sample of Roll Numbers in R

Problem Statement

Write an R code to draw a random sample of 2 roll numbers from the list: `BS2445, BS2447, BS2448, BS2450, BS2452`. Discuss how to handle the `replace` argument in the sampling process.

R Code

```
# List of roll numbers
roll_numbers <- c("BS2445", "BS2447", "BS2448", "BS2450", "BS2452")

# Draw a random sample of 2 roll numbers without replacement
set.seed(123) # Set seed for reproducibility
sample_rolls <- sample(roll_numbers, size = 2, replace = FALSE)

# Display the results
cat("Randomly selected roll numbers:", sample_rolls, "\n")
```

Explanation

- **Simulation Process:** The list contains five roll numbers, and a random sample of two is drawn without replacement.
- **Key Parameters:**
  - `replace = FALSE`: Ensures no roll number is repeated in the selection (default setting for `sample()`).
  - `replace = TRUE`: Allows the same roll number to appear more than once in the sample.
- **Random Seed:** `set.seed(123)` ensures reproducibility.

Related Concepts

- **Sampling Without Replacement:** Ensures unique roll numbers in the selection.
- **Sampling With Replacement:** Allows repetition of roll numbers in the sample.
- **Reproducibility:** Using a seed ensures consistency in the selection across different runs of the code.

Related Questions

1. What happens if `replace = TRUE`?

   - **Answer:** Roll numbers can repeat in the sample, e.g., `"BS2445", "BS2445"`.

2. How do you ensure all roll numbers have equal probability of selection?

   - **Answer:** By default, `sample()` assigns equal probabilities. For unequal probabilities, use the `prob` argument:

     ```
     sample(roll_numbers, size = 2, replace = FALSE, prob = c(0.1, 0.2, 0.3,
     ```

3. How can you sample more roll numbers than the size of the list?

   - **Answer:** Set `replace = TRUE` to allow repetitions, e.g.,

     ```
     sample(roll_numbers, size = 10, replace = TRUE)
     ```

Adjusting Probabilities for Biased Sampling in R

Problem Statement

Continuing with the above problem, the student with roll number `BS2450` was called in the previous week. There is a plan to assign him half the probability of being selected, compared to the probability allocated to the four other students. Adjust the R code to reflect this bias.

R Code

```
# List of roll numbers
roll_numbers <- c("BS2445", "BS2447", "BS2448", "BS2450", "BS2452")

# Define adjusted probabilities
# Assign equal probabilities to all except BS2450, which gets half the probability
probabilities <- c(1, 1, 1, 0.5, 1)
probabilities <- probabilities / sum(probabilities) # Normalize probabilities

# Draw a random sample of 2 roll numbers without replacement
set.seed(123) # Set seed for reproducibility
sample_rolls <- sample(roll_numbers, size = 2, replace = FALSE, prob = probabilities)

# Display the results
cat("Randomly selected roll numbers:", sample_rolls, "\n")
```

Explanation

- **Adjusting Probabilities:**
  - Assign a weight of 1 to each of the four other students.
  - Assign a weight of 0.5 to `BS2450` for its reduced selection probability.
  - Normalize weights by dividing each weight by the total sum of weights.
- **Updated Parameters:**
  - `prob = probabilities`: Adjusted probabilities account for the bias.
- **Random Seed:** `set.seed(123)` ensures reproducibility.

Related Concepts

- **Biased Sampling:** Adjusting the probabilities allows for unequal chances of selection.
- **Probability Normalization:** Ensures the sum of probabilities equals 1.
- **Flexible Weighting:** Custom weights can be implemented using the `prob` argument.

Related Questions

1. How would the probabilities change if one student was completely excluded?

   - **Answer:** Assign a probability of 0 to the excluded student and normalize the rest:

     ```
     probabilities <- c(1, 1, 1, 0, 1)
     probabilities <- probabilities / sum(probabilities)
     ```

2. How can one assign probabilities based on student performance?

   - **Answer:** Use performance-based weights, e.g., test scores:

     ```
     scores <- c(85, 90, 95, 70, 80) # Example scores
     probabilities <- scores / sum(scores)
     ```

3. What happens if `replace = TRUE`?

   - **Answer:** Students can be selected more than once in the sample, and probabilities are applied independently for each selection.

Drawing a Sample from a Poisson Distribution in R

Problem Statement

Write an R code to draw a sample of size 1 from the Poisson distribution with a mean of 3.

R Code

```
# Draw a sample of size 1 from the Poisson distribution with mean = 3
set.seed(123) # Set seed for reproducibility
poisson_sample <- rpois(n = 1, lambda = 3)

# Display the result
cat("Random sample from the Poisson distribution:", poisson_sample, "\n")
```

Explanation

- **Poisson Distribution:**
  - Models the number of events occurring in a fixed interval with mean rate λ.
  - Here, $\lambda = 3$, meaning the average number of events is 3.
- **R Function:**
  - `rpois(n, lambda)` generates n random samples from a Poisson distribution with mean λ.
  - `n = 1`: Specifies a sample size of 1.
  - `lambda = 3`: Specifies the mean.
- **Random Seed:** `set.seed(123)` ensures reproducibility.

Related Concepts

- **Poisson Distribution Properties:** The variance of a Poisson random variable is equal to its mean ($\text{Var}(X) = \lambda$).
- **Random Number Generation:** R provides a family of functions for random generation from various distributions, e.g., `rpois()`, `rnorm()`, `runif()`.
- **Reproducibility:** Setting a random seed ensures consistent random number generation.

Related Questions

1. How would you generate 10 samples instead of 1?

   - **Answer:** Change the `n` parameter to 10:

     ```
     samples <- rpois(n = 10, lambda = 3)
     ```

2. How can you calculate the probability of a specific outcome in a Poisson distribution?

   - **Answer:** Use the `dpois()` function, e.g., to find $P(X = 2)$ when $\lambda = 3$:

     ```
     probability <- dpois(x = 2, lambda = 3)
     ```

3. How can you plot the Poisson probability mass function?

   - **Answer:** Use the following code:

     ```
     x <- 0:10
     probabilities <- dpois(x, lambda = 3)
     barplot(probabilities, names.arg = x, main = "Poisson PMF ( = 3)",
             xlab = "x", ylab = "P(X = x)", col = "blue")
     ```

Drawing a Sample from an Exponential Distribution in R

Problem Statement

Write an R code to draw a sample of size 1 from the Exponential distribution with a mean of 3.

R Code

```
# Draw a sample of size 1 from the Exponential distribution with mean = 3
set.seed(123) # Set seed for reproducibility
rate <- 1 / 3 # Rate parameter (lambda) is the reciprocal of the mean
exp_sample <- rexp(n = 1, rate = rate)

# Display the result
cat("Random sample from the Exponential distribution:", exp_sample, "\n")
```

Explanation

- **Exponential Distribution:**
  - Models the time between events in a Poisson process.
  - The mean is $\mu = \frac{1}{\lambda}$, where λ is the rate parameter.
  - Here, the mean is 3, so $\lambda = \frac{1}{3}$.
- **R Function:**
  - `rexp(n, rate)` generates n random samples from an Exponential distribution with rate λ.
  - `n = 1`: Specifies a sample size of 1.
  - `rate = 1 / 3`: Specifies the rate parameter.
- **Random Seed:** `set.seed(123)` ensures reproducibility.

Related Concepts

- **Rate Parameter:** The rate parameter λ is inversely proportional to the mean ($\lambda = \frac{1}{\text{Mean}}$).
- **Random Number Generation:** R provides functions to generate random samples from distributions, e.g., `rexp()`, `rpois()`, `rnorm()`.
- **Reproducibility:** Setting a random seed ensures consistent random number generation.

Related Questions

1. How would you generate 10 samples instead of 1?

   - **Answer:** Change the `n` parameter to 10:

     ```
     samples <- rexp(n = 10, rate = 1 / 3)
     ```

2. How can you calculate the probability density of a specific value in the Exponential distribution?

   - **Answer:** Use the `dexp()` function, e.g., to find $f(x = 2)$ when $\lambda = \frac{1}{3}$:

     ```
     probability <- dexp(x = 2, rate = 1 / 3)
     ```

3. How can you plot the probability density function of the Exponential distribution?

   - **Answer:** Use the following code:

     ```
     x <- seq(0, 15, by = 0.1)
     density <- dexp(x, rate = 1 / 3)
     plot(x, density, type = "l", main = "Exponential PDF (Mean = 3)",
          xlab = "x", ylab = "f(x)", col = "blue", lwd = 2)
     ```

Generating a Sample from the Normal Distribution in R

Problem Statement

Generate a sample of size 100 from the normal distribution with mean 2 and standard deviation 1. Is the mean of this sample close to two? What happens when you repeat the exercise? What happens when the sample size is greater or smaller?

R Code

```
# Generate a sample of size 100 from the normal distribution with mean = 2, sd = 1
set.seed(123) # Set seed for reproducibility
n <- 100
mean_value <- 2
sd_value <- 1

# Generate the sample
normal_sample <- rnorm(n, mean = mean_value, sd = sd_value)

# Display the mean of the sample
cat("Mean of the generated sample:", mean(normal_sample), "\n")

# Repeat the exercise by generating a new sample and calculating the mean again
new_sample <- rnorm(n, mean = mean_value, sd = sd_value)
cat("Mean of the new generated sample:", mean(new_sample), "\n")

# What happens with smaller and larger sample sizes?
sample_size_small <- 10
sample_size_large <- 1000
small_sample <- rnorm(sample_size_small, mean = mean_value, sd = sd_value)
large_sample <- rnorm(sample_size_large, mean = mean_value, sd = sd_value)

cat("Mean of small sample (size 10):", mean(small_sample), "\n")
cat("Mean of large sample (size 1000):", mean(large_sample), "\n")
```

Explanation

- **Normal Distribution:**
  - The Normal distribution is defined by its mean μ and standard deviation σ.
  - Here, the mean is 2, and the standard deviation is 1.
- **R Function:**

  - `rnorm(n, mean, sd)` generates n random samples from a Normal distribution with the specified mean and standard deviation.
  - `n = 100`: Specifies a sample size of 100.
  - `mean = 2`: Specifies the mean of the distribution.
  - `sd = 1`: Specifies the standard deviation of the distribution.
- **Sample Size Impact:**
  - Larger sample sizes tend to provide a more accurate estimate of the population mean.
  - Smaller sample sizes may have more variability in the sample mean.
- **Repetition of the Exercise:**
  - Repeating the exercise with different random samples allows us to observe how the sample mean varies.
  - Sample means should be close to 2, but they will fluctuate.

Related Concepts

- **Central Limit Theorem:** As the sample size increases, the sample mean tends to the population mean, and the distribution of sample means approaches normality.
- **Random Number Generation:** R provides functions for generating random samples from various distributions, e.g., `rnorm()`, `runif()`.
- **Sample Size and Variability:** Larger samples reduce variability in the sample mean, while smaller samples introduce more randomness.

Related Questions

1. What happens to the mean when the sample size is smaller (e.g., size 10)?
   - **Answer:** The sample mean may be further from the true mean, as smaller samples have more variability.
2. What happens when the sample size is larger (e.g., size 1000)?
   - **Answer:** The sample mean tends to be closer to the true mean, as larger samples provide a more accurate estimate.
3. How would you visualize the distribution of the sample means for different sample sizes?
   - **Answer:** Use the following code to visualize the distribution of means for different sample sizes:

```
sample_means <- replicate(1000, mean(rnorm(100, mean = 2, sd = 1)))
hist(sample_means, main = "Distribution of Sample Means (n = 100)", xlab
```

Coin Toss Simulation in R

Problem Statement

A tosses a fair coin until he gets a Head. B tosses a coin as many times as A took to get the first Head. Write an R code to get an idea of the expected number of Heads obtained by B in her tosses. Explain why this number should be less than 1.

R Code

```
# Define a function to simulate A's tosses
simulate_A <- function() {
  tosses <- 0
  while (TRUE) {
    tosses <- tosses + 1
    if (runif(1) < 0.5) {  # 50% chance of getting a Head
      break
    }
  }
  return(tosses)
}

# Define a function to simulate B's tosses based on A's tosses
simulate_B <- function(A_tosses) {
  heads_count <- 0
  for (i in 1:A_tosses) {
    if (runif(1) < 0.5) {  # 50% chance of getting a Head
      heads_count <- heads_count + 1
    }
  }
  return(heads_count)
}

# Run the simulation for multiple iterations
set.seed(123) # Set seed for reproducibility
num_simulations <- 10000
total_heads <- 0

for (i in 1:num_simulations) {
  A_tosses <- simulate_A()        # A's tosses to get the first Head
  heads_B <- simulate_B(A_tosses) # B's tosses based on A's tosses
  total_heads <- total_heads + heads_B
}

# Calculate the average number of Heads obtained by B
```

```
expected_heads <- total_heads / num_simulations
cat("Expected number of Heads obtained by B:", expected_heads, "\n")
```

Explanation

- **Simulating A's Tosses:**
  - A continues tossing until the first Head appears. This is modeled by incrementing tosses until a random number less than 0.5 is generated (representing a Head).
- **Simulating B's Tosses:**
  - B tosses the coin the same number of times as A's tosses, with each toss having a 50
- **Expected Number of Heads:**
  - The expected number of tosses A makes is 2, and since B has a 50

Why the Expected Number Should Be Less Than 1

- A's tosses follow a geometric distribution, where the expected number of tosses until the first Head is 2.
- B tosses the coin the same number of times as A, and each toss has a 50
- Therefore, on average, B will get 1 Head, but due to the variability, the actual mean across many simulations will be slightly less than 1.

Related Concepts

- **Geometric Distribution:** The number of tosses A makes follows a geometric distribution, and the expected number of tosses until the first Head is $\frac{1}{p} = 2$ for $p = 0.5$.
- **Expected Value:** The expected number of Heads obtained by B is the average number of Heads across many trials.
- **Random Simulation:** Random simulations allow approximation of theoretical expected values when an analytical solution is complex.

