Guideline Book

Deepaneesh R V

March 17, 2025

Contents

1 HYOTHESIS FOR TEST

In general, a hypothesis is a testable statement or assumption about a relationship between two or more variables. It serves as the foundation for research and statistical analysis, guiding experiments and observations.

Key Characteristics of a Hypothesis:

- **Testable** – It should be possible to confirm or refute the hypothesis using data or experiments.
- **Falsifiable** – There should be a possibility of proving it wrong.
- **Specific** – It should clearly define variables and their expected relationship.
- **Based on Prior Knowledge** – A hypothesis is often formed based on previous research, theories, or observations.

Types of Hypotheses:

Null Hypothesis (H_0): Assumes no effect or no relationship between variables

Alternative Hypothesis (H_1): Suggests a relationship or effect exists

In research, hypothesis testing is conducted to determine if there is enough statistical evidence to reject the null hypothesis in favor of the alternative hypothesis

A **simple hypothesis** and a **composite hypothesis** are two types of statistical hypotheses used in hypothesis testing.

Simple Hypothesis

A **simple hypothesis** specifies a single value for a parameter. It gives a precise statement about the population.

Example:

- **Null Hypothesis** (H_0): The average height of students is exactly 50 cm.

$$H_0 : \mu = 50$$

- **Alternative Hypothesis** (H_1): The average height is different from 50 cm.

$$H_1 : \mu \neq 50$$

Composite Hypothesis

A **composite hypothesis** does not specify a single value but rather a range or set of possible values for a parameter.

Example:

- **Null Hypothesis** (H_0): The average height is at least 50 cm.

$$H_0 : \mu \geq 50$$

- **Alternative Hypothesis** (H_1): The average height is less than 50 cm.

$$H_1 : \mu < 50$$

Here, the null hypothesis is **composite** because it includes multiple values (μ can be 50, 51, 52, etc.).

2 R PROGRAMMING

R is a programming language and software environment for statistical computing and graphics. It is widely used among statisticians and data miners for developing statistical software and data analysis. Use of a software is desirable and moreover an essential part of any analysis. It supports many free packages which helps the data scientist and analyst.It is a **free software**.

R Programming is Developed by **Ross Ihaka and Robert Gentleman** in 1993. It is dialect of **S language.** which was developed at Bell Laboratories by **John Chambers** and colleagues.

what is R ?

- R is an environment for data manipulation, statistical computing, graphics display and data analysis.
- Effective data handling and storage of outputs is possible.
- Simple as well as complicated calculations are possible.
- Simulations are possible.
- Graphical display on-screen and hard copy are possible.
- Programming language is effective which includes all possibilities just like any other good programming language.
- R has a statistical computing environment.
- It has a computer language which is convenient to use for statistical and graphical applications.

Installing R

- **Download the R software** from the Comprehensive R Archive Network (CRAN) website:**https://cran.r-project.org/**

Working with R

- Use commandline to type and execute the commands
- Some free software like **R Studio, Tinn R** etc. are also available to work with R software. It is an interface between R and us .
- Such software are more useful for beginners.
- It makes coding and execution of programmes easier
- **RStudio** is available at **https://www.rstudio.com/**
- **Tinn R** is available at **https://sourceforge.net/projects/tinn-r/**

Features of R Programming :

- Free to Run and Use
- Free to Study
- Free to Share
- Free to update

2.1 R Operators

There are several types of operators in R programming. They are :

- Assignment Operators
- Arithmetic Operators
- Relational Operators

- Logical Operators
- Miscellaneous Operators

Assignment Operators

```
<-  --> Leftward Assignment
->  --> Rightward Assignment
->> --> Rightward Assignment
=   --> Assignment
```

Example for Assignment Operators:

1. **Leftward Assignment:**

```
a <- 10
a1 <<- 100
print(c(a,a1))
```

```
## [1]  10 100
```

2. **Rightward Assignment:**

```
11 -> b
111 ->> b1
print(c(b,b1))
```

```
## [1]  11 111
```

3. **Equal Assignment:**

```
c = 12
print(c)
```

```
## [1] 12
```

Arithmetic Operators

```
+ -> Addition
- -> Subtraction
* -> Multiplication
/ -> Division
** or ^ -> For power or Exponential
%% -> Modulus
%/% -> Integer Division
```

