R Programming Basics for Statistics

Mahesh Divakaran

Amity University, Lucknow

2024-10-05

What is R?

R is an open-source environment for statistical computing and visualisa- tion. It is based on the S language developed at Bell Laboratories in the 1980's [20], and is the product of an active movement among statisti- cians for a powerful, programmable, portable, and open computing en- vironment, applicable to the most complex and sophsticated problems, as well as "routine" analysis, without any restrictions on access or use.

What is R?

Here is a description from the R Project home page:

“R is an integrated suite of software facilities for data manip- ulation, calculation and graphical display. It includes:

- an effective data handling and storage facility,
- a suite of operators for calculations on arrays, in partic- ular matrices,
- a large, coherent, integrated collection of intermediate tools for data analysis,
- graphical facilities for data analysis and display either on- screen or on hardcopy, and
- a well-developed, simple and effective programming lan- guage which includes conditionals, loops, user-defined re- cursive functions and input and output facilities.”

Advantages of R

- **Open Source & Free**
  R is an open-source programming language available free of cost, which encourages wide community support and rapid updates.
- **Comprehensive Statistical Analysis**
  Provides a rich ecosystem of libraries and packages for statistical analysis, data visualization, and machine learning.
- **Data Visualization**
  Known for its powerful data visualization libraries (like ggplot2, lattice, etc.), R is great for producing publication-quality plots.
- **Extensible**
  Easy to extend with custom functions, as well as C++ integration for performance optimization.
- **Cross-Platform**
  Runs on multiple operating systems like Windows, Mac, and Linux, making it flexible for users on different platforms.

Disadvantages of R

- **Memory Intensive**
  R stores objects in memory, which can become an issue with large datasets as it may lead to memory limitations.
- **Steep Learning Curve**
  Beginners might find R challenging to learn, especially compared to languages like Python.
- **Slower Than Compiled Languages**
  Being an interpreted language, R tends to be slower than compiled languages like C++ or Java.
- **Less Friendly for Big Data**
  Without specific packages (e.g., data.table, SparkR), R is not as efficient for handling large datasets compared to Python or Hadoop/Spark.
- **Complexity in Package Management**
  Managing and installing packages can sometimes lead to compatibility issues between different libraries and R versions.

Data Types in R

- **Numeric**
  Represents real numbers or decimal values. E.g., `5.2`, `3.14`
- **Integer**
  Whole numbers. E.g., `10L` (Note the 'L' indicating integer in R)
- **Logical**
  Represents Boolean values. E.g., `TRUE`, `FALSE`
- **Character**
  Text data or strings. E.g., `"Hello World"`
- **Factor**
  Categorical data with defined levels. E.g., `factor(c("Low", "Medium", "High"))`
- **Complex**
  Numbers with imaginary components. E.g., `2+3i`
- **Raw**
  Used to store raw bytes.

Basics of Vector, Matrix, and Data Frame

- **Vector**
  A one-dimensional array that holds data of the same type.
  Example: `v <- c(1, 2, 3, 4)`
- **Matrix**
  A two-dimensional array where all elements are of the same data type.
  Example: `m <- matrix(1:9, nrow=3, ncol=3)`
- **Data Frame**
  A table or a two-dimensional array-like structure where columns can have different data types.
  Example: `df <- data.frame(name=c("A", "B"), age=c(25, 30))`

Basic Functions in R

- **seq()**
  Generates a sequence of numbers.
  Example: `seq(1, 10, by=2) # Generates 1, 3, 5, 7, 9`
- **rep()**
  Repeats elements of a vector.
  Example: `rep(1:3, times=2) # Generates 1, 2, 3, 1, 2, 3`
- **scan()**
  Reads input from the console or a file.
  Example: `scan() # Prompts user for input`
- **factor()**
  Converts a vector to a factor, which is used to handle categorical data.
  Example: `factor(c("Low", "Medium", "High"))`
- **table()**
  Creates a contingency table of counts for factor data.
  Example: `table(factor(c("A", "B", "A", "C")))`
- **cut()**
  Divides continuous data into intervals or bins.

Sampling and Frequency Tables

- **Sampling** is the process of selecting a subset of data from a population to estimate characteristics of the whole population.
- **Frequency Tables** show how often values in a dataset occur. There are two types:
    - **Ungrouped Frequency Tables**: Lists each unique value and its frequency.
    - **Grouped Frequency Tables**: Divides the data into intervals (bins) and counts the frequency of data within each bin.

