



AdusumilliGopalakrishnaiah & Sugarcane Growers Siddhartha Degree College of Arts and Science

Autonomous College :: Aided College of Govt. of AP

NAAC 'A' Grade College

Vuyyuru, Krishna (Dt.), Andhra Pradesh-521165

VALUE ADDED COURSE

TITLE: Statistical Computing

VAC CODE: STCO-001

On 01-04-2023 to 10-05-2023

Duration of the Course: 30DAYS

Organized By

Department of Computer Science



A.G. & S.G. Siddhartha Degree College of Arts & Science

Vuyyuru-521165, Krishna District, Andhra Pradesh

(Managed by: Siddhartha Academy of General & Technical Education, Vijayawada-10)

An Autonomous College in the Jurisdiction of Krishna University

Accredited by NAAC with "A" Grade ISO 9001:2015 Certified Institution



DEPARTMENT OF COMPUTER SCIENCE

Value Added Course in Statistical Computing

Name of the Lecturer : Sri. T. Naga Prasada Rao

Class : II B.Sc (MSCS)

Duration of the Course : 30 Days

VAC Code : STCO-001

A.G. & S.G. Siddhartha Degree College of Arts & Science

Vuyyuru-521165, Krishna District, Andhra Pradesh

Value Added Course in Statistical Computing

Objectives:

This course will review and expand upon core topics in probability and statistics through the study and practice of data analysis and graphical interpretation using 'R'.

Methodology:

- 1) Lecture-based Learning
- 2) Experimental Learning
- 3) ICT

Duration:

30 Days

A.G. & S.G. Siddhartha Degree College of Arts & Science


Vuyyuru-521165, Krishna District, Andhra Pradesh


Value Added Course in Statistical Computing

Student Enrolment Sheet

Class: II B.Sc. (MSCS)

S. No	Roll No.	Name of the Student	Signature
1	2155301	K. Dharani	K. Dharani
2	2155302	G. Divya	G. Divya
3	2155303	M. Sri Lakshmi	M. Sri Lakshmi
4	2155304	S. Sai Sowmya	S. Sai Sowmya
5	2155305	A. Sushanth	A. Sushanth
6	2155306	B. Manoj Phanindra	B. Manoj Phanindra
7	2155307	J. Keerthi Priya	J. Keerthi Priya
8	2155308	K. Alekhya	K. Alekhya
9	2155310	N. Anusha	N. Anusha
10	2155311	A. Chakradhar	A. Chakradhar
11	2155312	MD. Khadeera Begam	MD. Khadeera Begam
12	2155314	J. Jhansi Lakshmi	J. Jhansi Lakshmi
13	2155315	K. Tulasi	K. Tulasi
14	2155316	P. Hima Sri	P. Hima Sri
15	2155317	K. Naga Sravani	K. Naga Sravani
16	2155319	M. Karuna Sri	M. Karuna Sri
17	2155320	P. Phani Supraja	P. Phani Supraja
18	2155321	K. Hema Sri	K. Hema Sri
19	2155322	R. Durga Bhavani	R. Durga Bhavani
20	2155323	D. Naga Gireesha	D. Naga Gireesha


Head of the Department of Computer Science
A.G. & S.G. Siddhartha Degree College
Vuyyuru-521165


Principal
Adusumilli Gopalakrishnaiah & Sugarcane
Siddhartha Degree College of Arts & Science,
Vuyyuru-521165, Krishna District

A.G. & S.G. Siddhartha Degree College of Arts & Science
Vuyyuru-521165, Krishna District, Andhra Pradesh

Value Added Course in Statistical Computing

Date(s) From: 01-04-2023 to: 10-05-2023

Content	Module No.
Introduction to R Programming, Data types of R, Expressions, Variables, Functions	I
Conditional Statements, Looping statements, Control flow functions	II
Arrays, Matrix, Vectors, Factors, Packages	III
Data frames, Data frame access, ordering data frames, functions of data frames	IV

What is R:

R is a programming language and environment used for statistical computing and graphics. It was first released in 1993 and has since become a popular tool for data analysis and research in academia, industry, and government.



R is open source software, which means that it is free to use, distribute, and modify. It is highly extensible and has a large collection of packages and libraries that enable users to perform a wide range of statistical analyses, data visualization, and machine learning tasks.

R is widely considered to be one of the most powerful and flexible tools for data analysis available today.

Data types in R:

In R, there are several data types that are used to store and manipulate data. Some of the commonly used data types in R are:

- **Logical:**

In R, the logical data type is used to store Boolean values which can take on one of two possible values: TRUE or FALSE. The logical data type is commonly used in conditional statements and logical operations. For example, consider the following code:

```
x <- 5
y <- 10
z <- x < y
```

In this code, we first assign the value 5 to the variable x and the value 10 to the variable y. We then use the logical operator "<" to compare x and y, which returns TRUE because x is less than y. This value is then assigned to the variable z.

- **Numeric:**

In R, the numeric data type is used to store numbers, including integers and real numbers. Numeric values are represented by the "numeric" data type in R. Numeric variables can be used in a variety of operations, including arithmetic operations such as addition, subtraction, multiplication, and division. For example:

```
a <- 5
b <- 10
c <- a + b
d <- b - a
```

```
e <- a * b
f <- b / a
```

In this code, we first assign the values 5 and 10 to the variables a and b, respectively. We then perform various arithmetic operations using these variables, including addition (c), subtraction (d), multiplication (e), and division (f).