Example for Arithmetic Operators:

```
1+1 # Addition
```

```
## [1] 2
```

```
2-1 # Subtraction
```

```
## [1] 1
```

```
3*2 # Multiplication
```

```
## [1] 6
```

```
4/2 # Division
```

```
## [1] 2
```

```r
5^2 # Exponential
```

```
## [1] 25
```

```r
5**2 # Exponential
```

```
## [1] 25
```

```r
5%%2 # Modulus (the remainder will be the result)
```

```
## [1] 1
```

```r
5%/%2 # Integer Division
```

```
## [1] 2
```

Relational Operators

The Result of Relational Operators is always a **Boolean Value** (True or False)

```
<  --> Less than
>  --> Greater than
<= --> Less than or equal to
>= --> Greater than or equal to
== --> Equal to
!= --> Not Equal to
```

Example for Relational Operators:

```r
a = 10
b=20
print(c(a,b))
```

```
## [1] 10 20
```

```r
a<b
```

```
## [1] TRUE
```

```r
a>b
```

```
## [1] FALSE
```

```r
a<=b
```

```
## [1] TRUE
```

```r
a>=b
```

```
## [1] FALSE
```

```r
a==b
```

```
## [1] FALSE
```

```r
a!=b
```

```
## [1] TRUE
```

Logical Operators:

Logical operators are used to combine conditional statements. it returns a boolean value (True or False)

```
& -> AND
&& -> Logical And
| ->OR
|| -> Logical OR
! -> NOT
```

Example for Logical Operators:

```
a = 10
b=20
print(c(a,b))
```

```
## [1] 10 20
```

```
# And Operator
# The Statement both should be true to get the result as true
# Any one of the statement is false then the result will be #false
a==10 & b==20
```

```
## [1] TRUE
```

```
a==10 & b==30
```

```
## [1] FALSE
```

```
# OR Operator
# The Statement any one should be true to get the result as #true
# Both the statement is false then the result will be false
a==10 | b==30
```

```
## [1] TRUE
```

```
a==20 | b==30
```

```
## [1] FALSE
```

```
# NOT Operator
# The Statement should be false to get the result as true
# The Statement should be true to get the result as false
a!=10
```

```
## [1] FALSE
```

```
a!=20
```

```
## [1] TRUE
```

Miscellaneous Operators

In R, miscellaneous operators are special operators that don't fall into typical arithmetic, logical, or relational categories. Some important ones include:

```
 x <- c(1, 2, 3, 4, 5)
y <- c(2, 4, 6)
x %in% y
```

```
## [1] FALSE  TRUE FALSE  TRUE FALSE
```

```
1:5 # Sequence from 1 to 5
```

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

```r
seq(1,10,by=2) # With Difference of 2
```

```
## [1] 1 3 5 7 9
```

```r
seq(1,10,length=3) # the output should be 3 numbers between 1 to 10
```

```
## [1]  1.0  5.5 10.0
```

```r
 # $
df <- data.frame(Name = c("A", "B"), Age = c(25, 30))
df$Name
```

```
## [1] "A" "B"
```

2.2 Data types in R

Data Types in R , data types are classified into two type.

- Object
- Attributes

There are 5 basic data types in Object:

1. Numeric
2. Integer
3. Complex
4. Logical
5. Character

There are 6 basic data types in Attributes:

1. Vectors
2. Factors
3. Lists
4. Arrays / Matrics
5. Data Frames
6. Class

We can use `Typeof()` function to check the data type of the object. The Result for typeof function will be

```
integer -> integer
character -> Character
logical -> Logical
numeric -> Double
complex -> Complex
Factor -> Factor
```

```r
 # $
a=10
typeof(a)
```

```
## [1] "double"
```

we can use as. , is. functions to check and change the data type

```
# is function is used to check the data type
# if the data type is correct then it will return as true otherwise False
is.numeric()
is.integer()
is.character()
is.logical()
is.complex()
is.factor()
is.vector()
is.list()
is.matrix()
is.data.frame()
is.array()
```

Like this there are more functions available in R to check the data type of the object.

```
# as function is used to change the data type
as.numeric() # To change the data type to numeric
as.integer() # To change the data type to integer
as.character() # To change the data type to character
as.logical() # To change the data type to logical
as.matrix() # To change the data type to matrix
as.data.frame() # To change the data type to data frame
as.vector() # To change the data type to vector
as.factor() # To change the data type to factor
```

2.3 R for calculations

We can use R as a basic calculator to perform arithmetic operations. For example, we can add, subtract, multiply, and divide numbers using R.