Forming Frequency Tables

Ungrouped Frequency Table using `table()`

- **Example**:

```
  data <- c(1, 2, 2, 3, 3, 3, 4, 4, 5)
  ungrouped_table <- table(data)
  print(ungrouped_table)
#> data
#> 1 2 3 4 5
#> 1 2 3 2 1
```

Forming Frequency Tables

Grouped Frequency Table using `cut()` and `table()`

```
data <- c(1, 2, 2, 3, 3, 3, 4, 4, 5)
grouped_data <- cut(data, breaks=3)
grouped_table <- table(grouped_data)
print(grouped_table)
#> grouped_data
#> (0.996,2.33]  (2.33,3.67]     (3.67,5]
#>            3            3            3
```

Simple Random Sampling (SRS)

- **SRSWR (Simple Random Sampling With Replacement)**
  Each selected unit is replaced back into the population before the next draw.
- **SRSWOR (Simple Random Sampling Without Replacement)**
  Once a unit is selected, it is not replaced back into the population for future draws.

SRSWR Example using `sample()`

- **Example**:

```
  population <- 1:10
  srswr_sample <- sample(population, size=5, replace=TRUE)
  print(srswr_sample)
#> [1]  7 10  3  4 10
```

SRSWOR Example using sample()

- **Example**:

```r
population <- 1:10
srswor_sample <- sample(population, size=5, replace=FALSE)
print(srswor_sample)
#> [1]  9  3  1  8 10
```

This will give a sample of 5 numbers without replacement from the population.

Measures of Central Tendency

- **Central Tendency** refers to the statistical measures that identify a single value as representative of an entire dataset.
- The three most common measures are:
  - **Mean**: The average value of the dataset.
  - **Median**: The middle value when the data is arranged in order.
  - **Mode**: The value that appears most frequently in the dataset.
- **Use in R**:

```r
data <- c(1, 2, 3, 4, 4, 5, 6)
mean(data)   # Mean
median(data) # Median
```

Descriptive Measures

- **sum()**: Returns the sum of all elements in the vector.

```
  sum(c(1, 2, 3, 4, 5)) # 15
#> [1] 15
```

- **sort()**: Sorts the elements of a vector in ascending or descending order.

```
sort(c(5, 2, 8, 1, 3)) # 1, 2, 3, 5, 8
#> [1] 1 2 3 5 8
```

- **min()**: Returns the minimum value from the dataset.

```
min(c(1, 2, 3, 4, 5)) # 1
#> [1] 1
```

Descriptive Measures

- **max()**: Returns the maximum value from the dataset.

```
max(c(1, 2, 3, 4, 5)) # 5
#> [1] 5
```

- **length()**: Returns the number of elements in the vector.

```
length(c(1, 2, 3, 4, 5)) # 5
#> [1] 5
```

Mean, Median, and Mode

- **Mean**: Sum of the data divided by the number of values.

```
  data <- c(1, 2, 3, 4, 4, 5, 6)
  mean(data) # 3.57
#> [1] 3.571429
```

- **Median**: The middle value when the data is sorted.

```
median(data) # 4
#> [1] 4
```

Mean, Median, and Mode

- **Mode**: The value that appears most frequently.

```
mode_func <- function(x) {
    uniq_x <- unique(x)
    uniq_x[which.max(tabulate(match(x, uniq_x)))]
}
mode_func(data) # 4
#> [1] 4
```

Geometric Mean

- **Geometric Mean**: The nth root of the product of all values.
  - Used in datasets that involve rates of growth.
- **Formula**:

$$G = \left(\prod_{i=1}^{n} x_i \right)^{1/n}$$

- **Use in R**:

```r
  data <- c(1, 2, 3, 4, 5)
  geometric_mean <- prod(data)^(1/length(data))
  print(geometric_mean) # 2.605
#> [1] 2.605171
```

Harmonic Mean

- **Harmonic Mean**: The reciprocal of the arithmetic mean of the reciprocals.
  - Used when dealing with rates or ratios.
- **Formula**:

$$H = \frac{n}{\sum_{i=1}^{n} \frac{1}{x_i}}$$

- **Use in R**:

```
  data <- c(1, 2, 3, 4, 5)
  harmonic_mean <- length(data) / sum(1 / data)
  print(harmonic_mean) # 2.189
#> [1] 2.189781
```

