**Data cleaning in R**

**Part 1: Introduction to Data Cleaning & Transformation**

**1.1 Understanding Data Structures in R**

Before diving into cleaning data, it’s crucial to understand the structures you're working with.

1. **Data Frames**:  
   A data frame is a table or two-dimensional array-like structure in R. It is the most common data structure for working with datasets.
   * **Example**:

data <- data.frame(name = c("John", "Alice", "Bob"), age = c(25, 30, 22))

print(data)

1. **Tibbles**:  
   A more modern and user-friendly version of data frames, provided by the tibble package. They show the data in a cleaner format and don’t convert strings to factors by default.
   * **Example**:

library(tibble)

data <- tibble(name = c("John", "Alice", "Bob"), age = c(25, 30, 22))

1. **Vectors**:  
   These are one-dimensional arrays that can hold numeric, character, or logical data.
   * **Example**:

age <- c(25, 30, 22)

1. **Lists**:  
   Lists are more complex data structures that can store multiple types of objects.
   * **Example**:

info <- list(name = "John", age = 25, scores = c(85, 90, 78))

**1.2 Loading and Exploring Data**

To start cleaning data, you first need to load it into R.

1. **Reading Data**:
   * **CSV**:  
     The most common file format. Use read.csv() to load it into R.

R

Copy code

data <- read.csv("data.csv")

* + **Excel**:  
    Use readxl to read Excel files.

R

Copy code

library(readxl)

data <- read\_excel("data.xlsx")

1. **Exploring Data**: After loading the data, you can quickly explore its structure and summary.
   * **str()**: Displays the structure of an object (e.g., columns and types).

R

Copy code

str(data)

* + **summary()**: Provides a summary of each column (e.g., min, max, mean).

R

Copy code

summary(data)

* + **head()**: Displays the first few rows of the data.

R

Copy code

head(data)

**1.3 Common Data Issues to Address**

Here are the common issues you’ll encounter in real-world datasets:

1. **Missing Data (NA)**:  
   Missing data is a significant problem that can be dealt with in several ways.
   * **Detecting missing values**:  
     Use is.na() to check for missing values.

R

Copy code

is.na(data$age)

* + **Removing rows with missing values**:  
    You can remove rows with missing values using na.omit().

R

Copy code

clean\_data <- na.omit(data)

* + **Replacing missing values**:  
    You can replace NA with the mean, median, or a specified value.

R

Copy code

data$age[is.na(data$age)] <- mean(data$age, na.rm = TRUE)

1. **Duplicates**:  
   Duplicate rows can skew analysis.
   * **Identifying duplicates**:  
     Use duplicated() to find duplicates.

R

Copy code

duplicated(data)

* + **Removing duplicates**:  
    You can remove duplicates using !duplicated().

R

Copy code

clean\_data <- data[!duplicated(data), ]

**Part 2: Basic Data Cleaning and Transformation**

**2.1 Cleaning Text Data**

Working with text data is a common task in data cleaning. Functions from the stringr package make this easier.

1. **Trimming Whitespace**:
   * **str\_trim()** removes any leading or trailing spaces.

R

Copy code

library(stringr)

clean\_name <- str\_trim(data$name)

1. **Changing Case**:
   * **str\_to\_upper()** and **str\_to\_lower()** change the case of characters.

R

Copy code

data$name <- str\_to\_upper(data$name)

1. **Replacing Text**:
   * **str\_replace()** replaces a specified pattern with a new string.

R

Copy code

data$name <- str\_replace(data$name, "Dr.", "")

1. **Extracting Patterns**:
   * **str\_extract()** extracts the first match of a pattern.

R

Copy code

email <- str\_extract(data$email, "[A-Za-z0-9]+@[A-Za-z0-9]+\\.[a-z]+")

1. **Splitting Strings**:
   * **str\_split()** splits a string into components.

R

Copy code

split\_names <- str\_split(data$name, "\_")

**2.2 Handling Date-Time Data**

Dates often come in various formats. The lubridate package helps you handle date-time data seamlessly.

1. **Parsing Dates**:
   * **ymd()**, **dmy()**, **mdy()**: Functions to parse dates in various formats.

R

Copy code

library(lubridate)

data$date <- ymd(data$date)

1. **Extracting Date Components**:
   * You can extract parts like year, month, or day using year(), month(), and day().

R

Copy code

year(data$date)

month(data$date)

1. **Time Manipulation**:
   * **floor\_date()** and **ceiling\_date()**: Round dates to the nearest specified unit.

R

Copy code

floor\_date(data$date, "month")