Related Questions

1. What would happen if A's coin was biased, say the probability of a Head is 0.7?
   - **Answer:** A's expected number of tosses would be $\frac{1}{0.7} \approx 1.43$, and B's expected number of Heads would be $\frac{1.43}{2}$.

2. How can you visualize the distribution of the number of Heads obtained by B?

    - **Answer:** Use a histogram to visualize the distribution of Heads obtained by B in many trials.

3. What if B performed multiple trials (e.g., tosses) for each round?

    - **Answer:** This would change the expected number of Heads and could be modeled by adjusting the number of tosses.

Statistical Methods II: Week 3 Assignment (due 30 January, 2025)

1. Write an R code to create a geometric progression with a specified first term and a specified common ratio.

2. Suppose n is the the length of an arithmetic progression (AP) and i is a positive integer smaller than n. Write an R code (without a loop) to demonstrate that the sum of the two numbers, which are the ith terms of the AP from the left and from the right, does not depend on i.

3. Write an R code that can generate an integer sequence that contains the successive digits of the number π, by using a for loop.

4. How many digits after the decimal place can be produced by the above code?

5. Write an R code without a loop to evaluate the median of the gamma distribution with the scale parameter 1 and the shape parameter 1.5, up to the first place after decimal. (You may use the R function `pgamma` to evaluate the cumulative distribution function of this distribution.)

6. Write an R code to generate a sample of size 25 from the standard normal distribution, store it in the array x, and compute the difference between the first element of x and the mean of the remaining 24 elements. (This will be the prediction error if the first data value is lost and is "predicted" by the mean of the available values.)

7. Extend the R code in the above problem to calculate the "cross validation sum of squares" of the prediction error
$$\sum_{i=1}^{25}(x_i - \hat{x}_i)^2,$$
where x_i is the ith element of x and $\hat{x}_i$ is its prediction through the mean of the remaining 24 elements of x.

8. Write an R code to load the package `lmreg` and store the data set `worldpop` of the package into an R object named by you.

9. Write an R code to plot the increase in the midyear population of the world in successive years, as computed from the above data set.

1. R Code for Geometric Progression

```
geometric_progression <- function(a, r, n) {
  return(a * r^(0:(n-1)))
}

# Example usage
first_term <- 2   # Specify the first term
common_ratio <- 3 # Specify the common ratio
num_terms <- 10   # Specify the number of terms

gp_series <- geometric_progression(first_term, common_ratio, num_terms)
print(gp_series)
```

Geometric Progression (GP)

A geometric progression (or geometric sequence) is a sequence of numbers where each term after the first is found by multiplying the previous one by a fixed, non-zero number called the common ratio (r).

General Formula

$$T_n = a \cdot r^{(n-1)}$$

where:

- a = First term
- r = Common ratio
- n = Number of terms

Sum of First n Terms of a GP

$$S_n = a\frac{1 - r^n}{1 - r}, \quad \text{if } r \neq 1$$

Related Questions and Answers

Q1: How do you find the nth term of a geometric sequence?

A: The nth term is given by:

$$T_n = a \cdot r^{(n-1)}$$

Q2: What happens when the common ratio is 1?

A: The sequence remains constant, as each term is the same as the first term.

Q3: What is an infinite geometric series?

A: An infinite geometric series is a series with infinitely many terms, given by:

$$S = \frac{a}{1-r}, \quad \text{if } |r| < 1$$

2. R Code for Arithmetic Progression Sum

```
arithmetic_progression <- function(a, d, n, i) {
  left_term <- a + (i - 1) * d
  right_term <- a + (n - i) * d
  sum_terms <- left_term + right_term
  return(sum_terms)
}

# Example usage
first_term <- 3  # Specify the first term
common_diff <- 2 # Specify the common difference
num_terms <- 10  # Specify the number of terms
i_value <- 4     # Specify the term index

result <- arithmetic_progression(first_term, common_diff, num_terms, i_value)
print(result)
```

Arithmetic Progression (AP)

An arithmetic progression (AP) is a sequence of numbers in which the difference between consecutive terms remains constant.

General Formula for the nth Term

$$T_n = a + (n-1)d$$

where:

- a = First term
- d = Common difference
- n = Number of terms

Sum of First n Terms of an AP

$$S_n = \frac{n}{2}\left(2a + (n-1)d\right)$$

Proof: The Sum of Two Symmetric Terms is Constant

Left and Right Terms

The ith term from the left:

$$T_i = a + (i-1)d$$

The ith term from the right:

$$T_{n-i+1} = a + (n-i)d$$

Their sum:

$$S = T_i + T_{n-i+1} = (a + (i-1)d) + (a + (n-i)d)$$

Simplifying:

$$S = 2a + (n-1)d$$

This sum is independent of i.

Related Questions and Answers

Q1: How do you find the sum of an arithmetic series?

A: The sum of the first n terms is:

$$S_n = \frac{n}{2}(2a + (n-1)d)$$

Q2: What happens when $d = 0$ in an arithmetic sequence?

A: The sequence remains constant, as every term is equal to a.

Q3: Why does the sum of symmetric terms remain constant?

A: The symmetric terms are always positioned in a way that their sum simplifies to $2a + (n-1)d$, which is independent of i.

3. R Code for Generating Pi Digits

```
generate_pi_digits <- function(n) {
  pi_string <- as.character(pi)  # Convert  to a string
  pi_digits <- unlist(strsplit(substr(pi_string, 3, n + 2), "")) # Extract digits
  pi_digits <- as.integer(pi_digits) # Convert characters to integers
  return(pi_digits)
}

# Example usage
num_digits <- 10  # Specify the number of digits to extract
pi_sequence <- generate_pi_digits(num_digits)
print(pi_sequence)
```

Digits of Pi (π)

The number π is an **irrational number**, meaning it has an infinite number of non-repeating decimal places:

$$\pi = 3.14159265358979\ldots$$

Generating Pi in Programming

Since π is irrational, generating its digits programmatically requires:

- Using built-in constants like `pi` in R.
- Using mathematical formulas (e.g., **Leibniz series**, **BBP formula**).
- Extracting digits from a string representation.

Related Questions and Answers

Q1: Why is π an irrational number?

A: π cannot be expressed as a fraction of two integers ($\frac{p}{q}$), and its decimal expansion never terminates or repeats.

Q2: What is a famous series expansion for π?

A: The **Leibniz series**:

$$\pi = 4\sum_{k=0}^{\infty}\frac{(-1)^k}{2k+1}$$

This formula converges very slowly to π.

Q3: How can π be computed efficiently?

A: The **Bailey–Borwein–Plouffe (BBP) formula** allows computing any digit of π without computing previous ones.

4. R Code to Determine Maximum Digits of Pi

```
max_pi_digits <- function() {
  pi_string <- as.character(pi)  # Convert  to a string
  pi_digits <- nchar(substr(pi_string, 3, nchar(pi_string))) # Count decimal digits
  return(pi_digits)
}

# Example usage
num_producible_digits <- max_pi_digits()
print(num_producible_digits)
```

Precision of π in R

The built-in constant `pi` in R is stored as a **double-precision floating-point number**. This means π is represented with approximately **16 significant digits** in R.

Floating-Point Precision

R uses the **IEEE 754 double-precision format**, which provides about **15–16 decimal digits** of accuracy. Higher precision requires external libraries such as **MPFR (Multiple Precision Floating-Point Reliable)**.

Related Questions and Answers

Q1: How many decimal places of π does R store?

A: R stores π with approximately **16 decimal digits** of precision.

Q2: Can R compute more digits of π?

A: Not directly using `pi`, but libraries like **Rmpfr** can compute π to arbitrary precision.

Q3: How does floating-point precision affect π calculations?

A: Since R uses **double precision (IEEE 754)**, operations involving π may suffer from **round-off errors**.

5. R Code to Compute the Median of the Gamma Distribution

```
gamma_median <- function() {
  median_value <- uniroot(function(x) pgamma(x, shape = 1.5, scale = 1) - 0.5,
                          lower = 0, upper = 10)$root
  return(round(median_value, 1))
}

# Example usage
result <- gamma_median()
print(result)
```

Gamma Distribution

The **Gamma distribution** is a continuous probability distribution used to model waiting times, among other applications.

Probability Density Function (PDF)

$$f(x) = \frac{x^{k-1} e^{-x/\theta}}{\theta^k \Gamma(k)}, \quad x > 0$$

where:

- k is the **shape** parameter.
- θ is the **scale** parameter.
- $\Gamma(k)$ is the Gamma function.

Cumulative Distribution Function (CDF)

$$F(x) = P(X \leq x) = \frac{1}{\Gamma(k)} \int_0^x t^{k-1} e^{-t} dt$$

Mean and Variance

$$\text{Mean} = k\theta, \quad \text{Variance} = k\theta^2$$

Finding the Median

The **median** M is the value satisfying:

$$P(X \leq M) = 0.5$$

To find M, we solve:

$$\text{pgamma}(M, 1.5, 1) = 0.5$$

using the `uniroot()` function in R.

Related Questions and Answers

Q1: What does the `pgamma` function in R do?

A: It evaluates the CDF of the Gamma distribution:

$$F(x) = P(X \leq x)$$

Q2: How does `uniroot()` help in finding the median?

A: It finds the root of an equation, which is useful for solving $F(x) = 0.5$.

Q3: What is the difference between shape and scale in a Gamma distribution?

A: The **shape** parameter k determines the form of the distribution, while the **scale** parameter θ stretches or compresses it.

6. R Code to Compute Prediction Error

```
set.seed(123)  # For reproducibility
x <- rnorm(25)  # Generate a sample of size 25 from standard normal distribution

# Compute prediction error
prediction_error <- x[1] - mean(x[-1])

# Print result
print(prediction_error)
```

Standard Normal Distribution

The **standard normal distribution** is a normal distribution with:

$$\mu = 0, \quad \sigma^2 = 1$$

where:

- μ is the mean.
- σ^2 is the variance.

Prediction Error in Missing Data

If a value is missing, a common strategy is to **predict** it using the mean of the available values. The difference between the actual and predicted value is the **prediction error**.

Mean as an Estimator

For a normal distribution:

$$\hat{\mu} = \frac{1}{n} \sum_{i=1}^{n} X_i$$

is an **unbiased estimator** of μ.

Related Questions and Answers

Q1: Why do we use `set.seed(123)`?

A: It ensures that the random numbers generated are the same every time the code is run, allowing reproducibility.

Q2: Why do we exclude the first element when computing the mean?

A: We assume the first value is lost and predict it using the mean of the remaining data.

Q3: What is the expected value of the prediction error?

A: Since the sample is from $\mathcal{N}(0, 1)$, the expected mean is 0, and the prediction error has an expected value of 0.

7. R Code to Compute Cross-Validation Sum of Squares (CVSS)

```
set.seed(123)  # For reproducibility
x <- rnorm(25)  # Generate a sample of size 25 from standard normal distribution

# Compute cross-validation sum of squares (CVSS)
cvss <- sum((x - sapply(1:25, function(i) mean(x[-i])))^2)

# Print result
print(cvss)
```

Cross-Validation

Cross-validation is a technique for **assessing predictive models**. It involves removing one observation, predicting it from the rest, and repeating for all data points.

Leave-One-Out Cross-Validation (LOO-CV)

- We predict each data point using the **mean of the remaining** data points.
- The sum of squared errors gives the **cross-validation sum of squares (CVSS)**.

Formula

$$\sum_{i=1}^{25} e_i^2 = \sum_{i=1}^{25} (x_i - \hat{x}_i)^2$$

where: - x_i is the original value. - $\hat{x}_i$ is the predicted value:

$$\hat{x}_i = \frac{1}{24} \sum_{\substack{j=1 \\ j \neq i}}^{25} x_j$$

Bias-Variance Tradeoff

- Using the **mean of remaining observations** is an **unbiased estimator**. - However, if the data is **highly variable**, the prediction errors can be large.

Related Questions and Answers

Q1: Why is the mean used as the predictor?

A: The sample mean is an **unbiased** and **minimum variance** estimator under the normal distribution.

Q2: How does CVSS relate to prediction accuracy?

A: A **lower CVSS** indicates **better predictive performance**.

Q3: Why do we use `sapply()` instead of a loop?

A: `sapply()` is **vectorized**, making it **faster and more efficient** than loops in R.

8. R Code to Load the `lmreg` Package and Access the `worldpop` Dataset

```
# Install and load the lmreg package (if not already installed)
if (!requireNamespace("lmreg", quietly = TRUE)) {
    install.packages("lmreg")
}
library(lmreg)

# Load the worldpop dataset into an R object
world_population <- worldpop

# Display the first few rows
head(world_population)
```

R Packages

- An R **package** is a collection of functions and datasets. - `install.packages("package_name")` is used to install packages. - `library(package_name)` is used to load packages.

Datasets in R Packages

- Many R packages include **built-in datasets**. - Datasets can be accessed directly **after loading the package**.

World Population Data

- The `worldpop` dataset contains **population-related data** for different years.
- Such datasets are often used for **regression analysis and trend prediction**.

Related Questions and Answers

Q1: What if `lmreg` is not available in CRAN?

A: You can install it manually from another source or check if it's been renamed or deprecated.

Q2: How to check the structure of `worldpop`?

A: Use:

```
str(world_population)
```

Q3: How to visualize world population trends?

A: Use:

```
plot(world_population)
```

9. R Code to Plot Population Increase

```
# Install and load the lmreg package (if not already installed)
if (!requireNamespace("lmreg", quietly = TRUE)) {
    install.packages("lmreg")
}
library(lmreg)

# Load the worldpop dataset
world_population <- worldpop

# Compute the increase in midyear population
population_increase <- diff(world_population$population)

# Plot the population increase over successive years
plot(world_population$year[-1], population_increase, type = "o", col = "blue",
     xlab = "Year", ylab = "Population Increase",
     main = "Increase in Midyear Population Over Time")

# Add grid for better readability
grid()
```

Population Growth

- **Absolute Increase**: Difference between successive years. - **Exponential Growth**: When population grows proportionally to its size.

Difference Operator (`diff()`)

- `diff(x)` computes **first-order differences**: $x_t - x_{t-1}$.