Measures of Dispersion

- **Dispersion** describes the spread of the data around a central value.
- The common measures of dispersion include:
    - **Range**
    - **Mean Deviation**
    - **Interquartile Range (IQR)**
    - **Quartile Deviation**
    - **Standard Deviation (SD)**
    - **Variance**
    - **Coefficient of Variation (CV)**
    - **Quantiles**
    - **Summary Statistics**

Range

- **Range**: The difference between the maximum and minimum values in a dataset. A simple measure, but sensitive to outliers.
    - Formula:

$$\text{Range} = \text{Max} - \text{Min}$$

- **Use in R**:

```
  data <- c(3, 7, 2, 9, 5)
  range_value <- max(data) - min(data)
  print(range_value) # 7
#> [1] 7
```

Mean Deviation

- **Mean Deviation**: The average of the absolute deviations from the mean or median.
- **Formula**:

$$MD = \frac{1}{n}\sum_{i=1}^{n} |x_i - \bar{x}|$$

- **Use in R**:

```
  data <- c(3, 7, 2, 9, 5)
  mean_deviation <- mean(abs(data - mean(data)))
  print(mean_deviation) # 2.4
#> [1] 2.24
```

Interquartile Range (IQR)

- **IQR**: The difference between the third quartile (Q3) and the first quartile (Q1), representing the middle 50% of the data. Less sensitive to outliers than the range.
  - Formula:

$$\text{IQR} = Q3 - Q1$$

- **Use in R**:

```
  data <- c(3, 7, 2, 9, 5)
  iqr_value <- IQR(data)
  print(iqr_value) # 4
#> [1] 4
```

Quartile Deviation

- **Quartile Deviation (QD)**: Half of the Interquartile Range (IQR). It is also called the semi-interquartile range.
    - Formula:

$$\text{QD} = \frac{Q3 - Q1}{2}$$

- **Use in R**:

```
  data <- c(3, 7, 2, 9, 5)
  qd_value <- IQR(data) / 2
  print(qd_value) # 2
#> [1] 2
```

Standard Deviation (SD)

- **Standard Deviation**: Measures the average distance of each data point from the mean. Widely used to describe variability in data.
  - Formula:

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

- **Use in R**:

```r
  data <- c(3, 7, 2, 9, 5)
  sd_value <- sd(data)
  print(sd_value) # 2.607
#> [1] 2.863564
```

Variance

- **Variance**: The square of the standard deviation, representing the average squared deviation from the mean. Gives more weight to larger deviations.
    - Formula:

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

- **Use in R**:

```
  data <- c(3, 7, 2, 9, 5)
  variance_value <- var(data)
  print(variance_value) # 6.8
#> [1] 8.2
```

Coefficient of Variation (CV)

- **CV**: The ratio of the standard deviation to the mean, expressed as a percentage. Useful for comparing the relative variability between datasets with different units or means.
  - Formula:

$$\text{CV} = \frac{SD}{\bar{x}} \times 100$$

- **Use in R**:

```
  data <- c(3, 7, 2, 9, 5)
  cv_value <- (sd(data) / mean(data)) * 100
  print(cv_value) # 52.71
#> [1] 55.06854
```

Quantiles

- **Quantiles**: Cut points dividing the dataset into equal-sized intervals (e.g., quartiles, percentiles).
    - **Use in R**:

```
data <- c(3, 7, 2, 9, 5)
quantiles <- quantile(data)
print(quantiles)
#>   0%  25%  50%  75% 100%
#>    2    3    5    7    9
```

Summary

- **Summary**: Provides a quick overview of key statistics, including min, Q1, median, Q3, and max.
    - **Use in R**:

```
summary(data)
#>    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
#>     2.0     3.0     5.0     5.2     7.0     9.0
```

Moments, Skewness, and Kurtosis

- **Moments**: Statistical measures used to understand the shape of a distribution. They help calculate skewness and kurtosis.
  - **Raw Moments**: Measures relative to the origin.
  - **Central Moments**: Measures relative to the mean.
- **Skewness**: A measure of asymmetry in the distribution.
  - Positive skewness: Data skewed to the right.
  - Negative skewness: Data skewed to the left.
- **Kurtosis**: A measure of the "tailedness" or peakedness of the distribution.
  - High kurtosis: More outliers (leptokurtic).
  - Low kurtosis: Fewer outliers (platykurtic).