Numeric variables can also be used in statistical calculations and analysis, as well as in creating graphs and visualizations. In fact, R has many built-in functions and packages that are specifically designed for statistical analysis and visualization.

- **Integer:**

In R, the integer data type is used to store whole numbers. Integers are represented by the "integer" data type in R. Integers can be used in a variety of operations, including arithmetic operations such as addition, subtraction, multiplication, and division. For example:

```
a <- 5L
b <- 10L
c <- a + b
d <- b - a
e <- a * b
f <- b / a
```

In this code, we first assign the values 5 and 10 to the variables a and b, respectively, using the "L" suffix to indicate that they should be treated as integers. We then perform various arithmetic operations using these variables, including addition (c), subtraction (d), multiplication (e), and division (f).

- **Character:**

A character data type represents text or string values. A character object is created by enclosing text within quotes (either single or double quotes). For example:

```
my_string <- "Hello, world!"
```

In this example, my_string is a character object that contains the text "Hello, world!". Note that the quotes are required to indicate that this is a character value.

- **Double:**

A double data type is used to represent floating-point numbers (i.e., numbers with a decimal point). Double precision is the default for R's numeric data type. For example, you can create a double variable my_var with the value 3.14159 like this:

```
my_var <- 3.14159
```

R provides a variety of operators and functions for working with numeric data, including the basic arithmetic operators (+, -, *, /), as well as more advanced functions for trigonometry, logarithms, and other mathematical operations.

- **Complex:**

A complex data type is used to represent numbers with both a real and imaginary component. Complex numbers are created using the complex() function or by appending an i or I to the imaginary component. For example:

```
# Create a complex number using the complex() function
my_complex <- complex(real = 3, imaginary = 4)
```

```
# Create a complex number using the i notation
my_complex2 <- 2 + 3i
```

In these examples, `my_complex` and `my_complex2` are both complex numbers. `my_complex` has a real component of 3 and an imaginary component of 4, while `my_complex2` has a real component of 2 and an imaginary component of 3.

- **Raw:**

A raw data type is used to represent raw bytes of data. Raw vectors are created using the `raw()` function or by using a hexadecimal notation. For example:

```
# Create a raw vector using the raw() function
my_raw <- raw(4)
my_raw[1] <- as.raw(0x4c)
my_raw[2] <- as.raw(0x6f)
my_raw[3] <- as.raw(0x72)
my_raw[4] <- as.raw(0x65)
```

```
# Create a raw vector using hexadecimal notation
my_raw2 <- as.raw(c(0x48, 0x65, 0x6c, 0x6c, 0x6f))
```

In these examples, `my_raw` and `my_raw2` are both raw vectors. `my_raw` contains the raw bytes for the ASCII characters "L", "o", "r", and "e", while `my_raw2` contains the raw bytes for the ASCII characters "H", "e", "l", "l", and "o".

Raw vectors are typically used to represent binary data, such as images, audio files, or network packets. R provides a variety of functions for working with raw vectors, including functions for reading and writing binary data to files, converting raw data to other formats, and manipulating the bytes within a raw vector.

Expressions:

In R, an expression is a set of instructions that can be evaluated to produce a value or result. Expressions can include arithmetic or logical operations, function calls, and variable assignments. For example, the following code is an expression that calculates the sum of two numbers:

```
2 + 3
```

This expression evaluates to the value 5, which can be printed or assigned to a variable. Expressions can also include function calls, such as the following code:

```
sqrt(25)
```

This expression calls the `sqrt()` function with an argument of 25, which evaluates to the value 5. In addition to simple expressions, R also supports more complex expressions that combine multiple operations and function calls. For example, the following code calculates the factorial of a number:


```

factorial <- function(n)
{
  if (n == 0)
  {
    return(1)
  }
  Else
  {
    return(n * factorial(n - 1))
  }
}
factorial(5)

```

This expression defines a function called `factorial()` that recursively calculates the factorial of a number. When the expression `factorial(5)` is evaluated, it returns the value 120.

Variables:

Variables are used to store data values that can be used in expressions or operations. A variable is created by assigning a value to a name using the assignment operator `<-` or `=`. The name of the variable can contain letters, numbers, underscores, and dots, but it cannot start with a number or a dot, and it cannot be a reserved keyword. For example, to create a variable called `x` with a value of 5, we can use the following code:

```
x <- 5
```

We can then use the variable `x` in an expression or operation, such as:

```
y <- x + 2
```

This code assigns the result of adding 2 to the value of `x` to a new variable called `y`.

Variables can store values of different types, such as numbers, strings, logical values, and more complex data structures like arrays and data frames. The type of a variable is determined by the value it holds, and can be checked using the `class()` function.

Variables in R are mutable, which means that their values can be changed by assigning a new value to the same name. For example, we can change the value of `x` by assigning a new value to it:

```
x <- 10
```

Overall, variables are a fundamental concept in programming and are used extensively in R to store, manipulate, and analyse data.

Functions:

A function is a set of instructions that perform a specific task and can be called repeatedly with different inputs. R provides a series of in-built functions, and it allows the user to create their own functions. Functions are used to avoid repeating the same task and to reduce complexity. A function should be:

- Written to carry out a specified task.
- May or may not have arguments
- Contain a body in which our code is written.
- May or may not return one or more output values.

