1. **List the data types in R.**
   - **Numeric:** Represents numeric values (e.g., 1.5, -3.2).
   - **Integer:** Represents integer values (e.g., 1L, -3L).
   - **Logical:** Represents Boolean values (TRUE or FALSE).
   - **Character:** Represents text or string values.
   - **Complex:** Represents complex numbers with real and imaginary parts.
   - **Raw:** Represents raw byte values.

2. **What is the use of tapply() and get() functions.**
   1. **tapply():** Used to apply a function over subsets of a vector or data frame, split by one or more factors.
   # Example: Applying mean function to subsets based on a factor
   tapply(mtcars$mpg, mtcars$cyl, mean)

   2. **get():** Used to retrieve the value of an object with a specified name.
   # Example: Retrieving the value of a variable with the name 'x'
   x <- 5
   get("x")

3. **What is the use of readline() function.**
   Used to read a line from the console input. It is often used to interactively input data from the user.

   # Example: Reading a line of text from the user
   user_input <- readline(prompt = "Enter your name: ")

4. **Give the example for lowess() function .**
   Computes a locally weighted scatterplot smoothing (LOWESS) fit.

   # Example: Generating a LOWESS smooth curve
   x <- 1:50
   y <- rnorm(50)
   smooth_curve <- lowess(x, y)
   plot(x, y)
   lines(smooth_curve)

5. **Define correlation function.**
   Measures the degree of association between two variables. In R, cor() is commonly used.

   # Example: Calculating correlation between two vectors
   x <- c(1, 2, 3, 4, 5)
   y <- c(2, 4, 1, 3, 5)
   correlation <- cor(x, y)

6. **What is multiple regression?**
   A statistical technique that models the relationship between multiple independent variables and a dependent variable.

   # Example: Multiple regression with lm() function
   model <- lm(y ~ x1 + x2, data = mydata)

7. **What is interactive mode?**
   Refers to running R code interactively, where commands are entered one at a time, and the results are immediately displayed.

8. **What is batch mode?**
   Refers to running R code as a script or from a file without user interaction. The entire script is executed without user intervention.

9. **Give example for cumulative sums and products.**
   cumsum() and cumprod() functions are used for cumulative sums and products, respectively.

   # Example: Cumulative sums and products
   x <- c(1, 2, 3, 4)
   cum_sum <- cumsum(x) #1 3 6 10
   cum_prod <- cumprod(x) #1 2 6 24

10. **What is the use of point() function.**
    There is no standard function named point() in base R. If you refer to a specific package or context, please provide more details.

11. **Define poisson distribution.**
    A probability distribution that describes the number of events occurring within fixed intervals of time or space. In R, it can be simulated using functions like dpois() for probability density, ppois() for cumulative distribution, etc.

12. **min() vs pmin() function.**
    min() returns the minimum value among a set of values.
    pmin() returns the element-wise minimum of several vectors.

    # Example:
    min_value <- min(3, 7, 1, 9) #1
    pmin_values <- pmin(c(1, 5, 3), c(2, 4, 6)) #1 43

13. **Define binomial distribution.**

A probability distribution that describes the number of successes in a fixed number of independent Bernoulli trials. In R, it can be simulated using functions like dbinom() for probability density, pbinom() for cumulative distribution, etc.

14. What is spline?

A piecewise continuous polynomial function used for interpolation or smoothing. In R, the spline() function can be used to generate spline fits.

 # Example:

```
x <- 1:10
y <- c(3, 1, 4, 1, 5, 9, 2, 6, 5, 3)
spline_fit <- spline(x, y, n = 100, method = "natural")
plot(x, y)
lines(spline_fit, col = "red")
```

1. What is R, and what are its main characteristics?

R is a programming language and environment widely used for solving data science problems and particularly designed for statistical computing and data visualization.

Its main characteristics include:

- Open source
- Interpreted (i.e., it supports both functional and object-oriented programming)
- Highly extensible due to its large collection of data science packages
- Functional and flexible
- Compatible with many operating systems
- Can be easily integrated with other programming languages and frameworks
- Allows powerful statistical computing
- Offers a variety of data visualization tools for creating publication-quality charts
- Equipped with the command-line interface
- Supported by a strong online community

2. What are some disadvantages of using R?

- Non-intuitive syntax and hence a steep learning curve, especially for beginners in programming
- Relatively slow
- Inefficient memory usage
- Inconsistent and often hard-to-read documentation of packages
- Some packages are of low quality or poorly-maintained
- Potential security concerns due to its open-source nature

3. List and define some basic data types in R.

1. **Numeric** : decimal numbers.
2. **Integer:** whole numbers.
3. **Character**: a letter, number, or symbol, or any combination of them, enclosed in regular or single quotation marks.
4. **Factor**: categories from a predefined set of possible values, often with an intrinsic order.
5. **Logical**: the Boolean values TRUE and FALSE, represented under the hood as 1 and 0, respectively.

4. List and define some basic data structures in R.

1. **Vector :** a one-dimensional data structure used for storing values of the same data type.
2. **List:** a multi-dimensional data structure used for storing values of any data type and/or other data structures.
3. **Matrix:** a two-dimensional data structure used for storing values of the same data type.
4. **Data frame:** a two-dimensional data structure used for storing values of any data type, but each column must store values of the same data type.