Raw Moments

- **Raw Moments**: Measures based on deviations from zero. The raw moments help understand the general shape of the data distribution.
  - The k-th raw moment is given by:

$$M_k = \frac{1}{n} \sum_{i=1}^{n} x_i^k$$

- **Example in R**:

```
  data <- c(2, 4, 6, 8, 10)
  raw_moment_2 <- mean(data^2) # Second raw moment
  print(raw_moment_2) # 44
#> [1] 44
```

Central Moments

- **Central Moments**: Measures based on deviations from the mean. The second central moment is the variance, and the third and fourth moments are related to skewness and kurtosis.
    - The k-th central moment is given by:

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

- **Example in R**:

```
  data <- c(2, 4, 6, 8, 10)
  central_moment_2 <- mean((data - mean(data))^2) # Second central moment (Va
  print(central_moment_2) # 8
#> [1] 8
```

Skewness

- **Skewness**: Measures the asymmetry of the distribution relative to the mean. A skewness value of 0 indicates a symmetric distribution.
  - Formula:

$$\text{Skewness} = \frac{1}{n}\sum_{i=1}^{n}\left(\frac{x_i - \bar{x}}{\sigma}\right)^3$$

- **Interpretation**:
  - Skewness > 0: Positively skewed (right-tailed).
  - Skewness < 0: Negatively skewed (left-tailed).
  - Skewness 0: Symmetric distribution.

Skewness

- **Example in R**:

```
  library(e1071)  # For skewness function
  data <- c(2, 4, 6, 8, 10)
  skewness_value <- skewness(data)
  print(skewness_value) # 0
#> [1] 0
```

Kurtosis

- **Kurtosis**: Measures the “tailedness” or peak of the distribution.
  - Formula:

$$\text{Kurtosis} = \frac{1}{n}\sum_{i=1}^{n}\left(\frac{x_i - \bar{x}}{\sigma}\right)^4$$

- **Interpretation**:
  - **High Kurtosis** (> 3): Leptokurtic (sharp peak, heavy tails).
  - **Low Kurtosis** (< 3): Platykurtic (flat peak, light tails).
  - **Normal Kurtosis** (3): Mesokurtic.

Kurtosis

- **Example in R**:

```
  library(e1071)  # For kurtosis function
  data <- c(2, 4, 6, 8, 10)
  kurtosis_value <- kurtosis(data)
  print(kurtosis_value) # -1.3
#> [1] -1.912
```

Moment Measures of Skewness and Kurtosis

- **Moment-Based Skewness**:
  - $$\text{Skewness} = \frac{\mu_3}{\sigma^3}$$
  - Third central moment μ_3 represents the asymmetry.
- **Moment-Based Kurtosis**:
  - $$\text{Kurtosis} = \frac{\mu_4}{\sigma^4}$$
  - Fourth central moment μ_4 indicates how sharp or flat the distribution is.

Moment Measures of Skewness and Kurtosis

- **Example in R** (using moments):

```r
data <- c(2, 4, 6, 8, 10)
third_moment <- mean((data - mean(data))^3)
fourth_moment <- mean((data - mean(data))^4)

skewness_moment <- third_moment / (sd(data)^3)
kurtosis_moment <- fourth_moment / (sd(data)^4)
```

Moment Measures of Skewness and Kurtosis

- **Example in R** (using moments):

```
  print(skewness_moment) # 0
#> [1] 0
  print(kurtosis_moment) # 1.875
#> [1] 1.088
```