Function Definition:

An R function is created by using the keyword 'function'. There is the following syntax of R function:

```
func_name <- function(arg_1, arg_2, ...)  
{  
  Function body  
}
```

Here is an example of a simple function in R:

```
square <- function(x)  
{  
  result <- x2  
  return(result)  
}
```

This function takes an input parameter x, calculates its square, and returns the result as an output value. To call the function with a specific input value, we can simply pass the value as an argument:

```
y <- square(5)
```

This code calls the square() function with an input value of 5, and assigns the resulting output value (25) to the variable y.

Conditional Statements:

Conditional statements in R allow you to execute different blocks of code based on the result of a logical test.

1) if Statement:

The most common conditional statement is the if statement, which has the following syntax:

```
if (condition)  
{  
  # Code to execute if the condition is TRUE  
}
```

For example, the following code uses an if statement to print a message if a number is positive:

```
x <- 5  
if (x > 0)  
{  
  print("x is positive")  
}
```

This code checks whether x is greater than 0, and if it is, prints the message "x is positive".

2) if-else statement:

This statement allows you to execute one block of code if a condition is TRUE, and another block of code if the condition is FALSE. The basic syntax of an if-else statement in R is as follows:

```
if (condition)
{
    # code to execute if condition is TRUE
}
else
{
    # code to execute if condition is FALSE
}
For example:
x <- -5
if (x > 0)
{
    print("x is positive")
}
else
{
    print("x is negative or zero")
}
```

Loops:

Loops in R allow you to execute the same block of code multiple times, based on certain conditions or criteria. The most common types of loops are ‘for’ and ‘while’ loops.

1) For Loop

The ‘for loop’ has the following syntax:

```
for (variable in sequence)
{
    # Code to execute for each value of the variable
}
```

For example, the following code uses a for loop to print the first five positive integers:

```
for (i in 1:5)
{
    print(i)
}
```

This code loops through the sequence 1:5, and for each value of i, prints the value to the console.

2) While loop:

The while loop has the following syntax:

```
while (condition)
{
    # Code to execute while the condition is TRUE
}
```

```
}
```

For example, the following code uses a while loop to print the first five positive even numbers:

```
i <- 2
count <- 0
while (count < 5)
{
  print(i)
  i <- i + 2
  count <- count + 1
}
```

This code uses a while loop to increment the value of i by 2 for each iteration, until it prints the first 5 even numbers.

Control Flow Functions:

Control flow functions in R allow you to perform various control operations, such as breaking out of loops, skipping iterations, and restarting loops. The most common control flow functions are break, next, and return.

1) **Break statement:**

The break statement is used to break out of a loop prematurely:

```
for (i in 1:10)
{
  if (i == 5)
  {
    break
  }
  print(i)
}
```

This code loops through the sequence 1:10, and if the value of i is 5, it breaks out of the loop prematurely.

2) **next statement:**

The next statement is a control flow statement in R that allows you to skip to the next iteration of a loop without executing the remaining code in the current iteration. Here is an example of using the next statement in R:

```
for (i in 1:10)
{
  if (i %% 2 == 0)
  {
    next
  }
  print(i)
}
```

In this example, the loop iterates over the values of *i* from 1 to 10. Inside the loop, the `if` statement tests whether *i* is even (i.e., whether it is divisible by 2 using the modulus operator `%%`). If *i* is even, the next statement is executed, which skips to the next iteration of the loop without executing the `print(i)` statement. If *i* is odd, the `print(i)` statement is executed, which prints the value of *i*.

3) return statement:

The `return` statement is a control flow statement in R that allows you to terminate a function and return a value to the calling environment. When a `return` statement is encountered inside a function, the function immediately stops executing and returns the specified value to the calling environment. Here is an example of using the `return` statement in R:

```
sum_squared <- function(x, y)
{
  result <- x^2 + y^2
  return(result)
}
z <- sum_squared(3, 4)
print(z)
```

In this example, the `sum_squared` function takes two arguments (*x* and *y*) and returns the sum of their squares. Inside the function, the `result` variable is assigned the value of $x^2 + y^2$, and then the `return(result)` statement is executed to return the value of `result` to the calling environment. Outside the function, the value returned by `sum_squared(3, 4)` is assigned to the variable `z`, and then the `print(z)` statement is executed to print the value of `z`, which is 25 (i.e., the sum of 3^2 and 4^2).

Array:

An array is a data structure that allows you to store a collection of values of the same data type. An array can have one or more dimensions, and each dimension can have a specific length. To create an array in R, you can use the `array()` function, which has the following syntax:

```
array(data, dim, dimnames = NULL)
```

The `data` parameter specifies the values to be stored in the array, while the `dim` parameter specifies the dimensions of the array. The `dimnames` parameter is an optional argument that can be used to specify names for the dimensions.

Here is an example of how to create a simple 2-dimensional array in R:

```
# Create a 2x3 array
my_array <- array(c(1:6), dim = c(2, 3))

# Print the array
print(my_array)
```

This code creates a 2-dimensional array with 2 rows and 3 columns, and stores the numbers 1 through 6 in the array. The resulting output will look like this:

```
  [,1] [,2] [,3]
[1,]  1  3  5
[2,]  2  4  6
```

In R, arrays can be indexed using square brackets [], where each index corresponds to a specific dimension of the array. For example, to access the value at row 1, column 2 of the my_array array, you can use the following code:

```
value <- my_array[1, 2]
```

This code assigns the value 3 to the variable value, which corresponds to the value at row 1, column 2 of the array.

