AI and Biotechnology/Bioinformatics

R Crash Course (2025)

Session 5: Basics of R Programming Language Part IV

(Data Types in R)

Welcome back to another session of the R Crash Course!

In today's session, we'll dive into an essential topic: Data Types in R.
Understanding data types is the foundation of writing efficient, clean and error-free R code. for data analysis

> If you missed the previous sessions or want to catch up, feel free to explore the complete course materials on our **GitHub repository**.

By the end of today's session, you'll be able to:

- Identify the most common data types in R.
- Convert between data types safely.
- Apply these concepts while creating real-world datasets.
- Understand how R handles strings, numbers, logical values, and categorical data.

Let's get started and build a strong foundation for data analysis in R!

Introduction

Understanding data types is important in R, as it affects how data is stored, manipulated, and analyzed.
This session introduces the five most common data types in R:

- Numeric
- Integer
- Character
- Factor
- Logical

---

1. Numeric

Numeric data includes real number with decimal place.
Either positive or negative.
This type is continuous and is the default for numbers in R. Decimal values are called numeric in R.
It is the default R data type for numbers in R.

```
var <- 40.8
class(var)  # "numeric"
```

```
## [1] "numeric"
```

```
var2 <- -40.6
class(var2)
```

```
## [1] "numeric"
```

```
var3 <- 40
class(var3)  # still "numeric"
```

```
## [1] "numeric"
```

2. Integer

Integer data includes whole numbers without decimal points.
Since by default R assign numeric class to number either with or without decimal point.
So for whole number we add L to a number that explicitly define it as an integer.

```
a <- 25
class(a)  # "numeric"
```

```
## [1] "numeric"
```

```
b <- 25L
class(b)  # "integer"
```

```
## [1] "integer"
```

Converting numeric to integer:

```
var <- as.integer(var)
class(var)  # "integer"
```

```
## [1] "integer"
```

Note:

R truncates the decimal part when converting to integer.

```
num_val <- 40.9
int_val <- as.integer(num_val)
int_val  # 40
```

```
## [1] 40
```

When to use integer?

- Integer consumes less memory.
- Useful for large datasets or ML pipelines.

Converting back to numeric:

```
back_to_num <- as.numeric(int_val)
class(back_to_num)  # "numeric"
```

```
## [1] "numeric"
```

---

3. Character

Character data holds text or string values. Use quotes(single ’’ or double ““) to define them.

```
chr <- "biotechnology"
class(chr)  # "character"
```

```
## [1] "character"
```

```
# Numbers in quotes are also treated as characters.
a <- "2"
class(a)  # "character"
```

```
## [1] "character"
```

```
b <- 2
class(b)  # "numeric"
```

```
## [1] "numeric"
```

---

4. Factor

Factors are used to represent categorical data such as group, disease status, gender, etc. It can be written as character or numeric datatype e.g normal, cancer or 0, 1 (0 represents normal, 1 is for cancer)

```
status <- c("normal", "cancer", "normal")
status_fac <- as.factor(status)
class(status_fac)  # "factor"
```

```
## [1] "factor"
```

```
levels(status_fac)
```

```
## [1] "cancer" "normal"
```

Releveling:

By default levels will be in alphabetical order i.e cancer, normal You can relevel factors to define custom order with function factor()

```
status_fac <- factor(status_fac,
                     levels = c("normal", "cancer"))

class(status_fac)
```

```
## [1] "factor"
```

```
levels(status_fac)
```

```
## [1] "normal" "cancer"
```

Convert factor to numeric:

```
# Convert character/factor to numeric using factor() function
# Example # 0 = normal, 1 = cancer

status_num_fac <- factor(status_fac,
                         levels = c("normal", "cancer"),
                         label = c(0, 1))
class(status_num_fac)
```

```
## [1] "factor"
```

```
levels(status_num_fac)
```

```
## [1] "0" "1"
```

Convert it by using ifelse statement

```
target <- ifelse(status == "normal", 0, 1)
class(target)  # numeric
```

```
## [1] "numeric"
```

```
target_fac <- as.factor(target)
class(target_fac)  # factor
```

```
## [1] "factor"
```

---

5. Logical

Logical data holds **TRUE or FALSE** values. Useful for filtering, conditions, and decision-making.

```
age <- c(15, 21, 35)
old <- age > 20
old  # FALSE TRUE TRUE
```

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

Practical Example: Data Frame with Mixed Data Types

Let's create simple dataset and understand all data types in one place.

```
df <- data.frame(
  name = c("Ali", "Sara", "John", "Ayesha", "Usman"),
  age = c(25, 30, 19, 45, 50),
  weight = c(70.5, 60.2, 55.0, 68.7, 72.1),
  height = c("Short", "Tall", "Medium", "Short", "Tall"),
  gender = c("Male", "Female", "Male", "Female", "Male")
)

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

```
## 'data.frame':    5 obs. of  5 variables:
##  $ name  : chr  "Ali" "Sara" "John" "Ayesha" ...
##  $ age   : num  25 30 19 45 50
##  $ weight: num  70.5 60.2 55 68.7 72.1
##  $ height: chr  "Short" "Tall" "Medium" "Short" ...
##  $ gender: chr  "Male" "Female" "Male" "Female" ...
```

Convert columns:

```
df$age <- as.integer(df$age)

df$height <- as.factor(df$height)
levels(df$height)
```

```
## [1] "Medium" "Short"  "Tall"
```

```
df$height <- factor(df$height, levels = c("Short", "Medium", "Tall"))
levels(df$height)
```

```
## [1] "Short"  "Medium" "Tall"
```

```r
df$gender <- as.factor(df$gender)
levels(df$gender)
```

```
## [1] "Female" "Male"
```

Numeric conversion from factor:

```r
df$gender_num <- ifelse(df$gender == "Female", 1, 0)
class(df$gender_num)  # numeric
```

```
## [1] "numeric"
```

Add a logical column:

```r
df$is_old <- df$age > 30
df
```

```
##     name age weight height gender gender_num is_old
## 1    Ali  25   70.5  Short   Male          0  FALSE
## 2   Sara  30   60.2   Tall Female          1  FALSE
## 3   John  19   55.0 Medium   Male          0  FALSE
## 4 Ayesha  45   68.7  Short Female          1   TRUE
## 5  Usman  50   72.1   Tall   Male          0   TRUE
```

Summary

```r
str(df)
```

```
## 'data.frame':    5 obs. of  7 variables:
##  $ name      : chr  "Ali" "Sara" "John" "Ayesha" ...
##  $ age       : int  25 30 19 45 50
##  $ weight    : num  70.5 60.2 55 68.7 72.1
##  $ height    : Factor w/ 3 levels "Short","Medium",..: 1 3 2 1 3
##  $ gender    : Factor w/ 2 levels "Female","Male": 2 1 2 1 2
##  $ gender_num: num  0 1 0 1 0
##  $ is_old    : logi  FALSE FALSE FALSE TRUE TRUE
```

That’s all about today !

- We learned the core data types in R
- Created a sample dataset
- Performed data type conversion
- Used factor releveling
- Used ifelse to create numeric columns from categorical values

What's Next? In the next session, you'll understand:

- Data structure in R
- Data assessment
- data manipulation
- Subsetting and filtering

> Want to try it yourself? Get the R script from GitHub
>
> **📂 Access GitHub Repository**