Graphical Methods

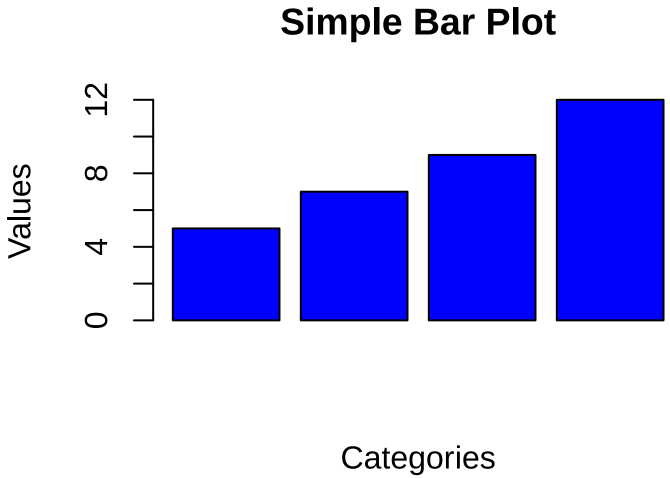
- **Bar Plots**:
  - Simple and Multiple (Side by Side and Subdivided).
  - Useful for comparing categorical data.
  - `barplot()` function in R.
- **Pie Chart**:
  - Displays the proportion of categories.
  - `pie()` function.
- **Histogram**:
  - Shows the distribution of continuous data.
  - `hist()` function.
- **Scatter Plot**:
  - Visualizes relationships between two variables.
  - `plot()` function.
- **Line Plot**:
  - Adds lines to a scatter plot to show trends.
  - `lines()` function.

Simple Bar Plot

- **Bar Plots** are used to display categorical data with rectangular bars.
- **Syntax in R**:

```
data <- c(5, 7, 9, 12)
barplot(data, main="Simple Bar Plot", xlab="Categories",
         ylab="Values", col="blue")
```

Simple Bar Plot



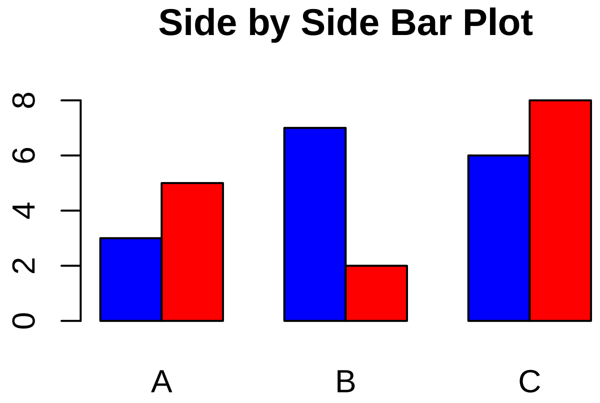
Multiple Bar Plot (Side by Side)

- **Multiple Bar Plot**: Displays multiple sets of data side by side for comparison.
- **Syntax in R**:

```r
data <- matrix(c(3, 5, 7, 2, 6, 8), nrow=2)
barplot(data, beside=TRUE, main="Side by Side Bar Plot", col=c("blue", "red'
```

Multiple Bar Plot (Side by Side)

- **Syntax in R**:

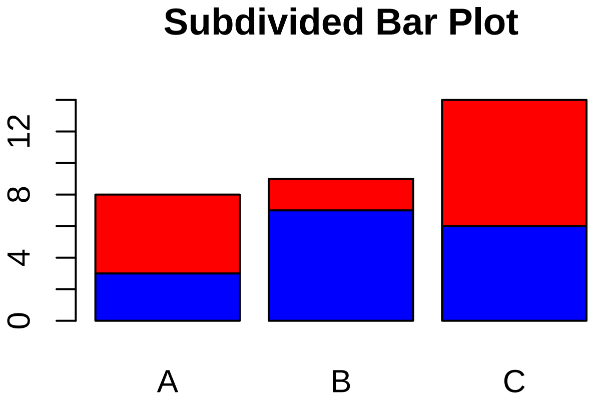


Multiple Bar Plot (Subdivided)

- **Subdivided Bar Plot**: Bars are stacked on top of each other to show the cumulative total.
- **Syntax in R**:

```
data <- matrix(c(3, 5, 7, 2, 6, 8), nrow=2)
barplot(data, beside=FALSE, main="Subdivided Bar Plot", col=c("blue", "red"
```

Multiple Bar Plot (Subdivided)

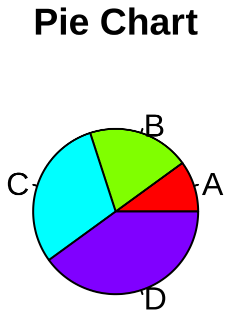


Pie Chart

- **Pie Chart**: A circular chart divided into sectors to show relative proportions of categories.
- **Syntax in R**:

```r
data <- c(10, 20, 30, 40)
labels <- c("A", "B", "C", "D")
pie(data, labels=labels, main="Pie Chart", col=rainbow(4))
```

