Essential R Commands for Statistical Work

1. Basic R Commands

Purpose	Command	Example
Check working directory	`getwd()`	
Set working directory	`setwd("path/to/folder")`	
List objects in memory	`ls()`	
Remove object	`rm(x)`	
Help for a function	`?mean` or `help(mean)`	
Install a package	`install.packages("dplyr")`	
Load a package	`library(dplyr)`	

2. Syntax and Working with Functions

Purpose	Command / Example	Description
Assignment	`x <- 10`	Create a variable
Comment	`# This is a comment`	
Vector	`x <- c(1, 2, 3, 4)`	Basic data structure
Sequence	`1:10` or `seq(1, 10, by=2)`	Generate sequences
Repeat values	`rep(5, times=3)`	Result: 5 5 5
Vector of 1000 of 1	`x <- rep(1, times = 1000)`	
Check object type	`class(x)` or `typeof(x)`	
Length of object	`length(x)`	
Conditional expression	`if (x > 5) {print("yes")} else {print("no")}`	
Loop	`for (i in 1:5) print(i)`	
Apply function to vector	`sapply(x, sqrt)`	

Creating custom functions:

```
my_function <- function(a, b = 2) {
  result <- a * b
  return(result)
}

my_function(3)  # 6
```

3. Basic Statistical Functions

Purpose	Command	Example
Mean	`mean(x)`	
Median	`median(x)`	
Standard deviation	`sd(x)`	
Variance	`var(x)`	
Quantiles	`quantile(x)`	
Correlation	`cor(x, y)`	
Covariance	`cov(x, y)`	

4. Basic Visualization

Purpose	Command	Example
Histogram	`hist(x)`	
Scatter plot	`plot(x, y)`	
Boxplot	`boxplot(x)`	
Bar plot	`barplot(table(x))`	
Save plot to PDF	`pdf("plot.pdf"); plot(x, y); dev.off()`	

Modeling of random variables.

Simulation of Random Variables in R

1. Direct Simulation (Built-in Distributions)

R provides built-in random number generators for most common probability distributions. They all follow the same naming convention:

$$r < distribution > (n, \text{parameters})$$

where `n` is the number of random values to generate.

Distribution	Function	Example
Uniform	`runif()`	`runif(1000, min = 0, max = 1)`
Normal	`rnorm()`	`rnorm(1000, mean = 0, sd = 1)`
Bernoulli / Binomial	`rbinom()`	`rbinom(1000, size = 1, prob = 0.5)`
Poisson	`rpois()`	`rpois(1000, lambda = 4)`
Exponential	`rexp()`	`rexp(1000, rate = 1/2)`
Chi-square	`rchisq()`	`rchisq(1000, df = 5)`
Student's t	`rt()`	`rt(1000, df = 10)`
F distribution	`rf()`	`rf(1000, df1 = 5, df2 = 10)`
Beta	`rbeta()`	`rbeta(1000, 2, 5)`
Gamma	`rgamma()`	`rgamma(1000, shape = 2, rate = 1)`

2. Inverse Transform Method

This method is used when the cumulative distribution function (CDF) $F(x)$ is known, but there is no direct generator for the random variable.

Idea:

1. Generate $U \sim \text{Uniform}(0, 1)$.
2. Compute $X = F^{-1}(U)$, where F^{-1} is the inverse CDF.

Example: Simulating an Exponential Distribution manually

```
# Inverse Transform Sampling for an Exponential RV with rate = 2
u <- runif(1000)
x <- -log(1 - u) / 2
hist(x, main = "Simulated Exponential Distribution", col = "lightblue")
```

The generated variable `x` now follows an exponential distribution with rate parameter $\lambda = 2$.

3. Working with Vectors in R

Vectors are the fundamental data structure in R. They can store numeric, character, or logical values and allow vectorized operations.

Creating vectors:

```
x <- c(1, 2, 3, 4, 5)       # combine values into a vector
y <- rep(1, 1000)           # 1000 ones
z <- seq(0, 10, by = 2)     # sequence: 0, 2, 4, 6, 8, 10
```

Basic operations:

```
x + 2        # adds 2 to each element
x * y        # element-wise multiplication
x[3]         # third element
x[1:3]       # first three elements
```

The `sum()` function: `sum()` adds up all the elements of a numeric vector and returns their total. It is a fully vectorized operation, meaning it automatically processes each element.

```
x <- c(1, 2, 3, 4)
sum(x)            # 10
sum(x > 2)        # counts how many elements are greater than 2
sum(is.na(x))     # counts missing values (NA)
```

Note: Because R is vectorized, expressions like `mean(x)`, `var(x)`, and `sd(x)` also work directly on vectors without loops.