Time Series Visualization

- **Line plots** (`type = "o"`) are useful for trend analysis. - **Grids** improve clarity.

Related Questions and Answers

Q1: Why do we use `diff()`?

A: It computes **changes** between consecutive years instead of absolute values.

Q2: What does `type = "o"` do?

A: It **connects points** with a line while marking them.

Q3: How to make the plot more visually appealing?

A: You can use `ggplot2`:

```
library(ggplot2)
ggplot(data.frame(year = world_population$year[-1], increase = population_increase),
       aes(x = year, y = increase)) +
    geom_line(color = "blue") +
    geom_point(color = "red") +
    labs(title = "Increase in Midyear Population Over Time",
         x = "Year", y = "Population Increase") +
    theme_minimal()
```

Statistical Methods II: Week 4 Assignment (due 3 February, 2025)

1. Write an R code to read the data set `airpoll` from the website `https://people.stat.sc.edu/Hitchcock/airpoll.txt`, and to create a data frame with city names as row labels.

2. Continuing with the `airpoll` data, write an R code to generate a bubble plot for displaying the values of the variables `Popden`, `SO2` and `Mortality`, by judiciously choosing the variables for the axes and the bubble size. What can you conclude from the plot?

3. Comment on the nature of the parallel lines plot of data on two strongly correlated variables.

4. Consider a parallel lines plot of data on three variables, two of which are highly correlated. Which ordering of the three variables will be least suitable for the correlation to be apparent from the plot?

5. Write an R code to generate a parallel lines plot for displaying the values of the variables `Sepal.Length`, `Sepal.Width`, `Petal.Length` and `Petal.Width` of the `iris` data, by judiciously choosing the order of the variables. What can you conclude from the plot?

6. A star plot undergoes a rotation if the variables are permuted while keeping the cyclic order (e.g., if 1-2-3-4 is permuted as 2-3-4-1). How many ways can one draw a star plot of p variables, without distinguishing between rotations of the plots?

7. Continuing with the `iris` data, write an R code to plot the variable `Sepal.Length` against the variable `Petal.Width`, while using different colours for the data for different species.

8. Continuing with the `iris` data, write an R code to plot the variable `Sepal.Length` against the variable `Petal.Width`, while using different colours for `Petal.Length` above and below 4.55.

9. Continuing with the `iris` data, write an R code to generate comparative box plots of the variable `Sepal.Width` for the three species.

10. Continuing with the `iris` data, write an R code to create a 'centered' version of the variable `Sepal.Width`, by subtracting a suitable real number from the vector, such that its mean is 0.

11. Can you compute the variance of `Sepal.Length` from the 'centered' version of the variable obtained above, without using the `var` or the `sd` functions?

12. Continuing with the `iris` data, describe what is accomplished by the following R code? Explain.

```
irc <- as.matrix(t(iris[,-5]) - colMeans(iris[,-5])); diag(irc%*%t(irc))/dim(irc[2])
```

1. Reading Data in R and Setting Row Names

Shaan

February 2, 2025

1 R Code to Read the Dataset and Create a Data Frame

Copy-paste this code into R:

```
# Read the dataset from the given URL
url <- "https://people.stat.sc.edu/Hitchcock/airpoll.txt"
airpoll_data <- read.table(url, header = TRUE)

# Convert the first column (City names) into row names
rownames(airpoll_data) <- airpoll_data[, 1]

# Remove the first column since it's now set as row names
airpoll_data <- airpoll_data[, -1]

# Display the first few rows of the data frame
head(airpoll_data)
```

2 Related Concepts

2.1 Reading Data from a URL in R

The `read.table()` function allows us to read structured data from an external source. The `header = TRUE` argument ensures that the first row is treated as column names.

2.2 Setting Row Names in a Data Frame

We can set row names using:

```
rownames(df) <- df[,1]
```

2.3 Removing a Column from a Data Frame

To remove a specific column:

```
df <- df[, -1]
```

2.4 Displaying Data

To display the first few rows:

```
head(df)
```

3 Related Questions and Answers

3.1 Q1: How can we read a CSV file from a URL into R?

Answer: We can use the `read.csv()` function:

```
url <- "https://example.com/data.csv"
data <- read.csv(url, header = TRUE)
head(data)
```

3.2 Q2: How can we change column names in a data frame in R?

Answer: We can use the `colnames()` function:

```
colnames(airpoll_data) <- c("Pollutant1", "Pollutant2", "Pollutant3")
```

3.3 Q3: How do we check the structure of a data frame in R?

Answer: We use the `str()` function:

```
str(airpoll_data)
```

2. Bubble Plot for Air Pollution Data

Shaan

February 2, 2025

1 R Code to Generate a Bubble Plot for airpoll Data

Copy-paste this code into R:

```
# Load necessary library
library(ggplot2)

# Read the dataset from the given URL
url <- "https://people.stat.sc.edu/Hitchcock/airpoll.txt"
airpoll_data <- read.table(url, header = TRUE)

# Convert the first column (City names) into row names
rownames(airpoll_data) <- airpoll_data[, 1]

# Remove the first column since it's now set as row names
airpoll_data <- airpoll_data[, -1]

# Create the bubble plot
ggplot(airpoll_data, aes(x = Popden, y = Mortality, size = SO2)) +
  geom_point(alpha = 0.5, color = "blue") +
  labs(title = "Bubble Plot of Population Density, SO2, and Mortality",
       x = "Population Density",
       y = "Mortality",
       size = "SO2 Concentration") +
  theme_minimal()
```

2 Conclusion from the Bubble Plot

- Cities with **higher SO_2 levels** tend to have **higher mortality rates**, suggesting a potential correlation. - Areas with **higher population density** generally exhibit increased **SO_2 pollution and mortality**. - The **bubble size represents SO_2 levels**—larger bubbles indicate higher pollution.

3 Related Concepts

3.1 Bubble Plots in R

A bubble plot is an extension of a scatter plot where the size of points represents an additional variable.

3.2 ggplot2 for Data Visualization

- `aes(size = variable_name)` in `ggplot2` maps the bubble size to a variable.
- `alpha` controls transparency.

3.3 Air Pollution and Health

- **SO_2 (Sulfur Dioxide)** is a pollutant linked to respiratory diseases. - **Higher SO_2** levels are associated with **increased mortality rates**.

4 Related Questions and Answers

4.1 Q1: How do we create a scatter plot instead of a bubble plot?

Answer: Remove the `size` argument:

```
ggplot(airpoll_data, aes(x = Popden, y = Mortality)) +
  geom_point(color = "blue") +
  labs(title = "Scatter Plot of Population Density and Mortality",
       x = "Population Density",
       y = "Mortality") +
  theme_minimal()
```

4.2 Q2: How can we modify the color of the bubbles based on another variable?

Answer: Use the `color` aesthetic:

```
ggplot(airpoll_data, aes(x = Popden, y = Mortality, size = SO2, color = SO2)) +
  geom_point(alpha = 0.5) +
  labs(title = "Bubble Plot with Color Mapping",
       x = "Population Density",
       y = "Mortality",
       size = "SO2 Concentration",
       color = "SO2 Level") +
  theme_minimal()
```

4.3 Q3: How can we log-transform an axis to improve visibility in a ggplot?

Answer: Use `scale_x_log10()` or `scale_y_log10()`:

```
ggplot(airpoll_data, aes(x = Popden, y = Mortality, size = SO2)) +
  geom_point(alpha = 0.5, color = "blue") +
  scale_x_log10() +
  labs(title = "Log-Transformed Bubble Plot",
       x = "Log(Population Density)",
       y = "Mortality",
       size = "SO2 Concentration") +
  theme_minimal()
```

3. Parallel Coordinate Plot for Strongly Correlated Variables

Shaan

February 2, 2025

1 R Code to Generate a Parallel Coordinate Plot

Copy-paste this code into R:

```
# Load necessary library
library(MASS)  # For parallel plot
library(GGally)  # For correlation visualization

# Read the dataset from the given URL
url <- "https://people.stat.sc.edu/Hitchcock/airpoll.txt"
airpoll_data <- read.table(url, header = TRUE)

# Convert the first column (City names) into row names
rownames(airpoll_data) <- airpoll_data[, 1]

# Remove the first column since it's now set as row names
airpoll_data <- airpoll_data[, -1]

# Select two strongly correlated variables (e.g., SO2 and Mortality)
selected_data <- airpoll_data[, c("SO2", "Mortality")]

# Standardize the variables for better visualization
selected_data <- as.data.frame(scale(selected_data))

# Generate the parallel coordinate plot
parcoord(selected_data, col = "blue", var.label = TRUE, lty = 1)
```

2 Comment on the Nature of the Parallel Coordinate Plot

- **Parallel coordinate plots** are useful for visualizing high-dimensional data. - When **two variables are strongly correlated**, the parallel lines in the plot **tend to remain nearly parallel** rather than crossing frequently. - In the case of **positive correlation**, the lines move **in the same direction** (i.e., increasing together). - In the case of **negative correlation**, the lines move in **opposite directions** (i.e., one increases while the other decreases). - If SO_2 and Mortality are highly correlated, the parallel coordinate plot will show **smooth, nearly parallel lines** indicating their strong relationship.

3 Related Concepts

3.1 Parallel Coordinate Plots

- Used to visualize relationships between multiple continuous variables. - Lines represent data points, and the spacing between them shows correlations.

3.2 Correlation in Data

- **Strong correlation**: Lines in the parallel coordinate plot remain nearly parallel. - **Weak correlation**: Lines tend to cross each other frequently.

3.3 Standardization for Visualization

Scaling data ensures all variables are on the same scale for accurate visual comparison.

4 Related Questions and Answers

4.1 Q1: How can we check the correlation between two variables in R?

Answer: Use the `cor()` function:

```
cor(airpoll_data$SO2, airpoll_data$Mortality)
```

A value close to **1 or -1** indicates a strong correlation.

4.2 Q2: How can we visualize correlation using a scatter plot?

Answer:

```
ggplot(airpoll_data, aes(x = SO2, y = Mortality)) +
  geom_point(color = "blue") +
  geom_smooth(method = "lm", se = FALSE, color = "red") +
  labs(title = "Scatter Plot of SO2 vs Mortality",
       x = "SO2 Concentration",
       y = "Mortality") +
  theme_minimal()
```

This adds a **linear regression line** to visualize correlation.

4.3 Q3: How can we create a correlation matrix for multiple variables in R?

Answer: Use `GGally::ggpairs()`:

```
library(GGally)
ggpairs(airpoll_data[, c("SO2", "Mortality", "Popden")])
```

This generates **pairwise scatter plots and correlation coefficients**.

4. Parallel Coordinate Plot for Three Variables

Shaan

February 2, 2025

1 R Code to Generate a Parallel Lines Plot for Three Variables

Copy-paste this code into R:

```
# Load necessary library
library(MASS)  # For parallel plot

# Read the dataset from the given URL
url <- "https://people.stat.sc.edu/Hitchcock/airpoll.txt"
airpoll_data <- read.table(url, header = TRUE)

# Convert the first column (City names) into row names
rownames(airpoll_data) <- airpoll_data[, 1]

# Remove the first column since it's now set as row names
airpoll_data <- airpoll_data[, -1]

# Select three variables: SO2, Mortality, and Popden (assumed to be correlated)
selected_data <- airpoll_data[, c("SO2", "Mortality", "Popden")]

# Standardize the variables for better visualization
selected_data <- as.data.frame(scale(selected_data))

# Generate the parallel coordinate plot with three variables
parcoord(selected_data, col = "blue", var.label = TRUE, lty = 1)
```

2 Analysis of Least Suitable Ordering for Correlation Visibility

- **When two variables are highly correlated**, the parallel coordinate plot will show **lines that are almost parallel** when those two variables are next to each other. - The **least suitable ordering** of the three variables would be

when the two **highly correlated variables** are placed on **opposite ends** of the plot. This increases the likelihood that the lines will intersect and makes the correlation less visually apparent. - A **bad ordering example**: If we have three variables, such as $\mathbf{SO_2}$, **Mortality**, and **Population Density** (with $\mathbf{SO_2}$ and **Mortality** being highly correlated), placing **Population Density** in the middle and $\mathbf{SO_2}$ and **Mortality** at the extremes would obscure the correlation between $\mathbf{SO_2}$ and **Mortality**.

3 Related Concepts

3.1 Parallel Coordinate Plots

- Used to visualize relationships among multiple continuous variables. - When variables are **highly correlated**, their parallel lines remain closely aligned.

3.2 Impact of Ordering in Parallel Coordinate Plots

- When two **highly correlated variables** are **not adjacent**, the plot's ability to show the correlation is diminished. - **Good ordering** places highly correlated variables next to each other, keeping their lines parallel.

3.3 Correlation and Data Structure

- Visualizing **correlated variables** through parallel coordinate plots can help us understand the strength and direction of relationships.

4 Related Questions and Answers

4.1 Q1: How do we detect the correlation between three variables in R?

Answer: You can check correlations using the `cor()` function for all combinations:

```
cor(airpoll_data[, c("SO2", "Mortality", "Popden")])
```

This will return the correlation matrix.

4.2 Q2: How can we visualize three variables in a scatter plot matrix?

Answer:

```
pairs(airpoll_data[, c("SO2", "Mortality", "Popden")])
```

This will generate scatter plots for each pair of variables.

4.3 Q3: How can we highlight the effect of ordering in parallel coordinate plots?

Answer: Try rearranging the order of the variables in the `parcoord()` function to see the effect:

```
# Place highly correlated variables together
parcoord(selected_data[, c("SO2", "Mortality", "Popden")], col = "blue", var.label = TRUE, l

# Place highly correlated variables at opposite ends
parcoord(selected_data[, c("Popden", "SO2", "Mortality")], col = "blue", var.label = TRUE, l
```

This demonstrates the effect of ordering on correlation visibility.