Pie Chart

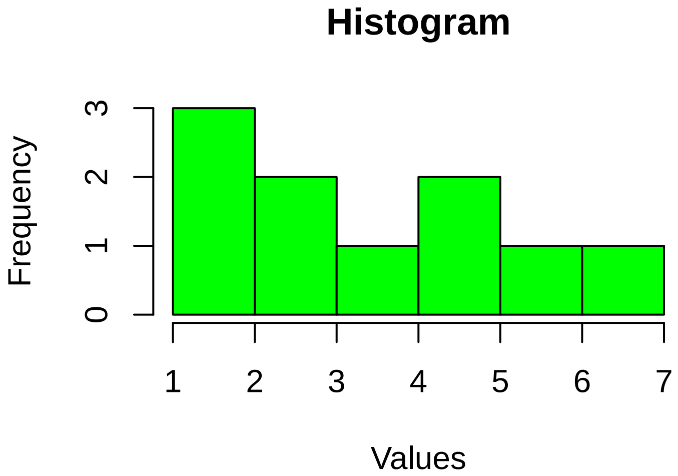


Histogram

- **Histogram**: Displays the distribution of continuous data by dividing it into bins.
- **Syntax in R**:

```r
data <- c(1, 2, 2, 3, 3, 4, 5, 5, 6, 7)
hist(data, main="Histogram", xlab="Values", col="green", breaks=5)
```

Histogram

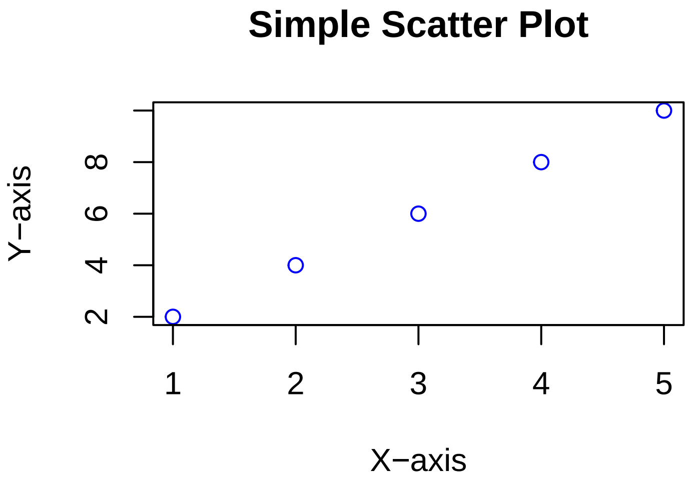


Plot Function

- **plot()**: Used for basic 2D plotting in R.
- **Syntax in R**:

```
x <- c(1, 2, 3, 4, 5)
y <- c(2, 4, 6, 8, 10)
plot(x, y, type="p", main="Simple Scatter Plot", xlab="X-axis", ylab="Y-axi
```

Plot Function

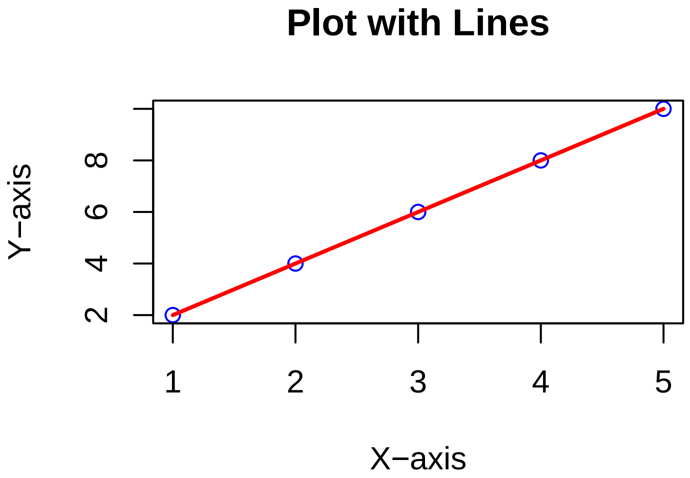


Lines Function

- **lines()**: Adds lines to an existing plot. The lines() function draws lines connecting the data points on the existing plot.
- **Syntax in R**:

```
x <- c(1, 2, 3, 4, 5)
y <- c(2, 4, 6, 8, 10)
plot(x, y, type="p", main="Plot with Lines", xlab="X-axis", ylab="Y-axis",
lines(x, y, col="red", lwd=2)
```