Multiple operators (BODMAS):

```
 6+5-4*3+2/4
```

```
## [1] -0.5
```

Concatenate:

Still now we have seen that using single value for the operation. But we can also use multiple values for the operation. by using concatenate.

```
x=c(1,2,3,4.......)
allow us to use multiple values for the operation
```

While using concatenate we can use multiple values but in a same data type

```
 x=c(1,2,3,4) # Multiple values
print(x)
```

```
## [1] 1 2 3 4
```

Some Basic Commands in R:

```
 rm(x) # To remove the variable
```

```
q() # To quit the R
```

: The character # marks the beginning of a commend

```
# this is command x=12
```

Power operators with vector versus scalar

```
c(3,4,5,6)^2 # Power operator with vector
```

```
## [1]  9 16 25 36
```

```
c(3,4,5,6)^c(2,3) # Power operator with vector
```

```
## [1]   9  64  25 216
```

```
c(2,3,4,5)^c(3,4,5) # Power operator with vector with warning
```

```
## Warning in c(2, 3, 4, 5)^c(3, 4, 5): longer object length is not a multiple of
## shorter object length
```

```
## [1]    8   81 1024  125
```

in general warning and error are different

- **Warning** is a message that is displayed when the code is executed
- **Error** is a message that is displayed when the code is not executed

Multiplication with vector versus scalar

```
c(2,3,4,5) * 6
```

```
## [1] 12 18 24 30
```

```
c(2,3,4,5) * c(-2,-3,-4,6)
```

```
## [1]  -4  -9 -16  30
```

```
c(2,3,4,5) * c(1,2)
```

```
## [1]  2  6  4 10
```

```
 c(2,3,4,5) * c(6,7,8) # Warning message
```

```
## Warning in c(2, 3, 4, 5) * c(6, 7, 8): longer object length is not a multiple
## of shorter object length
```

```
## [1] 12 21 32 30
```

Addition with vector versus scalar

```
c(2,3,4,5) + 20
```

```
## [1] 22 23 24 25
```

```
c(2, 3, 4, 5) + c(6, 7)
```

```
## [1]  8 10 10 12
```

```
c(2,3,4,5) + c(6,7,8) # with warning message
```

```
## Warning in c(2, 3, 4, 5) + c(6, 7, 8): longer object length is not a multiple
## of shorter object length
```

```
## [1]  8 10 12 11
```

Some Basic functions in R: To get the maximum and minimum value from the vector

```
max(4.2, 3.7, -2.3, 4)
```

```
## [1] 4.2
```

```
min(4.2, 3.7, -2.3, 4)
```

```
## [1] -2.3
```

```
x=c(1,2,3,4,5)
max(x)
```

```
## [1] 5
```

```
min(x)
```

```
## [1] 1
```

some more basic functions in R

```
abs()      # to get the absolute Values
sqrt()     # to get the square root
round()    # to get the round value
floor()    # to get the floor value
ceiling()  # to get the ceiling value
sum()      # to get the sum of the values
prod()     # to get the product of the values
mean()     # to get the average of the values
log()      # to get the log value
exp()      # to get the exponential value
length()   # to get the length of the vector
```

Trigonometric functions

```
sin()  # to get the sin value
cos()  # to get the cos value
tan()  # to get the tan value
asin() # to get the cosec value
acos() # to get the sec value
atan() # to get the cot value
```

Hyperbolic functions

```
sinh()  # to get the sinh value
cosh()  # to get the cosh value
tanh()  # to get the tanh value
asinh() # to get the cosech value
acosh() # to get the sech value
atanh() # to get the coth value
```