Matrix:

A matrix is a two-dimensional data structure that contains a collection of values of the same data type. The matrix can be thought of as a grid or table of values, where each row represents a separate observation or case, and each column represents a separate variable. To create a matrix in R, you can use the matrix() function, which has the following syntax:

```
matrix(data, nrow, ncol, byrow = FALSE, dimnames = NULL)
```

The data parameter specifies the values to be stored in the matrix, while the nrow and ncol parameters specify the number of rows and columns in the matrix, respectively. The byrow parameter is an optional argument that can be used to specify whether the values should be filled in by row or by column, and the dimnames parameter is an optional argument that can be used to specify names for the rows and columns. Here is an example of how to create a simple 2x3 matrix in R:

```
# Create a 2x3 matrix
my_matrix <- matrix(c(1, 2, 3, 4, 5, 6), nrow = 2, ncol = 3)
# Print the matrix
print(my_matrix)
```

This code creates a 2x3 matrix with the values 1 through 6, and stores the matrix in the variable my_matrix. The resulting output will look like this:

```
      [,1] [,2] [,3]
[1,]  1   3   5
[2,]  2   4   6
```

In R, matrices can be indexed using square brackets [], where the first index corresponds to the row number and the second index corresponds to the column number. For example, to access the value in row 1, column 2 of the my_matrix matrix, you can use the following code:

```
value <- my_matrix[1, 2]
```

This code assigns the value 3 to the variable value, which corresponds to the value at row 1, column 2 of the matrix.

Vectors:

A vector is a one-dimensional data structure that contains a collection of values of the same data type. The vector can be thought of as a simple list or array of values, and is often used to store a single variable or observation.

To create a vector in R, you can use the `c()` function, which is short for "combine". This function takes one or more values as arguments and combines them into a vector. Here is an example of how to create a simple vector in R:

```
# Create a vector  
my_vector <- c(1, 2, 3, 4, 5)  
# Print the vector  
print(my_vector)
```

This code creates a vector with the values 1 through 5, and stores the vector in the variable `my_vector`. The resulting output will look like this:

```
[1] 1 2 3 4 5
```

In R, vectors can be indexed using square brackets `[]`, where each index corresponds to a specific element of the vector. For example, to access the value at position 3 of the `my_vector` vector, you can use the following code:

```
value <- my_vector[3]
```

This code assigns the value 3 to the variable `value`, which corresponds to the third element of the vector.

Factors:

A factor is a data structure that represents categorical data. Factors are used to store data that can take on a limited number of possible values, such as nominal or ordinal data. Factors are useful in statistical analysis because they allow you to easily summarize and analyze categorical data.

To create a factor in R, you can use the `factor()` function, which takes one or more vectors as arguments and returns a factor object. Here is an example of how to create a simple factor in R:

```
# Create a factor  
my_factor <- factor(c("red", "green", "blue", "red", "green"))  
# Print the factor  
print(my_factor)
```

This code creates a factor with the values "red", "green", and "blue", and stores the factor in the variable `my_factor`. The resulting output will look like this:

```
[1] red green blue red green  
Levels: blue green red
```

In this example, the factor has three possible levels: "blue", "green", and "red". The levels of a factor are determined by the unique values in the vector that is used to create the factor.

You can also assign labels to the levels of a factor using the `levels()` function. For example:

```
# Assign labels to the levels of the factor  
levels(my_factor) <- c("R", "G", "B")  
# Print the factor with labels  
print(my_factor)
```

This code assigns the labels "R", "G", and "B" to the levels of the `my_factor` factor, and prints the factor with the labels.

R Packages:

A package is a collection of functions, data sets, and other resources that are designed to work together to solve a specific problem or accomplish a specific task. R packages are created and maintained by developers and are hosted on public repositories such as CRAN (Comprehensive R Archive Network) and GitHub.

To use a package in R, you first need to install it using the `install.packages()` function. For example, to install the `dplyr` package, which is a popular package for data manipulation, you can run the following code:

```
install.packages("dplyr")
```

Once the package is installed, you can load it into your R session using the `library()` function:

```
library(dplyr)
```

This makes all of the functions and data sets in the `dplyr` package available for use in your R code.

In addition to the base R functions and packages, there are thousands of third-party packages available for R that can be used for a wide range of tasks, including data manipulation, statistical analysis, machine learning, data visualization, and more. Some popular packages in R include `ggplot2`, `tidyr`, `caret`, `randomForest`, and `shiny`.

If you want to create your own package in R, you can use the `devtools` package to help you set up and manage the package. The `devtools` package provides functions for creating and building packages, adding documentation, and testing the package code.

Data frames:

A data frame is a two-dimensional array-like structure or a table in which a column contains values of one variable, and rows contains one set of values from each column. A data frame is a special case of the list in which each component has equal length.

A data frame is used to store data table and the vectors which are present in the form of a list in a data frame, are of equal length. In a simple way, it is a list of equal length vectors. A matrix can contain one type of data, but a data frame can contain different data types such as numeric, character, factor, etc.

There are following characteristics of a data frame.

- The columns name should be non-empty.
- The rows name should be unique.
- The data which is stored in a data frame can be a factor, numeric, or character type.
- Each column contains the same number of data items.