Lines Function



Probability Distributions

- Probability distributions describe the likelihood of various outcomes in random events.
- Common distributions:
    - **Binomial**
    - **Poisson**
    - **Normal**
    - **Chi-square**
    - **t-distribution**
    - **F-distribution**
- In R, these distributions are evaluated using:
    - `d` (density function),
    - `p` (cumulative distribution function),
    - `q` (quantile function),
    - `r` (random variates).

Binomial Distribution

- **Binomial distribution** models the number of successes in a fixed number of independent Bernoulli trials.
    - Parameters: (n) (number of trials), (p) (probability of success).
- **R Functions**:
    - `dbinom(x, size, prob)`: Probability mass function (PMF).
    - `pbinom(q, size, prob)`: Cumulative distribution function (CDF).
    - `qbinom(p, size, prob)`: Quantile function.
    - `rbinom(n, size, prob)`: Random variates.

Binomial Distribution

- **Example**:

```
  dbinom(3, size=10, prob=0.5)  # P(X=3) for 10 trials, p=0.5
#> [1] 0.1171875
  pbinom(3, size=10, prob=0.5)  # P(X<=3)
#> [1] 0.171875
  rbinom(5, size=10, prob=0.5)  # Generate 5 random binomial variates
#> [1] 6 6 4 8 4
```

Poisson Distribution

- **Poisson distribution** models the number of events occurring in a fixed interval of time or space.
  - Parameter: ( ) (average number of events).
- **R Functions**:
  - `dpois(x, lambda)`: PMF.
  - `ppois(q, lambda)`: CDF.
  - `qpois(p, lambda)`: Quantile function.
  - `rpois(n, lambda)`: Random variates.

Poisson Distribution

- **Example**:

```
  dpois(2, lambda=5)  # P(X=2) when lambda=5
#> [1] 0.08422434
  ppois(2, lambda=5)  # P(X<=2)
#> [1] 0.124652
  rpois(5, lambda=5)  # Generate 5 random Poisson variates
#> [1] 6 2 6 3 3
```

Normal Distribution

- **Normal distribution** (Gaussian distribution) is the most common continuous distribution, defined by mean ( ) and standard deviation ( ).
  - Parameters: ( ) (mean), ( ) (standard deviation).
- **R Functions**:
  - `dnorm(x, mean, sd)`: Probability density function (PDF).
  - `pnorm(q, mean, sd)`: CDF.
  - `qnorm(p, mean, sd)`: Quantile function.
  - `rnorm(n, mean, sd)`: Random variates.

Normal Distribution

- **Example**:

```
  dnorm(0, mean=0, sd=1)  # P(X=0) for standard normal
#> [1] 0.3989423
  pnorm(1.96, mean=0, sd=1)  # P(X<=1.96) for standard normal
#> [1] 0.9750021
  rnorm(5, mean=0, sd=1)  # Generate 5 random normal variates
#> [1] -0.3062899  0.3057279  0.2631002  0.9816717  0.1961402
```

Chi-Square Distribution

- **Chi-square distribution** is commonly used in hypothesis testing and confidence intervals for variance.
  - Parameter: (k) (degrees of freedom).
- **R Functions**:
  - `dchisq(x, df)`: PDF.
  - `pchisq(q, df)`: CDF.
  - `qchisq(p, df)`: Quantile function.
  - `rchisq(n, df)`: Random variates.