An assignment can also be used to save values in variables:

```
x = c(2,3,4,5)
y = x^2
print(y)
```

```
## [1]  4  9 16 25
```

3 GIT & GITHUB

3.1 Git / GitHub

Git Terminology Features and Advantages of Git TerminologyGit is a free, open-source **version control system (VCS)** that helps manage source code and its development history. It's the most widely used VCS in the world

GitHub is a website that helps developers store, manage, and collaborate on software projects.It's a social network for programmers that encourages collaboration and sharing of code . Github is an online platform that allows you to store remote repositories of your projects (Interactable)

3.2 Source Control / Version Control

- Some System used for tracking your file progress over time .
- It is usually saved in a series of snapshots and branches, which you can move back and forth between.
- Prevent against data loss / damage by Creating backup copies of your work.

Linux is the major / huge project in github Git is a Source Control Software . It was created by the same person who created Linux **Linus Torvalds**

3.3 Command Prompt

Before using git/github we need to have a basic idea about command prompt . Command prompt is a command line interpreter application available in most Windows operating systems. It's used to execute entered commands. Most of those commands automate tasks via scripts and batch files, perform advanced administrative functions, and troubleshoot or solve certain kinds of Windows issues.

Some of the basic commands prompt codes are :

```
E: -> To change the drive to E
cd -> To change the directory

# some time we have space in the directory name so we need to use double quotes
cd "C:\Users\Deepaneesh R V\Documents" -> To change the directory to Documents
cd.. -> To go back to the previous directory
cd/ -> To go to the root directory(Home)
cd/mnt/E -> To change the directory to E drive
dir ->To list the files in the directory
```

Folder Creation

Their is a command to create a new directory(Folder) and to remove a directory(Folder)

like:

```
mkdir -> To create a new directory(Folder)
rmdir -> To remove a directory( Folder)
```

File Creation

Their is a command to create a new file and to remove a file

like:

```
echo "Hello" > hello.txt |--> To create a new file
type null > hello.txt |--> To create a new file
copy con hello.txt |--> To create a new file
del hello.txt |--> To remove a file
del . |--> To remove all the files in the directory
```

3.4 Github Commands Repository

A repository is a storage space where your project lives. It can be local to a folder on your computer or it can be a storage space on GitHub or another online platform.

```
local repository -> Repository in your computer
Remote repository -> Repository in the cloud (Github)
```

Respositories are the timelines of the entire project including the previous changes

Git Commands

Before using git we need to install git in our system. then you need to configure the git with your username and email id. Code for Configuring the git (Only for the first time) like Registering the git with your username and email id

```
git config --global user.name "Username"
git config --global user.email "Email id"
```

Flow Chart of Github

```
Untracked -> Staged -> Committed -> Uploading
          |         |            |
        (add)    (Commit)      (Push)
```

This is the flow chart of github

Git basic Command

```
git init                      |---> initializing the git
git add .                     |---> add everything
git commit -m "first commit" |---> commit
git remote add origin git@github.com:username/<reponame>.git |--->choosing the git website
git push -u origin master   |---> pushing the changes to the master
```

Git Commit Commands

Commit takes the snapshot of all the changes / the modifications that you have done at the time and store them in a tree like structure. It is like a save point in a game.

```
git status                  |---> To check the status of the git
git commit -m "message"     |---> To commit the changes
git commit --help           |---> To get the help  the commit
git log                     |---> To show the history of the commit
```

as we know that git stores all the changes in a tree like structure so we can use the git log to see the history of the commit . some of the git commands allow su to move back and forth between the commit history

There are some git commands which allow us to move back and forth between the commits temporarily and permanently

```
git checkout <commit id>  |---> To move to the commit id
git checkout master       |---> To move to the master
```

this is the command to move back and forth between the commits as temporary

there are some commands which allow us to move back and forth between the commits as permanent.

```
git revert <commit id>    |---> To move to the commit id one step back
```

There are some commands which allow us to move back and forth between the commits as permanent with more than one step back . there is no return to future

that is git reset command

there are three types of reset commands:

1. soft
2. mixed
3. hard

git reset commands allow us to move to the commit id then make that as a master

```
git reset --hard <commit id> |---> To move to the commit id and make that as master
```