Data frames in R are usually created by importing data from external sources, such as CSV files or databases. They can also be constructed directly in R using functions like `data.frame()` or by converting other types of R objects like matrices or lists using functions like `as.data.frame()`.

Once a data frame is created, you can manipulate and analyze the data using a wide range of built-in functions and packages in R. Some common operations include selecting, filtering, and sorting rows and columns, calculating summary statistics, and creating visualizations.

Here is an example of creating a simple data frame in R:

```
# create a data frame
df<- data.frame
(
  name = c("Alice", "Bob", "Charlie"),
```



```

    age = c(25, 30, 35),
    gender = c("F", "M", "M")
)

# print the data frame
print(df)

```

This will create a data frame with three columns (name, age, gender) and three rows (one for each person), and print it to the console:

```

name age gender
1 Alice 25 F
2 Bob 30 M
3 Charlie 35 M

```

Data frame access:

In R, you can access and manipulate data frames using various methods. Here are some common ways to access data frames:

Consider the following data frame named df:

```

# Creating a sample data frame
df<- data.frame(
  Name = c("John", "Jane", "Alice", "Bob"),
  Age = c(25, 30, 35, 40),
  Country = c("USA", "Canada", "UK", "Australia")
)

```

Print the data frame

Output:

```

  Name Age Country
1 John 25 USA
2 Jane 30 Canada
3 Alice 35 UK
4 Bob 40 Australia

```

Accessing Columns:

You can access columns of a data frame using the dollar sign \$ or square brackets [] notation. Here's an example:

```

# Using the dollar sign notation
df$column_name
# Accessing the "Name" column using the dollar sign notation
df$Name

```

Output:

```
[1] "John" "Jane" "Alice" "Bob"
```

```

# Using the square brackets notation
df["column_name"]

```

```

# Accessing the "Country" column using the square brackets notation
df["Country"]

```

Output:

```
1 USA
2 Canada
3 UK
4 Australia
```

Accessing Rows:

You can access rows of a data frame using the square brackets [] notation. Here's an example:

```
# Accessing a single row
df[row_index, ]
```

```
# Accessing the second row
df[2, ]
```

Output:

```
  Name Age Country
2 Jane  30  Canada
```

```
# Accessing multiple rows
df[row_indices, ]
```

```
# Accessing multiple rows (3rd and 4th rows)
df[3:4, ]
```

Output:

```
  Name Age Country
3 Alice  35    UK
4  Bob  40 Australia
```

Accessing Specific Cells:

You can access individual cells in a data frame using the square brackets [] notation with row and column indices. Here's an example:

```
df[row_index, column_index]
```

```
# Accessing the cell in the third row and second column
df[3, 2]
```

Output:

```
[1] 35
```

Accessing Rows Based on Conditions:

You can use logical conditions to access rows based on specific criteria. Here's an example:

```
# Accessing rows where a condition is true
df[condition, ]
```

```
# Accessing rows where the age is greater than or equal to 30
df[df$Age >= 30, ]
```

Output:

```
Name Age Country
2 Jane 30 Canada
3 Alice 35 UK
4 Bob 40 Australia
```

Ordering data frames:

In R, you can order or sort a data frame based on specific columns using the `order()` function or the `dplyr` package's `arrange()` function. Let's explore both approaches with examples.

Using the `order()` function

Consider the following data frame `df`:

```
# Creating a sample data frame
df<- data.frame(
  Name = c("John", "Jane", "Alice", "Bob"),
  Age = c(25, 30, 35, 40),
  Country = c("USA", "Canada", "UK", "Australia")
)

# Print the data frame
df
```

Output:

```
Name Age Country
1 John 25 USA
2 Jane 30 Canada
3 Alice 35 UK
4 Bob 40 Australia
```

Now, let's order the data frame based on the `Age` column:

```
# Ordering the data frame by Age column
df_ordered <- df[order(df$Age), ]

# Print the ordered data frame
df_ordered
```

Output:

```
Name Age Country
1 John 25 USA
2 Jane 30 Canada
3 Alice 35 UK
4 Bob 40 Australia
```

The data frame `df_ordered` is now sorted based on the `Age` column in ascending order.

Using the `arrange()` function from `dplyr` package

First, make sure you have the `dplyr` package installed by running `install.packages("dplyr")`. Then, you can load the package and use the `arrange()` function:

```
# Loading the dplyr package
library(dplyr)

# Ordering the data frame by Age column using arrange()
df_ordered <- arrange(df, Age)

# Print the ordered data frame
df_ordered
```

Output:

```
  Name Age Country
1 John  25   USA
2 Jane  30 Canada
3 Alice 35   UK
4 Bob  40 Australia
```

Functions for data frames:

In R, there are several useful functions available for working with data frames. Here are some commonly used functions for data frames:

```
# Creating a sample data frame
df <- data.frame(
  Name = c("John", "Jane", "Alice", "Bob"),
  Age = c(25, 30, 35, 40),
  Country = c("USA", "Canada", "UK", "Australia")
)

# Print the data frame
df
```

Output:

```
  Name Age Country
1 John  25   USA
2 Jane  30 Canada
3 Alice 35   UK
4 Bob  40 Australia
```

- head(df) and tail(df): These functions display the first few rows (`head()`) or last few rows (`tail()`) of a data frame. By default, they show the first/last six rows, but you can specify a different number of rows if needed.

```
# Displaying the first few rows of the data frame
head(df)
```