5. Parallel Coordinate Plot for the Iris Dataset

Shaan

February 2, 2025

1 R Code to Generate a Parallel Lines Plot for the Iris Dataset

Copy-paste this code into R:

```
# Load necessary library
library(MASS)  # For parallel plot

# Load the Iris dataset
data(iris)

# Select the variables of interest: Sepal.Length, Sepal.Width, Petal.Length, Petal.Width
selected_data <- iris[, c("Sepal.Length", "Sepal.Width", "Petal.Length", "Petal.Width")]

# Standardize the variables for better visualization
selected_data <- as.data.frame(scale(selected_data))

# Generate the parallel coordinate plot with the selected variables
parcoord(selected_data, col = iris$Species, var.label = TRUE, lty = 1)
```

2 Analysis of the Plot

- **Sepal.Length** and **Petal.Length** are usually more highly correlated with each other compared to the other variable pairs, meaning placing these two variables next to each other will help their relationship stand out in the plot. - **Sepal.Width** and **Petal.Width** are expected to be less correlated with the other variables, and placing them in between the two highly correlated variables may not obscure their relationships. - The **Species** variable is used to color the lines and visually separate the three Iris species in the plot.

3 Related Concepts

3.1 Parallel Coordinate Plots

- Used for visualizing high-dimensional data, helping identify relationships between variables. - When highly correlated variables are next to each other, their lines are parallel, making it easy to spot correlations.

3.2 Correlation in the Iris Dataset

- **Sepal.Length** and **Petal.Length** are often strongly correlated. - **Sepal.Width** and **Petal.Width** tend to have weaker correlations with other variables.

3.3 Standardization

- Standardizing the data ensures that all variables are on the same scale, making the parallel coordinate plot more effective for comparison.

4 Related Questions and Answers

4.1 Q1: How can we determine the correlation between the variables in the Iris dataset?

Answer: Use the `cor()` function to check correlations between the variables:

```
cor(iris[, 1:4])
```

This will return the correlation matrix between the numeric variables in the dataset.

4.2 Q2: How can we color the lines based on species?

Answer: In the `parcoord()` function, we can pass the `Species` column to the `col` argument to color the lines:

```
parcoord(selected_data, col = iris$Species, var.label = TRUE, lty = 1)
```

This colors the lines based on the species of the Iris flower.

4.3 Q3: How can we visualize the relationship between the variables using a scatter plot matrix?

Answer: Use the `pairs()` function to generate scatter plots for all variable combinations:

```
pairs(iris[, 1:4], col = iris$Species)
```

This will show pairwise scatter plots colored by species.

6. Counting Distinct Star Plots with Cyclic Rotation Invariance

Shaan

February 2, 2025

1 Answer and Explanation

A **star plot** is a type of data visualization where multiple variables are represented as axes extending from a common point. The values of each variable are then plotted along these axes.

When permuting the variables in a star plot while keeping the cyclic order, the plot undergoes a **rotation**. The goal is to find how many distinct ways one can draw a star plot of p variables, without distinguishing between rotations of the plots.

1.1 Key Idea:

The main challenge is that rotating the star plot does not create a new plot; it's still the same star plot but in a different orientation. Therefore, the number of distinct star plots is determined by how many unique permutations of the variables exist, accounting for cyclic rotations.

1.2 Solution:

- The total number of ways to permute p variables is $p!$. - However, due to the cyclic nature of the plot, a rotation of the plot doesn't result in a different plot. Since there are p possible rotations, we need to divide by p. - Therefore, the number of distinct star plots is:

$$Numberofdistinctstarplots = \frac{p!}{p} = (p-1)!$$

2 Related Concepts

2.1 Star Plot

A star plot is a graphical representation of multivariate data where each variable is shown along an axis radiating from a central point. It is useful for comparing multiple variables across different observations.

2.2 Cyclic Order

A cyclic order refers to the fact that rotating the axes in a star plot does not change the plot. This means a star plot is invariant to rotations, and we need to count only distinct permutations of the variables that cannot be obtained through rotation.

2.3 Permutations and Rotations

The permutation of variables refers to the arrangement of variables in a specific order. When considering rotations, we treat any cyclic shift of the variables as the same arrangement.

2.4 Factorial Notation

The notation $p!$ represents the total number of permutations of p items. $(p-1)!$ represents the number of unique star plots when considering the rotational symmetry.

3 Related Questions and Answers

3.1 Q1: How many distinct ways can we arrange p variables in a linear plot?

Answer: In a linear plot, the number of distinct ways to arrange p variables is simply the number of permutations of p elements:

$$p!$$

This accounts for all possible arrangements, with no restrictions for cyclic rotations.

3.2 Q2: How do rotations affect visualizations like the star plot or other circular plots?

Answer: In visualizations such as the star plot or circular plots, rotations do not change the overall pattern of the data because the data points are symmetrically placed around the central point. This is why, when counting distinct visualizations, we account for rotations as the same.

3.3 Q3: What is the significance of the factorial notation in combinatorics?

Answer: Factorial notation $p!$ is used in combinatorics to represent the total number of ways to arrange p distinct objects. It is essential when considering all possible permutations or orderings of items.

7. Plotting Sepal Length vs Petal Width from the Iris Dataset

Shaan

February 2, 2025

1 R Code to Plot Sepal Length Against Petal Width

Copy-paste this code into R:

```
# Load necessary library
library(ggplot2)

# Load the Iris dataset
data(iris)

# Create the plot
ggplot(iris, aes(x = Sepal.Length, y = Petal.Width, color = Species)) +
  geom_point() +
  labs(title = "Sepal Length vs Petal Width",
       x = "Sepal Length",
       y = "Petal Width") +
  theme_minimal()
```

2 Explanation of the Code

- `ggplot(iris, aes(x = Sepal.Length, y = Petal.Width, color = Species))`: This specifies the Iris dataset, sets Sepal.Length as the x-axis, Petal.Width as the y-axis, and uses the Species variable to color the data points.
- `geom_`$point()$: $This function adds the data points to the plot.$`labs()` : $This adds a title and axis labels.$

- `theme_`$minimal()$: $This applies a minimal theme to the plot for a cleaner look.$

3 Related Concepts

3.1 ggplot2 Package

`ggplot2` is an R package used for creating high-quality data visualizations. It is based on the grammar of graphics and allows you to build plots by combining different components (e.g., data, aesthetics, geometry).

3.2 Aesthetic Mappings (aes)

In `ggplot2`, the `aes()` function is used to define the variables for the axes and other visual attributes, such as color, size, and shape. Here, we map `Sepal.Length` to the x-axis, `Petal.Width` to the y-axis, and `Species` to color.

3.3 Scatter Plot

A scatter plot is used to visualize the relationship between two continuous variables. By plotting `Sepal.Length` against `Petal.Width`, we can visually assess their relationship.

3.4 Categorical Color Coding

Using the `color` aesthetic in `ggplot2`, we can assign different colors to data points based on categories. In this case, the `Species` variable is categorical, so each species will have a different color.

4 Related Questions and Answers

4.1 Q1: How can we plot multiple variables in the same graph?

Answer: We can use **facets** in `ggplot2` or plot multiple variables by mapping them to different aesthetics like `color`, `size`, or `shape`. For instance, you can plot multiple variables by adding more `geom_` layers to the plot.

4.2 Q2: How can we modify the color scheme of the plot?

Answer: In `ggplot2`, we can modify the color palette using the `scale_color_manual()` function to manually assign specific colors to each species:

```
ggplot(iris, aes(x = Sepal.Length, y = Petal.Width, color = Species)) +
  geom_point() +
  scale_color_manual(values = c("setosa" = "red", "versicolor" = "green", "virginica" = "blu
  labs(title = "Sepal Length vs Petal Width",
       x = "Sepal Length",
       y = "Petal Width") +
  theme_minimal()
```

4.3 Q3: How can we visualize correlations between multiple variables?

Answer: You can use **pairwise scatter plots** using the `pairs()` function or a **correlation matrix heatmap** to visualize correlations between multiple variables. For instance:

```
pairs(iris[, 1:4], col = iris$Species)
```

8. Plotting Sepal Length vs Petal Width Colored by Petal Length Category

Shaan

February 2, 2025

1 R Code to Plot Sepal Length Against Petal Width Based on Petal Length Threshold

Copy-paste this code into R:

```
# Load necessary library
library(ggplot2)

# Load the Iris dataset
data(iris)

# Create a new variable to classify Petal.Length above and below 4.55
iris$Petal.Length.Category <- ifelse(iris$Petal.Length > 4.55, "Above 4.55", "Below 4.55")

# Create the plot
ggplot(iris, aes(x = Sepal.Length, y = Petal.Width, color = Petal.Length.Category)) +
  geom_point() +
  labs(title = "Sepal Length vs Petal Width (Colored by Petal Length)",
       x = "Sepal Length",
       y = "Petal Width") +
  scale_color_manual(values = c("Above 4.55" = "blue", "Below 4.55" = "red")) +
  theme_minimal()
```

2 Explanation of the Code

- `iris$Petal.Length.Category`: We create a new variable called `Petal.Length.Category`, which assigns a label based on whether `Petal.Length` is above or below 4.55. This is done using the `ifelse()` function.
- `ggplot(iris, aes(x = Sepal.Length, y = Petal.Width, color = Petal.Length.Category))`: This tells `ggplot2` to plot `Sepal.Length` on the x-axis, `Petal.Width` on the y-axis, and color the data points based on the newly created `Petal.Length.Category`.

- `geom_point()`: Adds the points to the plot.
- `scale_color_manual()`: Specifies the color scheme for the categories. Here, we assign "blue" for values above 4.55 and "red" for values below 4.55.
- `theme_minimal()`: Applies a minimal theme for a clean appearance.

3 Related Concepts

3.1 ggplot2 Package

`ggplot2` is a popular R package for creating high-quality visualizations. It works on the "grammar of graphics" and allows the user to define how the data should be visualized by specifying data, aesthetics, and geometries.

3.2 Conditional Coloring

Conditional coloring is a technique where the color of the data points (or other visual elements) is determined by a condition. In this case, we categorize the `Petal.Length` variable into two categories based on whether it is above or below 4.55.

3.3 ifelse() Function

The `ifelse()` function is used to apply a condition to the data. Here, it is used to classify each observation in the dataset as either "Above 4.55" or "Below 4.55" based on the value of `Petal.Length`.

3.4 Categorical Color Coding

Using the `color` aesthetic in `ggplot2`, you can assign different colors to data points based on categories. This is useful for distinguishing between different groups of data.

4 Related Questions and Answers

4.1 Q1: How can we create a new categorical variable based on continuous data?

Answer: We can create new categorical variables using the `ifelse()` function or the `cut()` function in R. For example, you can split a continuous variable into ranges and label the categories accordingly:

```
iris$Petal.Length.Category <- ifelse(iris$Petal.Length > 4.55, "Above 4.55", "Below 4.55")
```

4.2 Q2: How can we customize the color scheme of a plot in ggplot2?

Answer: You can customize the color scheme by using the `scale_color_manual()` function to manually assign colors to different categories. For instance:

```
scale_color_manual(values = c("Above 4.55" = "blue", "Below 4.55" = "red"))
```

4.3 Q3: How can we add a legend to a plot in ggplot2?

Answer: A legend is automatically added when you use color, size, or other aesthetics in `ggplot2`. However, you can customize the legend title using the `labs()` function:

```
labs(color = "Petal Length Category")
```

9. Comparative Box Plots of Sepal Width for the Three Species

Shaan

February 2, 2025

1 R Code to Generate Comparative Box Plots for Sepal.Width by Species

Copy-paste this code into R:

```
# Load necessary library
library(ggplot2)

# Load the Iris dataset
data(iris)

# Create a comparative box plot for Sepal.Width by Species
ggplot(iris, aes(x = Species, y = Sepal.Width, fill = Species)) +
  geom_boxplot() +
  labs(title = "Comparative Box Plots of Sepal Width for the Three Species",
       x = "Species",
       y = "Sepal Width") +
  theme_minimal()
```

2 Explanation of the Code

- `aes(x = Species, y = Sepal.Width, fill = Species)`: We are mapping the `Species` variable to the x-axis and the `Sepal.Width` variable to the y-axis. The `fill` aesthetic is used to color the box plots by species.
- `geom_boxplot()`: This function creates the box plots for each species.
- `labs(title = ..., x = ..., y = ...)`: Labels the plot with a title and axis labels.
- `theme_minimal()`: Applies a minimal theme to the plot for a clean appearance.

3 Related Concepts

3.1 Box Plot

A box plot is a graphical representation of the distribution of a dataset, showing the median, quartiles, and potential outliers. It is useful for comparing distributions between different groups or categories.

3.2 ggplot2 Package

`ggplot2` is a widely used R package for creating high-quality visualizations. It works on the principles of the "grammar of graphics," where you define layers to build complex plots.

3.3 Fill Aesthetic

The `fill` aesthetic is used to color the box plots based on the categorical variable (in this case, `Species`). It helps distinguish between different groups.

3.4 Comparative Visualization

A comparative box plot allows you to visually compare the distributions of a variable across different categories. Here, we are comparing the distribution of `Sepal.Width` for the three species in the Iris dataset.

4 Related Questions and Answers

4.1 Q1: What is the purpose of using a box plot?

Answer: A box plot is used to visualize the distribution of data, showing the median, quartiles, and outliers. It helps in understanding the spread of data and comparing distributions between different groups.

4.2 Q2: How can we color box plots based on a categorical variable in ggplot2?

Answer: We can color the box plots based on a categorical variable by using the `fill` aesthetic. For example:

```
ggplot(iris, aes(x = Species, y = Sepal.Width, fill = Species)) + geom_boxplot()
```

4.3 Q3: How can we add labels and titles to a plot in ggplot2?

Answer: In `ggplot2`, you can add titles and axis labels using the `labs()` function. For example:

```
labs(title = "Title", x = "X Axis Label", y = "Y Axis Label")
```

10. Centering the Sepal.Width Variable in the Iris Dataset

Shaan

February 2, 2025

1 R Code to Create a Centered Version of Sepal.Width

Copy-paste this code into R:

```
# Load the Iris dataset
data(iris)

# Center the Sepal.Width variable by subtracting its mean
iris$Centered_Sepal.Width <- iris$Sepal.Width - mean(iris$Sepal.Width)

# Display the first few rows of the dataset to see the new column
head(iris)
```

2 Explanation of the Code

- `mean(iris$Sepal.Width)`: This function calculates the mean of the `Sepal.Width` variable.
- `iris$Sepal.Width - mean(iris$Sepal.Width)`: We subtract the mean from each value in the `Sepal.Width` variable to center it.
- `iris$Centered_Sepal.Width`: This creates a new column in the dataset with the centered values.
- `head(iris)`: Displays the first few rows of the modified dataset to confirm the new column has been added.