5. How to import data in R?

The base R provides essential functions for importing data:

- **read.table():** the most general function of the base R for importing data, takes in tabular data with any kind of field separators, including specific ones, such as |.
- **read.csv():** comma-separated values (CSV) files with . as the decimal separator.
- **read.csv2():** semicolon-separated values files with , as the decimal separator.
- **read.delim():** tab-separated values (TSV) files with . as the decimal separator.

6. What is a package in R?

An R package is a collection of functions, code, data, and documentation, representing an extension of the R programming language and designed for solving specific kinds of tasks.

7. What is a factor in R?

A factor in R is a specific data type that accepts categories (aka levels) from a predefined set of possible values. These categories look like characters, but under the hood, they are stored as integers. Often, such categories have an intrinsic order.

8. What is RStudio?

RStudio is an open-source IDE (integrated development environment) that is widely used as a graphical front-end for working with the R programming language starting from version 3.0.1.

9. How to create a user-defined function in R?

```
function_name <- function(parameters){
   function body
}
```

- **Function name:** the name of the function object that will be used for calling the function after its definition.
- **Function parameters:** the variables separated with a comma and placed inside the parentheses that will be set to actual argument values each time we call the function.
- **Function body:** a chunk of code in the curly brackets containing the operations to be performed in a predefined order on the input arguments each time we call the function.

10. List some popular data visualization packages in R.

- **ggplot2:** the most popular R data visualization package allowing the creation of a wide variety of plots.
- **Lattice:** for displaying multivariate data as a tiled panel (trellis) of several plots.
- **Plotly:** for creating interactive, publication-quality charts.
- **Highcharter:** for easy dynamic plotting, offers many flexible features, plugins, and themes; allows charting different R objects with one function.
- **Leaflet:**for creating interactive maps.

11. How to assign a value to a variable in R?

1. Using the assignment operator <-, e.g., my_var <- 1—the most common way of assigning a value to a variable in R.
2. Using the equal operator =, e.g., my_var = 1—for assigning values to arguments inside a function definition.
3. Using the rightward assignment operator ->, e.g., my_var -> 1—can be used in pipes.
4. Using the global assignment operators, either leftward (<<-) or rightward (->>), e.g., my_var <<- 1—for creating a global variable inside a function definition.

12. What are the requirements for naming variables in R?

1. A variable name can be a combination of letters, digits, dots, and underscores. It can't contain any other symbols, including white spaces.
2. A variable name must start with a letter or a dot.
3. If a variable name starts with a dot, this dot can't be followed by a digit.
4. Reserved words in R (TRUE, for, NULL, etc.) can't be used as variable names.
5. Variable names are case-sensitive.

.

13. What types of loops exist in R, and what is the syntax of each type?

1. For loop: iterates over a sequence the number of times equal to its length (unless the statements break and/or next are used) and performs the same set of operations on each item of that sequence.

```
for (variable in sequence) {
   operations
}
```

2. While loop: performs the same set of operations until a predefined logical condition (or several logical conditions) is met—unless the statements break and/or next are used.

```
while (logical condition) {
   operations
   variable update
}
```

3. Repeat loop: repeatedly performs the same set of operations until a predefined break condition (or several break conditions) is met.

```
repeat {
   operations
   if(break condition) {
      break
   }
}
```

14. How to aggregate data in R?

To aggregate data in R, we use the aggregate() function. This function has the following

essential parameters, in this order:

- **x**:the data frame to aggregate.
- **by:**a list of the factors to group by.
- **FUN**:an aggregate function to compute the summary statistics for each group (e.g., mean, max, min, count, sum).

15. What is the difference between the functions apply(), lapply(), sapply(), and tapply()?

- While all these functions allow iterating over a data structure without using loops and perform the same operation on each element of it, they are different in terms of the type of input and output and the function they perform.
- **apply():** takes in a data frame, a matrix, or an array and returns a vector, a list, a matrix, or an array. This function can be applied row-wise, column-wise, or both.
- **lapply():** takes in a vector, a list, or a data frame and always returns a list. In the case of a data frame as an input, this function is applied only column-wise.
- **sapply():** takes in a vector, a list, or a data frame and returns the most simplified data structure, i.e., a vector for an input vector, a list for an input list, and a matrix for an input data frame.
- **tapply():** calculates summary statistics for different factors (i.e., categorical data).

16. List and define the control statements in R.

There are three groups of control statements in R: conditional statements, loop statements, and jump statements.

Conditional statements:

- **if**—tests whether a given condition is true and provides operations to perform if it's so.
- **if-else**—tests whether a given condition is true, provides operations to perform if it's so and another set of operations to perform in the opposite case.
- **if... else if**... else—tests a series of conditions one by one, provides operations to perform for each condition if it's true, and a fallback set of operations to perform if none of those conditions is true.
- **switch**—evaluates an expression against the items of a list and returns a value from the list based on the results of this evaluation.

Loop statements:

- **for**—in for loops, iterates over a sequence.
- **while**—in while loops, checks if a predefined logical condition (or several logical conditions) is met at the current iteration.
- **repeat**—in repeat loops, continues performing the same set of operations until a predefined break condition (or several break conditions) is met.

Jump statements:

- **next**—skips a particular iteration of a loop and jumps to the next one if a certain condition is met.
- **break**—stops and exits the loop at a particular iteration if a certain condition is met.
- **return**—exits a function and returns the result.

17. What are correlation and covariance, and how do you calculate them in R

Covariance measures the degree to which two variables change together, while correlation is a standardized measure of covariance that ranges from -1 to 1, indicating the strength and direction of the relationship.