Output:

```
  Name Age Country
1 John  25   USA
2 Jane  30 Canada
3 Alice 35   UK
```

```
# Displaying the last few rows of the data frame
tail(df)
```

Output:

```
Name Age Country
1 John 25 USA
2 Jane 30 Canada
3 Alice 35 UK
4 Bob 40 Australia
```

- nrow(df) and ncol(df): These functions return the number of rows (nrow()) or number of columns (ncol()) in a data frame.

nrow(df) and ncol(df):

```
# Getting the number of rows in the data frame
nrow(df)
```

Output:

```
[1] 4
```

```
# Getting the number of columns in the data frame
ncol(df)
```

Output:

```
[1] 3
```

- dim(df): This function returns a vector containing the dimensions of the data frame, i.e., the number of rows and columns.

```
# Getting the dimensions of the data frame
dim(df)
```

Output:

```
[1] 4 3
```

- names(df) and colnames(df): These functions return the column names of a data frame as a character vector.

```
# Getting the column names of the data frame
names(df)
```

Output:

```
[1] "Name" "Age" "Country"
```

- summary(df): This function provides a summary of the data frame, including descriptive statistics (e.g., minimum, maximum, mean) for each numeric column and frequency counts for each categorical column.

```
# Getting the summary of the data frame
summary(df)
```

Output:

```
Name      Age      Country
Length:4   Min.   :25.00 Length:4
Class :character 1st Qu.:27.50 Class :character
Mode :character Median :32.50 Mode :character
Mean  :32.50
```

3rd Qu.:37.50

Max. :40.00

- str(df): This function displays the structure of a data frame, including the variable names, data types, and the first few values of each variable.

```
str(df):  
# Displaying the structure of the data frame  
str(df)
```

Output:

```
'data.frame': 4 obs. of 3 variables:  
 $ Name : chr "John" "Jane" "Alice" "Bob"  
 $ Age : num 25 30 35 40  
 $ Country: chr "USA" "Canada" "UK" "Australia"
```

- subset(df, condition): This function allows you to subset a data frame based on a condition. You can specify the condition using logical expressions to filter rows based on specific criteria.

```
# Subsetting the data frame based on a condition  
subset(df, Age > 30)
```

Output:

```
  Name Age Country  
3 Alice 35    UK  
4  Bob 40 Australia
```

- unique(df\$column): This function returns the unique values in a specific column of a data frame.

```
# Getting the unique values in the "Country" column  
unique(df$Country)
```

Output:

```
[1] "USA"    "Canada" "UK"     "Australia"
```

- edit():

In R, the edit() function allows you to interactively edit the contents of a data frame. When you call edit() on a data frame, it opens a spreadsheet-like editor that allows you to make changes to the data directly. Here's an example of how to use the edit() function:

```
# Create a sample data frame  
df<- data.frame(  
  Name = c("Alice", "Bob", "Charlie"),  
  Age = c(25, 30, 35),  
  Gender = c("Female", "Male", "Male")  
)
```

```
# Open the data frame in the editor  
edit(df)
```

After executing the code, a separate window or tab will open with the data frame displayed in an editable format. You can modify the values, add or remove rows, or make any necessary changes to the data.

Value Added Course in Statistical Computing

Test Exercise

Answer all questions, each question carries 2 marks

1. Explain what is “R”?
2. What are the data structures in R that is used to perform statistical analyses and create graphs?
3. Write the example of creating vector in R?
4. Write the syntax of creating matrix in R?
5. Write the syntax of creating array in R?
6. Write the syntax of creating factor in R?
7. Example for creating Data frames in R?
8. Example statement for installing packages?
9. How to print dimensions of ‘df’ dataframe
10. Write the syntax for break statement

Key

1. R is data analysis software which is used by analysts, quants, statisticians, data scientists and others.
2. Vectors, Matrices, Arrays, Data frames
3. `my_vector <- c(1, 2, 3, 4, 5)`
4. `matrix(data, nrow, ncol, byrow = FALSE, dimnames = NULL)`
5. `array(data, dim, dimnames = NULL)`
6. `my_factor <- factor(c("red", "green", "blue", "red", "green"))`
7. `df <- data.frame`
(
 `name = c("Alice", "Bob", "Charlie"),`
 `age = c(25, 30, 35),`
 `gender = c("F", "M", "M")`
)
8. `install.packages("dplyr")`
9. `dim(df)`
10. `for (i in 1:10)`
{
 `if (i == 5)`
 {
 `break`
 }
 `print(i)`
}

A.G. & S.G. Siddhartha Degree College of Arts & Science

Vuyyuru-521165, Krishna District, Andhra Pradesh

Value Added Course - Attendance Register

Class / Section: II B.Sc (MSCS)