3 Related Concepts

3.1 Centering a Variable

Centering a variable involves subtracting its mean from each data point. This process ensures the transformed variable has a mean of 0. It's commonly used in statistical analysis to simplify interpretation, especially in regression modeling.

3.2 Mean of a Variable

The mean (or average) of a variable is calculated by summing all the values and dividing by the number of observations. Centering relies on the mean to shift the data distribution.

3.3 Data Preprocessing

Centering is a common data preprocessing step before performing various statistical analyses, particularly when dealing with multivariate data. It makes the data easier to interpret and scale.

3.4 Effect of Centering on Data

Centering does not affect the shape or spread of the data but shifts the distribution so that its mean is 0. This is helpful for algorithms that are sensitive to the scale of data (e.g., principal component analysis).

4 Related Questions and Answers

4.1 Q1: Why is centering a variable important in data analysis?

Answer: Centering a variable can help make interpretation easier, especially when working with multiple variables in regression models. It ensures that the variable has a mean of 0, which can reduce multicollinearity and make the coefficients in regression models more interpretable.

4.2 Q2: How do you compute the mean of a variable in R?

Answer: You can compute the mean of a variable using the `mean()` function in R. For example:

```
mean(iris$Sepal.Width)
```

4.3 Q3: What happens to the distribution of a variable when you center it?

Answer: Centering shifts the data so that the mean becomes 0, but it does not change the spread, variance, or shape of the data. It simply shifts the entire distribution to the left or right.

11. Computing the Variance of Sepal.Length from the Centered Version in the Iris Dataset

Shaan

February 2, 2025

1 R Code to Compute the Variance of Sepal.Length from the Centered Version

Copy-paste this code into R:

```
# Load the Iris dataset
data(iris)

# Center the Sepal.Length variable by subtracting its mean
centered_sepal_length <- iris$Sepal.Length - mean(iris$Sepal.Length)

# Compute the variance manually (without using var or sd functions)
n <- length(centered_sepal_length)  # Number of data points
variance <- sum(centered_sepal_length^2) / n

# Display the computed variance
variance
```

2 Explanation of the Code

- `centered_sepal_length`: The centered version of `Sepal.Length` is obtained by subtracting the mean of `Sepal.Length` from each value in the variable.
- `sum(centered_sepal_length`^{2}`)` : $We square each element of the centered$ `Sepal.Length` $and sum them up.$ `n` : $This is the number of data points in the$ `Sepal.Length` $variable.$
- `variance`: The variance is calculated by dividing the sum of squared values by the number of data points.

3 Related Concepts

3.1 Variance

Variance measures how far a set of data points are spread out from the mean. In this case, we are calculating the variance of the centered `Sepal.Length`, which gives us the dispersion of the data around its mean of 0.

3.2 Centering and Variance

When a variable is centered by subtracting its mean, the variance is calculated by the average of the squared differences from the mean, which is equivalent to the sum of squared values in the centered data.

3.3 Manual Calculation of Variance

Variance can be computed manually using the formula, which involves squaring the differences from the mean, summing them, and then dividing by the number of data points.

3.4 Standard Deviation

Standard deviation is simply the square root of the variance. While we aren't calculating it here, it is often used to interpret variance in the same units as the original data.

4 Related Questions and Answers

4.1 Q1: What is the formula for calculating variance?

Answer: The formula for variance is:

$$Variance = \frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^2$$

Where x_i are the data points, $\bar{x}$ is the mean, and n is the number of data points.

4.2 Q2: How do you calculate variance without using built-in functions in R?

Answer: To calculate variance manually, subtract the mean from each data point, square the result, sum these squared values, and then divide by the number of data points:

```
variance <- sum((data - mean(data))^2) / length(data)
```

4.3 Q3: Why is centering the data important before calculating variance?

Answer: Centering the data (subtracting the mean) ensures that the variance reflects the spread of the data around 0. It simplifies calculations in certain statistical methods, such as principal component analysis, where the variance of centered data is more interpretable.

12. Explanation of R Code for Variance Calculation from Centered Data

Shaan

February 2, 2025

1 R Code Analysis and Explanation

The R code provided is:

```
irc <- as.matrix(t(iris[,-5]) - colMeans(iris[,-5]))
diag(irc %*% t(irc)) / dim(irc[2])
```

1.1 Step-by-Step Explanation

- `iris[,-5]`: Selects all columns of the iris dataset except the 5th column (Species).
- `colMeans(iris[,-5])`: Calculates the mean for each variable in the dataset.
- `t(iris[,-5])`: Transposes the dataset, swapping rows and columns.
- `t(iris[,-5]) - colMeans(iris[,-5])`: Subtracts the mean from each variable, centering the data.
- `as.matrix(...)`: Converts the centered data to a matrix format.
- `irc %*% t(irc)`: Multiplies the centered data matrix by its transpose, resulting in a matrix with sum of squared deviations.
- `diag(irc %*% t(irc))`: Extracts the diagonal elements of the matrix, which represent the variances.
- `diag(irc %*% t(irc)) / dim(irc[2])`: Divides the sum of squared deviations by the number of observations to compute the variance.

2 What is Accomplished by the Code?

This code computes the variance for each variable in the iris dataset after centering the data (subtracting the mean). The process involves matrix operations to calculate the sum of squared deviations and then normalize by the number of observations.

3 Related Concepts

3.1 Centering the Data

Centering involves subtracting the mean from each data point, ensuring each variable has a mean of 0.

3.2 Variance

Variance measures how spread out the values in a dataset are from the mean.

3.3 Matrix Multiplication

Matrix multiplication is used here to efficiently calculate the sum of squared deviations for all variables.

3.4 Covariance Matrix

Multiplying the centered data by its transpose produces a covariance matrix, from which the variances are extracted.

4 Related Questions and Answers

4.1 Q1: Why do we center the data before calculating the variance?

Answer: Centering removes bias caused by the scale or location of the data, ensuring variance is calculated relative to the mean.

4.2 Q2: What is the difference between variance and covariance?

Answer: Variance measures the spread of a single variable, while covariance measures the relationship between two variables.

4.3 Q3: Why is matrix multiplication used to compute the variance in this code?

Answer: Matrix multiplication allows for efficient computation of the sum of squared deviations between variables, making the process faster and more scalable for multivariate datasets.

Statistical Methods II: Week 5 Assignment (due 10 February, 2025)

1. Show that a function g of two random variables, X_1 and X_2, best approximates a third random variable Y in the sense of the smallest mean squared error of approximation, $E[\{Y - g(X_1, X_2)\}^2]$, when $g(X_1, X_2) = E(Y|X_1, X_2)$. [You can work initially with the conditional mean squared error, $E[\{Y - g(X_1, X_2)\}^2|X_1, X_2]$.]

2. Suppose a random variable Y has to be approximated by a *linear* function $\beta_0 + \beta_1 X_1 + \beta_2 X_2$ of two random variables, X_1 and X_2, in the sense of the smallest mean squared error of approximation, $E[\{Y - (\beta_0 + \beta_1 X_1 + \beta_2 X_2)\}^2]$. All the three random variables have zero mean. For which values of the the three parameters is the best approximation achieved? [You can answer the question by expressing the mean squared error in terms of the variances and covariances.]

3. How does the answer to the above question change when the random variables have non-zero mean?

4. An important problem of optical and electrical communication is to detect whether a signal is present in a sequence of observations, which is modelled as

$$y_t = a\cos(2\pi f t + \phi) + \varepsilon_t, \quad t = 1, 2, \ldots, n,$$

   where t is the time index, $a\cos(2\pi f t + \phi)$ is a sinusoidal function of t (called signal) having amplitude a, frequency f and phase ϕ, and ε_t is the error at time t (called noise). If a is zero, the observations consist only of noise, that is, there is no signal. Assuming that the frequency f is known, show that the above signal plus noise model with unspecified amplitude and phase reduces to a multiple linear regression model after a suitable transformation of the parameters. What are the explanatory variables of this linear model?

5. The salary (y) of an employee in an organization is modelled as

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3 + \beta_4 x_4 + \varepsilon,$$

   where x_1 and x_2 are binary indicators of graduation from high-school and college, respectively, x_3 is the indicator of at least one post graduate degree, x_4 is the number of years in service and ε is the error term of the model. What are the possible sources of the model error?

6. Consider the piecewise linear (broken line) regression model

$$y = \begin{cases} \alpha_0 + \alpha_1 x + \varepsilon & \text{if } x \le x_0, \\ \beta_0 + \beta_1 x + \varepsilon & \text{if } x > x_0. \end{cases}$$

   Show that if x_0 is known, this model can be rewritten as a linear regression model with a suitable choice of explanatory variables.

7. According to the above model, $E(y|x)$ may be discontinuous at x_0, unless the parameters of the model satisfy a certain restriction. Can you rewrite the above model with an equivalent set of fewer parameters, so that $E(y|x)$ is continuous for all x?

1. Minimization of Mean Squared Error

Problem Statement

We need to show that the function $g(X_1, X_2)$ that best approximates Y in terms of minimizing the mean squared error (MSE):

$$E\left[(Y - g(X_1, X_2))^2\right] \tag{1}$$

is given by the conditional expectation:

$$g(X_1, X_2) = E(Y|X_1, X_2). \tag{2}$$

Step 1: Define the Conditional Mean Squared Error

Consider the conditional mean squared error given X_1 and X_2:

$$E\left[(Y - g(X_1, X_2))^2 \mid X_1, X_2\right]. \tag{3}$$

This represents the expected squared difference between Y and $g(X_1, X_2)$, given the values of X_1 and X_2.

Step 2: Express the Conditional MSE Using Expectation

Expanding the squared term:

$$E\left[Y^2 - 2Y g(X_1, X_2) + g(X_1, X_2)^2 \mid X_1, X_2\right]. \tag{4}$$

Using the linearity of expectation:

$$E[Y^2 \mid X_1, X_2] - 2g(X_1, X_2)E[Y \mid X_1, X_2] + g(X_1, X_2)^2. \tag{5}$$

Let $\mu(X_1, X_2) = E[Y \mid X_1, X_2]$, so we rewrite this as:

$$E[Y^2 \mid X_1, X_2] - 2g(X_1, X_2)\mu(X_1, X_2) + g(X_1, X_2)^2. \tag{6}$$

Step 3: Minimize the Conditional MSE

To find the best function $g(X_1, X_2)$, we take the derivative with respect to $g(X_1, X_2)$:

$$\frac{d}{dg(X_1, X_2)} \left[E[Y^2 \mid X_1, X_2] - 2g(X_1, X_2)\mu(X_1, X_2) + g(X_1, X_2)^2 \right] = 0. \quad (7)$$

Computing the derivative:

$$-2\mu(X_1, X_2) + 2g(X_1, X_2) = 0. \quad (8)$$

Solving for $g(X_1, X_2)$:

$$g(X_1, X_2) = \mu(X_1, X_2) = E[Y \mid X_1, X_2]. \quad (9)$$

Step 4: Conclude for the Overall MSE

Since minimizing the conditional MSE at each (X_1, X_2) also minimizes the overall expectation:

$$E\left[(Y - g(X_1, X_2))^2\right] = E\left[E\left[(Y - g(X_1, X_2))^2 \mid X_1, X_2\right]\right], \quad (10)$$

it follows that the function that minimizes the unconditional mean squared error is:

$$g(X_1, X_2) = E[Y \mid X_1, X_2]. \quad (11)$$

Thus, the best approximation of Y in the least mean squared error sense is the conditional expectation $E[Y|X_1, X_2]$.

2. Optimal Linear Approximation of a Random Variable

1 Problem Statement

Suppose a random variable Y is approximated by a linear function of two random variables, X_1 and X_2, in the sense of minimizing the mean squared error:

$$E\left[(Y - (\beta_0 + \beta_1 X_1 + \beta_2 X_2))^2\right]. \tag{1}$$

All three random variables have zero mean, i.e.,

$$E[Y] = E[X_1] = E[X_2] = 0. \tag{2}$$

We determine the values of β_0, β_1, and β_2 that minimize the mean squared error.

2 Expanding the Mean Squared Error

Expanding the squared term:

$$MSE = E[Y^2] + E[(\beta_0 + \beta_1 X_1 + \beta_2 X_2)^2] - 2E[Y(\beta_0 + \beta_1 X_1 + \beta_2 X_2)]. \tag{3}$$

Since $E[Y] = E[X_1] = E[X_2] = 0$, we get $E[Y\beta_0] = E[X_1\beta_0] = E[X_2\beta_0] = 0$, eliminating β_0.