Chi-Square Distribution

- **Example**:

```
  dchisq(3, df=2)  # P(X=3) for chi-square with 2 degrees of freedom
#> [1] 0.1115651
  pchisq(3, df=2)  # P(X<=3)
#> [1] 0.7768698
  rchisq(5, df=2)  # Generate 5 random chi-square variates
#> [1] 1.660393043 0.990008417 0.185147941 0.255591299 0.006957785
```

t-Distribution

- **t-distribution** is used in hypothesis testing and confidence intervals when the sample size is small.
    - Parameter: (df) (degrees of freedom).
- **R Functions**:
    - `dt(x, df)`: PDF.
    - `pt(q, df)`: CDF.
    - `qt(p, df)`: Quantile function.
    - `rt(n, df)`: Random variates.

t-Distribution

- **Example**:

```
  dt(1.96, df=10)  # P(T=1.96) for t-distribution with 10 degrees of freedom
#> [1] 0.06509475
  pt(1.96, df=10)  # P(T<=1.96)
#> [1] 0.9607819
  rt(5, df=10)  # Generate 5 random t-distribution variates
#> [1]  0.6387792  0.3444552 -0.2000194  0.8770567 -0.9640084
```

F-Distribution

- **F-distribution** is used to compare two variances (ANOVA).
    - Parameters: (df_1) (numerator degrees of freedom), (df_2) (denominator degrees of freedom).
- **R Functions**:
    - `df(x, df1, df2)`: PDF.
    - `pf(q, df1, df2)`: CDF.
    - `qf(p, df1, df2)`: Quantile function.
    - `rf(n, df1, df2)`: Random variates.

F-Distribution

- **Example**:

```
  df(1, df1=5, df2=10)  # P(F=1) for F-distribution with 5 and 10 df
#> [1] 0.4954798
  pf(1, df1=5, df2=10)  # P(F<=1)
#> [1] 0.5348806
  rf(5, df1=5, df2=10)  # Generate 5 random F-distribution variates
#> [1] 2.5728202 1.2987937 2.5118008 0.8337675 0.2682479
```

Covariance for Bivariate Data

- **Covariance** measures the relationship between two variables and how they vary together. Covariance helps determine the direction of the linear relationship between variables.
  - Formula:

$$\text{Cov}(X, Y) = \frac{1}{n-1}\sum_{i=1}^{n}(X_i - \bar{X})(Y_i - \bar{Y})$$

- **R Function**:
  - `cov(x, y)`: Computes covariance between `x` and `y`.

Covariance for Bivariate Data

- **Example**:

```
  x <- c(2, 4, 6, 8)
  y <- c(3, 7, 5, 10)
  cov(x, y)  # Covariance between x and y
#> [1] 6.333333
```

Pearson's and Spearman's Correlation Coefficient

- **Pearson's Correlation**: Measures the strength and direction of the linear relationship between two continuous variables.
  - Formula:

$$r = \frac{\text{Cov}(X, Y)}{\sigma_X \sigma_Y}$$

- **Spearman's Correlation**: A non-parametric measure of rank correlation (monotonic relationships).
- **R Function**:
  - `cor(x, y, method="pearson")`: Pearson's correlation.
  - `cor(x, y, method="spearman")`: Spearman's correlation.

Pearson’s and Spearman’s Correlation Coefficient

Pearson’s `r` ranges from -1 to 1, indicating perfect negative or positive correlation, while Spearman’s measures the strength of monotonic relationships. - **Example**:

```
  x <- c(2, 4, 6, 8)
  y <- c(3, 7, 5, 10)
  cor(x, y, method="pearson")  # Pearson's correlation
#> [1] 0.8214416
  cor(x, y, method="spearman") # Spearman's correlation
#> [1] 0.8
```

Linear Regression Models

- **Linear Regression**: Models the relationship between a dependent variable and one or more independent variables using a linear equation.
  - Formula:

$$y = \beta_0 + \beta_1 x + \epsilon$$

  - $ _0$: Intercept, β_1: Slope.
- **Fitting a Linear Model in R**:
  - `lm(y ~ x)`: Fits a simple linear regression model.

Linear Regression Models

- **Example**:

```r
x <- c(2, 4, 6, 8)
y <- c(3, 7, 5, 10)
model <- lm(y ~ x)
summary(model)  # Displays the model summary
#> 
#> Call:
#> lm(formula = y ~ x)
#> 
#> Residuals:
#>    1    2    3    4 
#> -0.4  1.7 -2.2  0.9 
#> 
#> Coefficients:
#>             Estimate Std. Error t value Pr(>|t|)
#> (Intercept)   1.5000     2.5544   0.587    0.617
```

Thank You!

- Thank you for your attention!
- Feel free to reach out with any questions.

Contact:

- **Email**: mahesh.divakaran01@gmail.com
- **LinkedIn**: linkedin.com/in/imaheshdivakaran