Year : 2022-23

Paper: Statistical Computing

Lecturer: T. Naga Prasada Rao

Sl. No	Roll No	Student Name	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	Total
1	2155301	K. Dharani	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	15
2	2155302	G. Divya	P	P	P	A	P	P	P	P	P	P	P	P	P	P	P	14
3	2155303	M. Sri Lakshmi	P	P	P	P	P	A	A	P	P	P	P	P	P	P	P	14
4	2155304	S. Sai Sowmya	P	P	P	P	P	P	P	A	A	P	P	P	P	P	P	14
5	2155305	A. Sushanth	P	P	P	P	P	P	P	P	P	P	A	P	P	P	P	14
6	2155306	B. Manoj phanindra	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	15
7	2155307	J. Keerthi priya	P	P	P	P	P	P	P	A	P	P	P	P	P	P	P	14
8	2155308	K. Alekhya	P	P	P	A	P	P	P	P	P	P	P	P	P	P	P	14
9	2155310	N. Anusha	P	P	P	P	P	P	P	A	P	P	P	P	P	P	P	14
10	2155311	A. Chakradhar	A	P	P	P	P	P	P	P	P	P	P	P	P	P	P	14
11	2155312	MD. Khadeera Begam	P	P	P	P	P	A	P	P	P	P	P	P	P	P	P	14
12	2155313	J. Jhansi lakshmi	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	15
13	2155315	K. Tulasi	P	P	P	A	P	P	P	P	P	P	P	P	P	P	P	14
14	2155316	P. Himasri	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	15
15	2155317	K. Naga Sravani	P	P	A	P	P	P	P	P	P	P	P	P	P	P	P	14
16	2155319	M. Karuna sri	P	P	P	P	P	A	P	P	P	P	P	P	P	P	P	14
17	2155320	P. Phani Supraja	P	P	P	P	P	P	A	P	P	P	P	P	P	P	P	14
18	2155321	K. Himasri	P	P	A	P	P	P	P	P	P	P	P	P	P	P	P	14
19	2155322	B. Durga Bhavani	P	P	P	P	P	P	P	P	A	P	P	P	P	P	P	14
20	2155323	D. Naga Gireesha	P	P	P	P	P	P	P	P	P	A	P	P	P	P	P	14

A.G. & S.G. Siddhartha Degree College of Arts & Science

Vuyyuru-521165, Krishna District, Andhra Pradesh

Value Added Course - Attendance Register

Class / Section: II B.Sc (MSCS)

Year : 2022-23

Paper: Statistical Computing

Lecturer: T. Naga Prasada Rao

Sl. No	Roll No	Student Name	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	Total
1	2155301	K. Dharani	P	P	P	P	P	P	P	A	P	P	P	P	P	P	P	14
2	2155302	G. Divya	A	P	P	P	P	P	P	P	P	P	P	P	P	P	P	14
3	2155303	M. Sri Lakshmi	P	P	P	P	P	P	P	P	A	P	P	P	P	P	P	14
4	2155304	S. Sai Sowmya	P	P	P	P	P	P	A	P	P	P	P	P	P	P	P	14
5	2155305	A. Sushanth	P	P	P	P	P	P	A	P	P	P	P	P	P	P	P	14
6	2155306	B. Manoj phanindra	P	P	P	P	P	P	P	P	P	P	A	P	P	P	P	14
7	2155307	J. Keerthi priya	P	P	P	P	P	P	A	P	P	P	P	P	P	P	P	14
8	2155308	K. Alekhya	P	P	P	P	P	P	P	P	P	P	P	A	P	P	P	14
9	2155310	N. Anusha	P	P	P	P	P	P	P	P	P	A	P	P	P	P	P	14
10	2155311	A. Chakradhar	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	15
11	2155312	MD. Khadeera Begam	P	P	P	P	P	P	P	P	P	A	P	P	P	P	P	14
12	2155314	J. Jhansi Lakshmi	P	P	P	A	P	P	P	P	P	P	P	P	P	P	P	14
13	2155315	K. Tulasi	P	P	P	P	P	P	P	A	P	P	P	P	P	P	P	14
14	2155316	P. Alima Sri	P	P	P	P	A	P	P	P	P	P	P	P	P	P	P	14
15	2155317	K. Naga sravani	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	15
16	2155319	M. Karuna Sri	P	P	P	A	P	P	P	P	P	P	P	P	P	P	P	14
17	2155320	P. Phani Supraja	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	15
18	2155321	K. Alima Sri	P	P	P	A	P	P	P	P	P	P	P	P	P	P	P	14
19	2155322	R. Durga Bhavani	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	15
20	2155323	P. Naga Gireesha	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	15



A.G. & S.G. Siddhartha Degree College

VUYURU - 521 165



Adusumilli Gopalakrishnaiah & Sugarcane Group
Siddhartha Degree College of Arts & Science,
Vuyyuru-521165, Krishna District

A.G. & S.G. Siddhartha Degree College of Arts & Science
Vuyyuru-521165, Krishna District, Andhra Pradesh

Department of Computer Science

Value Added Course in Statistical Computing
Feed Back Form

1. Is the programme interested to you (Yes/No)
2. Have you attended all the session (Yes/No)
3. Is the content of the program is adequate (Yes/No)
4. Have the teacher covered the entire syllabus? (Yes/No)
5. Is the number of hours adequate? (Yes/No)
6. Do you have any suggestions for enhancing or reducing the number of weeks designed for the program? (Yes/No)
7. On the whole, is the program useful in terms of enriching your knowledge? (Yes/No)
8. Do you have any suggestions on the program? (Yes/No)



Head of the Department of Computer Science
A.G. & S.G. Siddhartha Degree College
VUYYURU - 521 165

K. Hema Sar
2155321



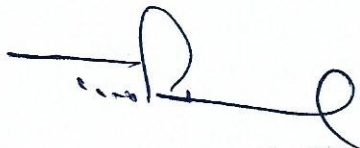
Head of the Department of Computer Science
A.G. & S.G. Siddhartha Degree College of Arts & Science,
Vuyyuru-521 165, Krishna District.