Expanding further:

$$\begin{aligned} MSE = {} & \mathrm{Var}(Y) + \beta_1^2 \mathrm{Var}(X_1) + \beta_2^2 \mathrm{Var}(X_2) + 2\beta_1\beta_2 \mathrm{Cov}(X_1, X_2) \\ & - 2\beta_1 \mathrm{Cov}(Y, X_1) - 2\beta_2 \mathrm{Cov}(Y, X_2). \end{aligned}$$

3 Minimizing the MSE

To find optimal values of β_1 and β_2, we take partial derivatives and set them to zero:

$$\frac{\partial MSE}{\partial \beta_1} = 2\beta_1 \text{Var}(X_1) + 2\beta_2 \text{Cov}(X_1, X_2) - 2\text{Cov}(Y, X_1) = 0, \tag{4}$$
$$\frac{\partial MSE}{\partial \beta_2} = 2\beta_2 \text{Var}(X_2) + 2\beta_1 \text{Cov}(X_1, X_2) - 2\text{Cov}(Y, X_2) = 0. \tag{5}$$

Dividing by 2:

$$\beta_1 \text{Var}(X_1) + \beta_2 \text{Cov}(X_1, X_2) = \text{Cov}(Y, X_1), \tag{6}$$
$$\beta_1 \text{Cov}(X_1, X_2) + \beta_2 \text{Var}(X_2) = \text{Cov}(Y, X_2). \tag{7}$$

4 Solving for β_1 and β_2

Rewriting in matrix form:

$$\begin{bmatrix} \text{Var}(X_1) & \text{Cov}(X_1, X_2) \\ \text{Cov}(X_1, X_2) & \text{Var}(X_2) \end{bmatrix} \begin{bmatrix} \beta_1 \\ \beta_2 \end{bmatrix} = \begin{bmatrix} \text{Cov}(Y, X_1) \\ \text{Cov}(Y, X_2) \end{bmatrix}. \tag{8}$$

Solving for β_1, β_2:

$$\begin{bmatrix} \beta_1 \\ \beta_2 \end{bmatrix} = \begin{bmatrix} \text{Var}(X_1) & \text{Cov}(X_1, X_2) \\ \text{Cov}(X_1, X_2) & \text{Var}(X_2) \end{bmatrix}^{-1} \begin{bmatrix} \text{Cov}(Y, X_1) \\ \text{Cov}(Y, X_2) \end{bmatrix}. \tag{9}$$

Since all variables have zero mean, we conclude:

$$\beta_0 = 0. \tag{10}$$

5 Final Answer

$$\beta_0 = 0,$$
$$\begin{bmatrix} \beta_1 \\ \beta_2 \end{bmatrix} = \begin{bmatrix} \text{Var}(X_1) & \text{Cov}(X_1, X_2) \\ \text{Cov}(X_1, X_2) & \text{Var}(X_2) \end{bmatrix}^{-1} \begin{bmatrix} \text{Cov}(Y, X_1) \\ \text{Cov}(Y, X_2) \end{bmatrix}.$$

This provides the best linear approximation of Y in terms of X_1 and X_2.

3. Optimal Linear Approximation of a Random Variable with Non-Zero Means

1 Problem Statement

Suppose a random variable Y is approximated by a linear function of two random variables, X_1 and X_2, in the sense of minimizing the mean squared error:

$$E\left[(Y - (\beta_0 + \beta_1 X_1 + \beta_2 X_2))^2\right]. \tag{1}$$

Unlike the previous case, the random variables now have non-zero means:

$$E[Y] = \mu_Y, \quad E[X_1] = \mu_{X_1}, \quad E[X_2] = \mu_{X_2}. \tag{2}$$

We determine the values of β_0, β_1, and β_2 that minimize the mean squared error.

2 Modified Approximation Model

Rewriting the random variables in terms of their deviations from the mean:

$$\begin{aligned}
\tilde{Y} &= Y - \mu_Y, \\
\tilde{X_1} &= X_1 - \mu_{X_1}, \\
\tilde{X_2} &= X_2 - \mu_{X_2}.
\end{aligned}$$

The approximation model remains:

$$Y \approx \beta_0 + \beta_1 X_1 + \beta_2 X_2. \tag{3}$$

Taking expectations on both sides:

$$\mu_Y = \beta_0 + \beta_1 \mu_{X_1} + \beta_2 \mu_{X_2}. \tag{4}$$

Thus, solving for β_0:

$$\beta_0 = \mu_Y - \beta_1 \mu_{X_1} - \beta_2 \mu_{X_2}. \tag{5}$$

3 Finding Optimal β_1 and β_2

The mean squared error to be minimized is:

$$E\left[(Y - (\beta_0 + \beta_1 X_1 + \beta_2 X_2))^2\right]. \tag{6}$$

Substituting β_0:

$$E\left[\left(\tilde{Y} - (\beta_1 \tilde{X}_1 + \beta_2 \tilde{X}_2)\right)^2\right]. \tag{7}$$

This is equivalent to the zero-mean case, where we solve:

$$\begin{bmatrix} \text{Var}(X_1) & \text{Cov}(X_1, X_2) \\ \text{Cov}(X_1, X_2) & \text{Var}(X_2) \end{bmatrix} \begin{bmatrix} \beta_1 \\ \beta_2 \end{bmatrix} = \begin{bmatrix} \text{Cov}(Y, X_1) \\ \text{Cov}(Y, X_2) \end{bmatrix}. \tag{8}$$

4 Final Answer: Effect of Non-Zero Means

$$\beta_0 = \mu_Y - \beta_1 \mu_{X_1} - \beta_2 \mu_{X_2},$$
$$\begin{bmatrix} \beta_1 \\ \beta_2 \end{bmatrix} = \begin{bmatrix} \text{Var}(\tilde{X}_1) & \text{Cov}(\tilde{X}_1, \tilde{X}_2) \\ \text{Cov}(\tilde{X}_1, \tilde{X}_2) & \text{Var}(\tilde{X}_2) \end{bmatrix}^{-1} \begin{bmatrix} \text{Cov}(\tilde{Y}, \tilde{X}_1) \\ \text{Cov}(\tilde{Y}, \tilde{X}_2) \end{bmatrix}.$$

Thus, the key difference is that β_0 is no longer necessarily zero and is adjusted based on the means of Y, X_1, and X_2, while β_1 and β_2 remain unchanged.

4. Transformation of Signal Detection Model into Linear Regression Form

Given Model

We consider the signal-plus-noise model:

$$y_t = a\cos(2\pi f t + \phi) + \varepsilon_t, \quad t = 1, 2, \ldots, n, \tag{1}$$

where:

- y_t is the observed signal at time t,
- $a\cos(2\pi f t + \phi)$ represents the signal component,
- ε_t represents noise,
- a is the amplitude,
- f (known) is the frequency,
- ϕ (unknown) is the phase.

Transformation Using Trigonometric Identity

Using the identity:

$$\cos(A + B) = \cos A \cos B - \sin A \sin B, \tag{2}$$

we expand the signal term:

$$a\cos(2\pi f t + \phi) = a\cos\phi\cos(2\pi f t) - a\sin\phi\sin(2\pi f t). \tag{3}$$

Defining new parameters:

$$\beta_1 = a\cos\phi, \quad \beta_2 = -a\sin\phi, \tag{4}$$

we rewrite the model as:

$$y_t = \beta_1 X_{1t} + \beta_2 X_{2t} + \varepsilon_t, \tag{5}$$

where:

$$X_{1t} = \cos(2\pi f t), \quad X_{2t} = \sin(2\pi f t). \tag{6}$$

Interpreting the Linear Model

In the transformed model:

- y_t is the dependent variable (response variable),
- X_{1t} and X_{2t} are the explanatory variables (predictors),
- β_1 and β_2 are unknown but constant regression coefficients,
- ε_t is the error term.

Since X_{1t} and X_{2t} are known functions of t, we can estimate β_1 and β_2 using standard linear regression techniques.

5. Sources of Model Error in Salary Prediction

1 Introduction

The salary of an employee is modeled using the equation:

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3 + \beta_4 x_4 + \varepsilon$$

where:

- x_1 = High school graduation (1 = Yes, 0 = No),
- x_2 = College graduation (1 = Yes, 0 = No),
- x_3 = At least one postgraduate degree (1 = Yes, 0 = No),
- x_4 = Years of service,
- ε = Error term capturing missing influences.

Even though this equation is useful, it does not capture all real-life salary variations. Below are the major sources of model error.

2 Sources of Model Error

2.1 1. Missing Important Factors (Omitted Variables Bias)

Layman's Explanation: The model only considers education and experience, but salary also depends on:

- **Industry type:** IT, healthcare, and teaching have different salaries.
- **Job role:** A manager earns more than an entry-level worker.
- **Location:** Salaries are higher in urban areas.
- **Skills and performance:** A skilled worker earns more than an average one.

Sophisticated Term: The model suffers from *omitted variable bias*, leading to incorrect estimates of coefficients.

2.2 2. Mistakes in Data (Measurement Errors)

Layman's Explanation: If employee records contain errors:

- Some may falsely report education levels.
- Years of service may include unpaid internships or job gaps.

Sophisticated Term: *Measurement error* leads to biased or inconsistent coefficient estimates.

2.3 3. Oversimplified Salary Growth (Model Misspecification)

Layman's Explanation: The model assumes salary increases linearly with experience, but in reality:

- Salary may grow fast at first, then slow down over time.

Sophisticated Term: The model assumes a *linear relationship*, while real-life salary growth may be *non-linear*.

2.4 4. Unequal Salary Variations (Heteroskedasticity)

Layman's Explanation: Even employees with the same experience can have different salaries due to:

- Performance-based bonuses.
- Company policies.

Sophisticated Term: *Heteroskedasticity* occurs when salary variance is not constant across observations.

2.5 5. Hidden Factors Influencing Both Education & Salary (Endogeneity)

Layman's Explanation: Highly intelligent people:

- Are more likely to pursue higher education.
- Often earn higher salaries.

The model wrongly assumes that education *causes* higher salaries, ignoring intelligence as an underlying factor. **Sophisticated Term:** *Endogeneity* arises when an independent variable (education) is correlated with unobserved factors (intelligence).

2.6 6. Random Factors in Life (Stochastic Variability)

Layman's Explanation: Even with all possible factors, salaries can be unpredictable due to:

- Salary negotiations.
- Unexpected promotions or layoffs.

Sophisticated Term: *Stochastic variability* accounts for randomness in salary determination.

2.7 7. Missing Interaction Effects

Layman's Explanation: The model assumes each factor affects salary separately, but:

- A postgraduate degree may only increase salary after 5+ years of experience.

Sophisticated Term: *Interaction effects* exist when the impact of one variable depends on another.

2.8 8. Salary Changes Over Time (Time Effects)

Layman's Explanation: Salaries change due to:

- Inflation (gradual salary increases).
- Economic recessions (salary stagnation or cuts).

Sophisticated Term: The model lacks a *temporal component*, failing to capture inflationary trends.

2.9 9. Differences Between Companies (Firm-Specific Effects)

Layman's Explanation: Some companies pay higher salaries even for the same role:

- Google pays more than a startup.
- Public sector vs. private sector pay differences.

Sophisticated Term: *Firm-specific heterogeneity* affects salary determination.

2.10 10. Social & Demographic Factors

Layman's Explanation: Salary disparities may exist due to:

- Gender, caste, or racial discrimination.
- Regional differences in salary structures.

Sophisticated Term: *Social biases and demographic effects* introduce structural disparities in salary predictions.

3 Conclusion

The error term (ε) captures these missing effects, making salary predictions imperfect. Addressing these errors requires:

- Adding more variables (e.g., industry, job role, location).
- Adjusting for interaction effects.
- Accounting for inflation and company-level differences.

Even the best model cannot fully eliminate randomness, but improvements can reduce errors significantly.

6. Rewriting a Piecewise Linear Regression Model as a Linear Model

Problem Statement

Consider the piecewise linear (broken line) regression model:

$$y = \begin{cases} \alpha_0 + \alpha_1 x + \varepsilon, & \text{if } x \leq x_0 \\ \beta_0 + \beta_1 x + \varepsilon, & \text{if } x > x_0 \end{cases}$$

where:

- y is the dependent variable.
- x is the independent variable.
- x_0 is a known threshold (or breakpoint).
- $\alpha_0, \alpha_1, \beta_0, \beta_1$ are unknown regression coefficients.
- ε is the error term.

Introducing an Indicator Variable

Define an indicator function $I(x > x_0)$ as:

$$I(x > x_0) = \begin{cases} 0, & \text{if } x \leq x_0 \\ 1, & \text{if } x > x_0 \end{cases}$$

This function allows us to model different intercepts and slopes while keeping the equation in a single form.

Rewriting the Model

For $x \leq x_0$, $I(x > x_0) = 0$, so the model simplifies to:

$$y = \alpha_0 + \alpha_1 x + \varepsilon.$$

For $x > x_0$, $I(x > x_0) = 1$, so the model becomes:

$$y = \beta_0 + \beta_1 x + \varepsilon.$$

Expressing β_0 and β_1 in terms of α_0 and α_1:

$$\beta_0 = \alpha_0 + \gamma_0, \quad \beta_1 = \alpha_1 + \gamma_1,$$

where $\gamma_0 = \beta_0 - \alpha_0$ and $\gamma_1 = \beta_1 - \alpha_1$ represent the jumps in the intercept and slope at x_0.

Thus, we rewrite y as:

$$y = \alpha_0 + \alpha_1 x + \gamma_0 I(x > x_0) + \gamma_1 x I(x > x_0) + \varepsilon.$$

Conclusion

The transformed equation is a standard multiple linear regression model with explanatory variables:

- x
- $I(x > x_0)$
- $xI(x > x_0)$

where the regression coefficients are:

- α_0 (intercept)
- α_1 (slope before x_0)
- γ_0 (intercept shift at x_0)
- γ_1 (slope shift at x_0)

Thus, the piecewise regression model is rewritten as a linear regression model using an indicator variable.

7. Ensuring Continuity in a Piecewise Linear Regression Model

Rewriting the Model to Ensure Continuity

We consider the piecewise linear regression model:

$$y = \begin{cases} \alpha_0 + \alpha_1 x + \varepsilon, & x \leq x_0, \\ \beta_0 + \beta_1 x + \varepsilon, & x > x_0. \end{cases} \tag{1}$$

Taking the expectation on both sides, we obtain:

$$E(Y|X) = \begin{cases} \alpha_0 + \alpha_1 x, & x \leq x_0, \\ \beta_0 + \beta_1 x, & x > x_0. \end{cases} \tag{2}$$

For continuity at x_0, we require:

$$\lim_{x \to x_0^-} E(Y|X) = \lim_{x \to x_0^+} E(Y|X). \tag{3}$$

Substituting from our piecewise model:

$$\alpha_0 + \alpha_1 x_0 = \beta_0 + \beta_1 x_0. \tag{4}$$

Rearranging:

$$\beta_0 - \alpha_0 = (\alpha_1 - \beta_1) x_0. \tag{5}$$

Define $\gamma_0 = \beta_0 - \alpha_0$ and $\gamma_1 = \beta_1 - \alpha_1$, so that:

$$\gamma_0 = -\gamma_1 x_0. \tag{6}$$

Substituting this into our previous transformed regression model:

$$y = \alpha_0 + \alpha_1 x - \gamma_1 x_0 I(x > x_0) + \gamma_1 x I(x > x_0) + \varepsilon. \tag{7}$$

Factorizing γ_1:

$$y = \alpha_0 + \alpha_1 x + \gamma_1 (x - x_0) I(x > x_0) + \varepsilon. \tag{8}$$

Since this formulation ensures continuity at x_0, the final equivalent regression model with fewer parameters is:

$$E(Y|X) = \alpha_0 + \alpha_1 x + \gamma_1 (x - x_0) I(x > x_0). \tag{9}$$

This new form eliminates the discontinuity and preserves the piecewise nature of the regression model.

Statistical Methods II: Week 6 Assignment (due 17 February, 2025)

1. The table given below gives men's and women's world record times for various outdoor running distances, recognized by the International Association of Athletics Federations (IAAF) as of 17 November, 2017. It may be assumed that the log of the record time is approximately a linear function of the log of the running distance.

Running distance (meters)	Men's record (seconds)	Women's record (seconds)
100	9.58	10.49
200	19.19	21.34
400	43.03	47.60
800	100.91	113.28
1000	131.96	148.98
1500	206.00	230.07
2000	284.79	323.75
3000	440.67	486.11
5000	757.35	851.15
10000	1577.53	1757.45

   If a linear model of the form $y = X\beta + \epsilon$ is used for the men's log-record times and another one for the women's log-record times, identify the matrix and vectors used in the model.

2. Continuing with the above data, construct a single 'grand model' with four parameters which can be used as a substitute for these two models, and identify the corresponding matrix and vectors.

3. Suppose a random variable Y has to be approximated by a linear function $\beta_0+\beta_1X_1+\beta_2X_2+\cdots+\beta_kX_k$ of k random variables, $X_1, X_2, \ldots, X_k$. All the random variables have zero mean. It is claimed that the best approximation, in the sense of the smallest mean squared error of approximation $E[\{Y - (\beta_0 + \beta_1X_1 + \beta_2X_2 + \cdots + \beta_kX_k)\}^2]$, is achieved only when $\beta_0 = 0$, irrespective of the choices of the coefficients β_1, β_2 $\ldots$, β_k.

4. In the above set-up, show that the quantity $E[\{Y - (\beta_1X_1 + \beta_2X_2 + \cdots + \beta_kX_k)\}^2]$ can be written as $var((Y - \beta^{*T}X) + (\beta^* - \beta)^TX)$, where

$$\beta = \begin{pmatrix} \beta_1 \\ \beta_2 \\ \vdots \\ \beta_k \end{pmatrix}, \quad X = \begin{pmatrix} X_1 \\ X_2 \\ \vdots \\ X_k \end{pmatrix}$$

   and β^* is any particular value of β.

5. Show that, in the above set-up, $cov(Y, \beta^TX) = \beta^T\Sigma_{xy}$ and $var(\beta^TX) = \beta^T\Sigma_{xx}\beta$, where

$$\Sigma_{xx} = \begin{pmatrix} var(X_1) & cov(X_1, X_2) & \cdots & cov(X_1, X_k) \\ cov(X_2, X_1) & var(X_2) & \cdots & cov(X_2, X_k) \\ \vdots & \vdots & \ddots & \vdots \\ cov(X_k, X_1) & cov(X_k, X_2) & \cdots & var(X_k) \end{pmatrix}, \quad \Sigma_{xy} = \begin{pmatrix} cov(X_1, Y) \\ cov(X_2, Y) \\ \vdots \\ cov(X_k, Y) \end{pmatrix}.$$

6. Show that, in the above set-up, the covariance between $Y - \beta^{*T} X$ and $(\beta^* - \beta)^T X$ is zero if $\beta^* = \Sigma_{xx}^{-1} \Sigma_{xy}$.

7. Show that, in the above set-up, the quantity $E[\{Y - (\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_k X_k)\}^2]$ is minimized when the vector β is chosen as β^*.

8. How does the answer to the above question change when the random variables have non-zero mean?

1. Linear Model for World Record Running Times

Problem Statement

The table below gives men's and women's world record times for various outdoor running distances, recognized by the International Association of Athletics Federations (IAAF) as of 17 November, 2017. It is assumed that the log of the record time is approximately a linear function of the log of the running distance.

Running Distance (m)	**Men's Record (s)**	**Women's Record (s)**
100	9.58	10.49
200	19.19	21.34
400	43.03	47.60
800	100.91	113.28
1000	131.96	148.98
1500	206.00	230.07
2000	284.79	323.75
3000	440.67	486.11
5000	757.35	851.15
10000	1577.53	1757.45

Table 1: World Record Running Times (IAAF, as of 17 Nov 2017)

If a linear model of the form

$$y = X\beta + \epsilon$$

is used for the men's log-record times and another one for the women's log-record times, identify the matrix and vectors used in the model.

Linear Model Formulation

We assume that the relationship follows the model:

$$\log(\text{Time}) = \beta_0 + \beta_1 \log(\text{Distance}) + \epsilon$$

where:

- y is the vector of log-record times,
- X is the design matrix containing predictor values,
- β is the coefficient vector,
- ϵ represents the error term.

Matrix Representation

For both men and women, the linear model can be written as:

$$y = X\beta + \epsilon$$

where:

$$X = \begin{bmatrix} 1 & \log(100) \\ 1 & \log(200) \\ 1 & \log(400) \\ 1 & \log(800) \\ 1 & \log(1000) \\ 1 & \log(1500) \\ 1 & \log(2000) \\ 1 & \log(3000) \\ 1 & \log(5000) \\ 1 & \log(10000) \end{bmatrix}$$

$$y_m = \begin{bmatrix} \log(9.58) \\ \log(19.19) \\ \log(43.03) \\ \log(100.91) \\ \log(131.96) \\ \log(206.00) \\ \log(284.79) \\ \log(440.67) \\ \log(757.35) \\ \log(1577.53) \end{bmatrix}, \quad y_w = \begin{bmatrix} \log(10.49) \\ \log(21.34) \\ \log(47.60) \\ \log(113.28) \\ \log(148.98) \\ \log(230.07) \\ \log(323.75) \\ \log(486.11) \\ \log(851.15) \\ \log(1757.45) \end{bmatrix}$$

The coefficient vector β is given by:

$$\beta = \begin{bmatrix} \beta_0 \\ \beta_1 \end{bmatrix}$$

Estimation of Parameters

The optimal estimate for β using Ordinary Least Squares (OLS) is given by:

$$\beta = (X^T X)^{-1} X^T y$$

where:

- $X^T X$ is a 2×2 matrix summarizing the predictor variable variation,
- $X^T y$ is a 2×1 matrix capturing the relationship between predictors and response,
- $(X^T X)^{-1}$ ensures a unique and optimal solution.

Conclusion

This linear regression model provides insight into how world record times scale with distance in a log-log relationship. The coefficient β_1 determines how log-time changes with log-distance, indicating the rate of increase in time required for longer distances. The intercept β_0 accounts for baseline differences in record times.

This analysis is useful for predicting potential future record times or identifying trends in athletic performance over time. Further studies can explore how external factors such as track conditions, altitude, and training methods affect this model.

2. Formulation of a Grand Model with Four Parameters

Problem Statement

Continuing with the data from the previous problem, construct a single *grand model* with four parameters, which can be used as a substitute for the two separate models for men's and women's record times. Identify the corresponding matrix and vectors.

Solution

To unify the models for both men's and women's record times, we assume a linear relationship between the logarithm of record time and the logarithm of distance:

$$\log(T) = \beta_0 + \beta_1 \log(D) + \beta_2 I_{\text{Women}} + \beta_3 I_{\text{Women}} \cdot \log(D) + \epsilon \tag{1}$$

where:

- T is the record time.
- D is the running distance.
- I_{Women} is an indicator variable:

$$I_{\text{Women}} = \begin{cases} 0, & \text{if male} \\ 1, & \text{if female} \end{cases}$$

- β_0 represents the intercept for men.
- β_1 represents the slope for men.
- β_2 represents the difference in intercept between women and men.
- β_3 represents the difference in slope between women and men.
- ϵ is the error term.

Design Matrix and Vectors

Using the provided data, the design matrix X is:

$$X = \begin{bmatrix} 1 & \log(100) & 0 & 0 \\ 1 & \log(200) & 0 & 0 \\ 1 & \log(400) & 0 & 0 \\ 1 & \log(800) & 0 & 0 \\ 1 & \log(1000) & 0 & 0 \\ 1 & \log(1500) & 0 & 0 \\ 1 & \log(2000) & 0 & 0 \\ 1 & \log(3000) & 0 & 0 \\ 1 & \log(5000) & 0 & 0 \\ 1 & \log(10000) & 0 & 0 \\ 1 & \log(100) & 1 & \log(100) \\ 1 & \log(200) & 1 & \log(200) \\ 1 & \log(400) & 1 & \log(400) \\ 1 & \log(800) & 1 & \log(800) \\ 1 & \log(1000) & 1 & \log(1000) \\ 1 & \log(1500) & 1 & \log(1500) \\ 1 & \log(2000) & 1 & \log(2000) \\ 1 & \log(3000) & 1 & \log(3000) \\ 1 & \log(5000) & 1 & \log(5000) \\ 1 & \log(10000) & 1 & \log(10000) \end{bmatrix}$$

The response vector y is:

$$y = \begin{bmatrix} \log(9.58) \\ \log(19.19) \\ \log(43.03) \\ \log(100.91) \\ \log(131.96) \\ \log(206.00) \\ \log(284.79) \\ \log(440.67) \\ \log(757.35) \\ \log(1577.53) \\ \log(10.49) \\ \log(21.34) \\ \log(47.60) \\ \log(113.28) \\ \log(148.98) \\ \log(230.07) \\ \log(323.75) \\ \log(486.11) \\ \log(851.15) \\ \log(1757.45) \end{bmatrix}$$

The parameter vector β to be estimated is:

$$\beta = \begin{bmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \\ \beta_3 \end{bmatrix}$$

Ordinary Least Squares (OLS) Estimation

The best estimate of β is given by:

$$\beta = (X^T X)^{-1} X^T y$$

where:

- $X^T X$ is a 4×4 matrix summarizing the structure of predictor variables.
- $X^T y$ is a 4×1 matrix capturing the relationship between predictors and response.
- $(X^T X)^{-1}$ ensures an optimal, unique solution.

Final Interpretation

This model allows:

1. A general relationship between distance and record time.
2. A direct comparison between men's and women's performance.
3. Different intercepts and slopes for each gender.

The final equation is:

$$\log(T) = \beta_0 + \beta_1 \log(D) + \beta_2 I_{\text{Women}} + \beta_3 I_{\text{Women}} \cdot \log(D) + \epsilon$$

where:

- If $I_{\text{Women}} = 0$ (men's model), the equation reduces to:

$$\log(T) = \beta_0 + \beta_1 \log(D) + \epsilon.$$

- If $I_{\text{Women}} = 1$ (women's model), the equation becomes:

$$\log(T) = (\beta_0 + \beta_2) + (\beta_1 + \beta_3) \log(D) + \epsilon.$$

Thus, the four-parameter model effectively captures both men's and women's record times in a single equation.

3. Minimizing Mean Squared Error in Linear Approximation

Problem Statement

Suppose a random variable Y has to be approximated by a linear function:

$$\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_k X_k$$

of k random variables, $X_1, X_2, \ldots, X_k$. All the random variables have zero mean.

It is claimed that the best approximation, in the sense of the smallest mean squared error of approximation,

$$E\left[(Y - (\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_k X_k))^2\right],$$

is achieved only when $\beta_0 = 0$, irrespective of the choices of the coefficients $\beta_1, \beta_2, \ldots, \beta_k$. Prove this claim.

Solution

The given approximation is:

$$\hat{Y} = \beta_0 + \sum_{i=1}^{k} \beta_i X_i.$$

The mean squared error (MSE) is given by:

$$E\left[(Y - \hat{Y})^2\right] = E\left[\left(Y - (\beta_0 + \sum_{i=1}^{k} \beta_i X_i)\right)^2\right].$$

Expanding the squared term,

$$E\left[Y^2 - 2Y(\beta_0 + \sum_{i=1}^{k} \beta_i X_i) + (\beta_0 + \sum_{i=1}^{k} \beta_i X_i)^2\right].$$

Using the linearity of expectation,

$$E[Y^2] - 2\beta_0 E[Y] - 2\sum_{i=1}^{k} \beta_i E[Y X_i] + \beta_0^2 + 2\beta_0 \sum_{i=1}^{k} \beta_i E[X_i] + E\left[\left(\sum_{i=1}^{k} \beta_i X_i\right)^2\right].$$

Since we are given that $E[Y] = 0$ and $E[X_i] = 0$ for all i, these terms vanish, leading to:

$$E[Y^2] - 2\sum_{i=1}^{k} \beta_i E[Y X_i] + \beta_0^2 + E\left[\left(\sum_{i=1}^{k} \beta_i X_i\right)^2\right].$$

Since $\beta_0^2 \geq 0$, it contributes a non-negative term to the error. To minimize MSE, we must set $\beta_0 = 0$, which eliminates this extra term.

Conclusion

Thus, the best approximation minimizing the mean squared error is obtained when $\beta_0 = 0$, proving the claim.

4. Mean Squared Error Representation

Problem Statement

In the given setup, show that the quantity:

$$E\left[(Y-(\beta_1X_1+\beta_2X_2+\cdots+\beta_kX_k))^2\right]$$

can be written as:

$$\text{Var}\left((Y-\beta^{*T}X)+(\beta^*-\beta)^TX\right),$$

where:

$$\beta=\begin{bmatrix}\beta_1\\ \beta_2\\ \vdots\\ \beta_k\end{bmatrix},\quad X=\begin{bmatrix}X_1\\ X_2\\ \vdots\\ X_k\end{bmatrix}$$

and β^* is any particular value of β.