A.G. & S.G. Siddhartha Degree College of Arts & Science
Vuyyuru-521165, Krishna District, Andhra Pradesh

Department of Computer Science

Value Added Course in Statistical Computing
Feed Back Form

1. Is the programme interested to you (Yes/No)
2. Have you attended all the session (Yes/No)
3. Is the content of the program is adequate (Yes/No)
4. Have the teacher covered the entire syllabus? (Yes/No)
5. Is the number of hours adequate? (Yes/No)
6. Do you have any suggestions for enhancing or reducing the number of weeks designed for the program? (Yes/No)
7. On the whole, is the program useful in terms of enriching your knowledge? (Yes/No)
8. Do you have any suggestions on the program? (Yes/No)



Head of the Department of Computer Science
A.G. & S.G. Siddhartha Degree College
VUYYURU - 521 165

K. Tulasi

2155315



Principal
Adusumilli Gopalakrishnaiah & Sugarcane Growers
Siddhartha Degree College of Arts & Science,
Vuyyuru-521 165, Krishna District.

A.G. & S.G. Siddhartha Degree College of Arts & Science
Vuyyuru-521165, Krishna District, Andhra Pradesh

Department of Computer Science

Value Added Course in Statistical Computing
Feed Back Form

1. Is the programme interested to you (Yes/No)
2. Have you attended all the session (Yes/No)
3. Is the content of the program is adequate (Yes/No)
4. Have the teacher covered the entire syllabus? (Yes/No)
5. Is the number of hours adequate? (Yes/No)
6. Do you have any suggestions for enhancing or reducing the number of weeks designed for the program? (Yes/No)
7. On the whole, is the program useful in terms of enriching your knowledge? (Yes/No)
8. Do you have any suggestions on the program? (Yes/No)



NAME OF THE DEPARTMENT OF STUDENT,
A.G. & S.G. Siddhartha Degree College,
VUYYURU - 521 165

A. Sushanth
2155305



Principal
Adusumilli Gopalakrishnaiah & Sugarcane Grower-
Siddhartha Degree College of Arts & Science,
Vuyyuru-521 165, Krishna District.

A.G. & S.G. Siddhartha Degree College of Arts & Science
Vuyyuru-521165, Krishna District, Andhra Pradesh

Department of Computer Science

Value Added Course in Statistical Computing

Marks List

Class: II B.Sc (MSCS)

S. No	Roll No.	Name of the Student	Marks
1	2155301	K. Dharani	20
2	2155302	G. Divya	18
3	2155303	M. Sri Lakshmi	18
4	2155304	S. Sai Sowmya	16
5	2155305	A. Sushanth	20
6	2155306	B. Manoj Phanindra	18
7	2155307	J. Keerthi Priya	18
8	2155308	K. Alekhya	16
9	2155310	N. Anusha	18
10	2155311	A. Chakradhar	20
11	2155312	MD. Khadeera Begam	18
12	2155314	J. Jhansi Lakshmi	18
13	2155315	K. Tulasi	16
14	2155316	P. Hima Sri	16
15	2155317	K. Naga Sravani	18
16	2155319	M. Karuna Sri	20
17	2155320	P. Phani Supraja	20
18	2155321	K. Hema Sri	16
19	2155322	R. Durga Bhavani	18
20	2155323	D. Naga Gireesha	16



ADUSUMILLI GOPALAKRISHNAIAH AND SUGARCANE GROWERS
SIDDHARTHA DEGREE COLLEGE OF ARTS AND SCIENCE,
(AUTONOMOUS) VUYYURU A.P
(Accredited at "A" level by NAAC, Bengaluru)



Department of Computer Science


VALUE ADDED COURSE: Statistical Computing

CERTIFICATE

This is to Certify that *G. Divya*.....of *T.B.S.A.(M.S.A.S.)*... has successfully completed Value Added Course in **Statistical Computing** organised by the Department of Computer Science during the Year 2022-2023 and passed the Examination in grade....*A*....

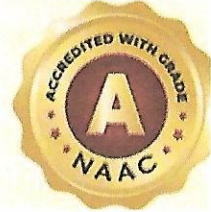

Co-ordinator


Head of Department
of the Department of Computer Science
A & S Siddhartha Degree College
VUYYURU - 521 161


Principal
Principal
Adusumilli Gopalakrishnaiah & Sugarcane Growers
Siddhartha Degree College of Arts & Science,
Vuyyuru-521 161, Krishna District.



ADUSUMILLI GOPALAKRISHNAIAH AND SUGARCANE GROWERS
SIDDHARTHA DEGREE COLLEGE OF ARTS AND SCIENCE,
(AUTONOMOUS) VUYYURU A.P
(Accredited at "A" level by NAAC, Bengaluru)



Department of Computer Science

VALUE ADDED COURSE: Statistical Computing

CERTIFICATE

This is to Certify that *K. Hema Sri*.....of *II Bsc (M.Sc.S)*..... has successfully completed Value Added Course in **Statistical Computing** organised by the Department of Computer Science during the Year 2022-2023 and passed the Examination in grade....*'A'*....

Co-ordinator

Head of Department

Head of the Department of

A. G. & S. G. Siddhartha Degree

VUYYURU

Principal

Principal

Adusumilli Gopalakrishnaiah & Sugarcane Growers

Siddhartha Degree College of Arts & Science

Vuyyuru-521 165, Krishna District