Solution

Step 1: Understanding What We Need to Prove: we need to show that:

$$E\left[(Y-\beta^TX)^2\right]=\text{Var}\left((Y-\beta^{*T}X)+(\beta^*-\beta)^TX\right).$$

where: - β^* is an arbitrary fixed choice of β, - X is the vector of predictor variables, - β is the coefficient vector.

Step 2: Decomposing $Y-\beta^TX$ Consider the error term:

$$Y-\beta^TX.$$

We introduce β^* and add/subtract $\beta^{*T}X$:

$$Y-\beta^TX=(Y-\beta^{*T}X)+(\beta^{*T}X-\beta^TX).$$

Since $\beta^{*T}X-\beta^TX$ simplifies to $(\beta^*-\beta)^TX$, we can rewrite:

$$Y-\beta^TX=(Y-\beta^{*T}X)+(\beta^*-\beta)^TX.$$

Thus, the squared error becomes:

$$E\left[(Y-\beta^TX)^2\right]=E\left[\left((Y-\beta^{*T}X)+(\beta^*-\beta)^TX\right)^2\right].$$

Step 3: Expanding $E[(A+B)^2]$ Using the identity $(a+b)^2=a^2+2ab+b^2$, we get:

$$E[(A+B)^2]=E\left[(Y-\beta^{*T}X)^2+2(Y-\beta^{*T}X)(\beta^*-\beta)^TX+((\beta^*-\beta)^TX)^2\right].$$

By linearity of expectation,

$$E[(A+B)^2]=E[(Y-\beta^{*T}X)^2]+2E[(Y-\beta^{*T}X)(\beta^*-\beta)^TX]+E[((\beta^*-\beta)^TX)^2].$$

Step 4: Expanding $(E[A+B])^2$

$$(E[A+B])^2 = \left(E[Y - \beta^{*T}X + (\beta^* - \beta)^T X]\right)^2.$$

Since we assumed all variables have zero mean,

$$E[Y - \beta^{*T}X] = 0, \quad E[X] = 0.$$

Thus,

$$E[A+B] = E[Y - \beta^{*T}X] + E[(\beta^* - \beta)^T X] = 0 + 0 = 0.$$

So,

$$(E[A+B])^2 = 0.$$

Step 5: Substituting into the Variance Formula Using:

$$\mathrm{Var}(A+B) = E[(A+B)^2] - (E[A+B])^2.$$

Substituting our results,

$$\mathrm{Var}(A+B) = E[(Y - \beta^{*T}X)^2] + 2E[(Y - \beta^{*T}X)(\beta^* - \beta)^T X] + E[((\beta^* - \beta)^T X)^2] - 0.$$

Now, assuming β^* is the optimal choice that minimizes MSE, the cross-term $E[(Y - \beta^{*T}X)(\beta^* - \beta)^T X]$ vanishes:

$$E[(Y - \beta^{*T}X)(\beta^* - \beta)^T X] = 0.$$

Thus,

$$\mathrm{Var}(A+B) = E[(Y - \beta^{*T}X)^2] + E[((\beta^* - \beta)^T X)^2].$$

Since $E[(Y - \beta^T X)^2] = \mathrm{Var}(A+B)$, we get:

$$E[(Y - \beta^T X)^2] = \mathrm{Var}((Y - \beta^{*T}X) + (\beta^* - \beta)^T X).$$

Conclusion

This result shows that the mean squared error consists of: 1. The variance of the error when using β^*. 2. An additional variance term due to the deviation of β from β^*.

Thus, we have successfully shown the required result.

5. Covariance and Variance of the Linear Combination of Random Variables

Problem Statement

Show that, in the given setup, the following hold:

$$\mathrm{Cov}(Y, \beta^T X) = \beta^T \Sigma_{xy}, \quad \mathrm{Var}(\beta^T X) = \beta^T \Sigma_{xx} \beta,$$

where:

$$\Sigma_{xx} = \begin{bmatrix} \mathrm{var}(X_1) & \mathrm{cov}(X_1, X_2) & \dots & \mathrm{cov}(X_1, X_k) \\ \mathrm{cov}(X_2, X_1) & \mathrm{var}(X_2) & \dots & \mathrm{cov}(X_2, X_k) \\ \vdots & \vdots & \ddots & \vdots \\ \mathrm{cov}(X_k, X_1) & \mathrm{cov}(X_k, X_2) & \dots & \mathrm{var}(X_k) \end{bmatrix},$$

$$\Sigma_{xy} = \begin{bmatrix} \mathrm{cov}(X_1, Y) \\ \mathrm{cov}(X_2, Y) \\ \vdots \\ \mathrm{cov}(X_k, Y) \end{bmatrix}.$$

Solution

Step 1: Covariance of Y and $\beta^T X$

We first compute:

$$\mathrm{Cov}(Y, \beta^T X) = E[(Y - E[Y])(\beta^T X - E[\beta^T X])].$$

Since expectation is linear, we can factor out β^T:

$$\mathrm{Cov}(Y, \beta^T X) = \beta^T E[(Y - E[Y])(X - E[X])].$$

The expectation inside the expression is simply the covariance matrix Σ_{xy}, thus:

$$\mathrm{Cov}(Y, \beta^T X) = \beta^T \Sigma_{xy}.$$

Step 2: Variance of $\beta^T X$

Now, we compute:

$$\mathrm{Var}(\beta^T X) = E[(\beta^T X - E[\beta^T X])^2].$$

Using linearity of expectation:

$$E[\beta^T X] = \beta^T E[X].$$

So the variance simplifies to:

$$\mathrm{Var}(\beta^T X) = E[(\beta^T (X - E[X]))^2].$$

Since β^T is a constant factor:

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

Recognizing that the expectation term inside is simply the covariance matrix Σ_{xx}, we get:

$$\text{Var}(\beta^T X) = \beta^T \Sigma_{xx} \beta.$$

Visualization and Interpretation

- **Covariance Interpretation:** The covariance $\text{Cov}(Y, \beta^T X)$ measures the linear relationship between Y and the linear combination $\beta^T X$. The result shows that it is given by a weighted sum of individual covariances between Y and each X_i.
- **Variance Interpretation:** The variance $\text{Var}(\beta^T X)$ measures the spread of the linear combination of predictors. The result shows that it is given by the quadratic form $\beta^T \Sigma_{xx} \beta$, which reflects how the variance of each predictor and their covariances contribute to the overall spread.
- **Geometric Insight:** In high-dimensional space, β defines a direction, and these formulas describe how covariance and variance behave when data is projected onto that direction.

Conclusion

We have successfully derived and interpreted the required expressions:

$$\text{Cov}(Y, \beta^T X) = \beta^T \Sigma_{xy}, \quad \text{Var}(\beta^T X) = \beta^T \Sigma_{xx} \beta.$$

6. Covariance Between $Y - \beta^{*T}X$ and $(\beta^* - \beta)^T X$

Problem Statement

Show that, in the given setup, the covariance between $Y - \beta^{*T}X$ and $(\beta^* - \beta)^T X$ is zero if

$$\beta^* = \Sigma_{xx}^{-1}\Sigma_{xy}.$$

where:

- Y is a random variable.
- X is a vector of random variables $(X_1, X_2, \ldots, X_k)^T$.
- β and β^* are k-dimensional coefficient vectors.
- Σ_{xx} is the covariance matrix of X.
- Σ_{xy} is the covariance vector between X and Y.

Solution

We need to evaluate:

$$\text{Cov}\left(Y - \beta^{*T}X, (\beta^* - \beta)^T X\right).$$

Using the linearity of covariance, we expand:

$$\text{Cov}\left(Y, (\beta^* - \beta)^T X\right) - \text{Cov}\left(\beta^{*T}X, (\beta^* - \beta)^T X\right).$$

Step 1: Expanding Covariances
Using the linearity property:

$$\text{Cov}(Y, (\beta^* - \beta)^T X) = (\beta^* - \beta)^T \text{Cov}(X, Y).$$

Similarly,

$$\text{Cov}(\beta^{*T}X, (\beta^* - \beta)^T X) = \beta^{*T}\text{Cov}(X, (\beta^* - \beta)^T X).$$

Since

$$\text{Cov}(X, a^T X) = \Sigma_{xx} a,$$

we obtain:

$$\text{Cov}(\beta^{*T}X, (\beta^* - \beta)^T X) = \beta^{*T}\Sigma_{xx}(\beta^* - \beta).$$

Step 2: Substituting β^*
By assumption,

$$\beta^* = \Sigma_{xx}^{-1}\Sigma_{xy}.$$

Thus,

$$\text{Cov}(X, Y) = \Sigma_{xy}.$$

So,

$$\text{Cov}(Y, (\beta^* - \beta)^T X) = (\beta^* - \beta)^T \Sigma_{xy}.$$

Similarly,

$$\text{Cov}(\beta^{*T} X, (\beta^* - \beta)^T X) = \beta^{*T} \Sigma_{xx} (\beta^* - \beta).$$

Substituting $\beta^* = \Sigma_{xx}^{-1} \Sigma_{xy}$, we get:

$$\beta^{*T} \Sigma_{xx} = \Sigma_{xy}^T.$$

Thus,

$$\beta^{*T} \Sigma_{xx} (\beta^* - \beta) = \Sigma_{xy}^T (\beta^* - \beta).$$

Since both terms are equal,

$$\text{Cov}(Y, (\beta^* - \beta)^T X) = \text{Cov}(\beta^{*T} X, (\beta^* - \beta)^T X).$$

This implies:

$$\text{Cov}\left(Y - \beta^{*T} X, (\beta^* - \beta)^T X\right) = 0.$$

Conclusion

Thus, we have proved that the covariance between $Y - \beta^{*T} X$ and $(\beta^* - \beta)^T X$ is **zero** when $\beta^* = \Sigma_{xx}^{-1} \Sigma_{xy}$. This result is significant in regression analysis as it implies that the residual $Y - \beta^{*T} X$ is uncorrelated with the predictors X when using the optimal coefficient β^*.

7. Minimization of the Expected Squared Error

Problem Statement

Show that, in the given setup, the quantity

$$E\left[(Y - (\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_k X_k))^2\right]$$

is minimized when the vector β is chosen as β^*, where

$$\beta^* = \Sigma_{xx}^{-1}\Sigma_{xy}.$$

Solution

Step 1: Expressing the Mean Squared Error (MSE)

From our previous results (Question 4), we have established that:

$$E\left[(Y - \beta^T X)^2\right] = \text{Var}((Y - \beta^{*T} X) + (\beta^* - \beta)^T X).$$

Using the variance expansion property:

$$\text{Var}(A + B) = \text{Var}(A) + \text{Var}(B) + 2\text{Cov}(A, B),$$

we apply this to our case:

$$\text{Var}((Y-\beta^{*T}X)+(\beta^*-\beta)^T X) = \text{Var}(Y-\beta^{*T}X)+\text{Var}((\beta^*-\beta)^T X)+2\text{Cov}(Y-\beta^{*T}X, (\beta^*-\beta)^T X).$$

Step 2: Using the Covariance Result

From our earlier derivation(Question 6), we established that:

$$\text{Cov}(Y - \beta^{*T} X, (\beta^* - \beta)^T X) = 0.$$

Thus, the above expression simplifies to:

$$E\left[(Y - \beta^T X)^2\right] = \text{Var}(Y - \beta^{*T} X) + \text{Var}((\beta^* - \beta)^T X).$$

Since variance is always non-negative, we conclude that:

$$E\left[(Y - \beta^T X)^2\right] \geq \text{Var}(Y - \beta^{*T} X).$$

Equality holds when:

$$\text{Var}((\beta^* - \beta)^T X) = 0,$$

which happens only if:

$$\beta = \beta^*.$$

Conclusion

Thus, the mean squared error $E[(Y - \beta^T X)^2]$ is minimized when:

$$\beta = \beta^* = \Sigma_{xx}^{-1}\Sigma_{xy}.$$

8. Effect of Nonzero Mean on the Optimal Regression Estimator

Problem Statement

How does the answer to the above question change when the random variables have non-zero mean?

Solution

Step 1: Recall the Result for Zero-Mean Variables

When X and Y are zero-mean, the least squares minimization problem:

$$E[(Y - \beta^T X)^2]$$

leads to the variance decomposition:

$$E[(\tilde{Y} - \beta^T \tilde{X})^2] = \text{Var}(\tilde{Y}) - 2\beta^T \text{Cov}(\tilde{X}, \tilde{Y}) + \beta^T \text{Var}(\tilde{X})\beta.$$

By differentiating and setting the gradient to zero, we obtain the optimal coefficient vector:

$$\beta^* = \Sigma_{XX}^{-1} \Sigma_{XY},$$

where:
- $\Sigma_{XX} = \text{Var}(X)$ is the covariance matrix of X, - $\Sigma_{XY} = \text{Cov}(X, Y)$ is the covariance vector between X and Y.

Step 2: Adjusting for Nonzero Means

Now, let X and Y have nonzero means:

$$\mu_X = E[X], \quad \mu_Y = E[Y].$$

The regression model is given by:

$$Y = \beta^T X + \epsilon.$$

Taking expectations on both sides:

$$E[Y] = \beta^T E[X] + E[\epsilon].$$

Since $E[\epsilon] = 0$, we obtain:

$$\mu_Y = \beta^T \mu_X.$$

This implies that the regression equation must include an intercept term:

$$\beta_0 = \mu_Y - \beta^T \mu_X.$$

Step 3: The Final Adjusted Model

When the random variables have nonzero mean, the optimal regression model takes the form:

$$Y = \beta_0 + \beta^T X + \epsilon,$$

where:

1. The coefficient vector β^* remains:

$$\beta^* = \Sigma_{XX}^{-1} \Sigma_{XY}.$$

2. The intercept term β_0 is introduced as:

$$\beta_0 = \mu_Y - (\beta^*)^T \mu_X.$$

Conclusion

The key difference from the zero-mean case is the introduction of an intercept term β_0, which accounts for the nonzero means of X and Y. However, the core result for β^* remains unchanged.