**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.

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

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

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).

str(data)

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

summary(data)

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

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.

is.na(data$age)

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

clean\_data <- na.omit(data)

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

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.

duplicated(data)

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

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.

library(stringr)

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

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

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

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

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

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

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.

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.

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().

year(data$date)

month(data$date)

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

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

**1. Key Text Cleaning Operations**

Below are common tasks and functions for cleaning text data in R.

**1.1 Removing Unwanted Characters**

* **gsub() (Base R)**: Replaces all matches of a pattern in a string.
* **Example**: Remove special characters like punctuation.

R

Copy code

text <- "Hello! How are you?"

clean\_text <- gsub("[[:punct:]]", "", text)

# Output: "Hello How are you"

* **str\_remove\_all() (stringr)**: Remove all occurrences of a pattern.

R

Copy code

library(stringr)

clean\_text <- str\_remove\_all(text, "[[:punct:]]")

**1.2 Removing Numbers**

* **gsub() or str\_remove\_all()**:

R

Copy code

text <- "Product ID: 12345"

clean\_text <- gsub("[0-9]", "", text)

# Output: "Product ID: "

**1.3 Lowercasing Text**

* Text normalization often requires making all text lowercase.
* **Function**: tolower() (Base R) or str\_to\_lower() (stringr).

R

Copy code

text <- "HELLO World"

clean\_text <- tolower(text)

# Output: "hello world"

**1.4 Removing Extra Whitespace**

* **Function**: str\_squish() (stringr).  
  This removes leading, trailing, and extra spaces between words.

R

Copy code

text <- " Hello World "

clean\_text <- str\_squish(text)

# Output: "Hello World"

**1.5 Stemming and Lemmatization**

* **Stemming**: Reduces words to their base/root form.
  + **Function**: wordStem() from the SnowballC package.

R

Copy code

library(SnowballC)

words <- c("running", "runs", "runner")

stems <- wordStem(words)

# Output: "run", "run", "runner"

* **Lemmatization**: Similar to stemming but ensures grammatically correct base forms.
  + **Tool**: Use external packages like textstem.

R

Copy code

library(textstem)

lemmatized\_words <- lemmatize\_words(c("running", "better", "children"))

# Output: "run", "good", "child"

**1.6 Tokenization**

Splitting text into individual components (words, sentences, or phrases).

* **Function**: unnest\_tokens() from tidytext package.

R

Copy code

library(tidytext)

text\_data <- data.frame(text = c("Hello World", "R is great"))

tokenized <- unnest\_tokens(text\_data, word, text)

# Output: "hello", "world", "r", "is", "great"

**1.7 Removing Stop Words**

Stop words (e.g., "is", "and", "the") are commonly removed during text analysis.

* **Function**: Use the stop\_words dataset from the tidytext package.

R

Copy code

library(tidytext)

data("stop\_words")

cleaned\_data <- anti\_join(tokenized, stop\_words, by = "word")

**2. Advanced Text Cleaning Operations**

**2.1 Regex for Pattern Matching**

**Regular Expressions (Regex)** allow powerful pattern-based text cleaning.

* **Removing Emails**:

R

Copy code

text <- "Contact me at john.doe@example.com"

clean\_text <- gsub("[a-zA-Z0-9.\_%+-]+@[a-zA-Z0-9.-]+\\.[a-zA-Z]{2,}", "", text)

# Output: "Contact me at "

* **Extracting Phone Numbers**:

R

Copy code

text <- "Call me at (123) 456-7890"

phone <- str\_extract(text, "\\(\\d{3}\\) \\d{3}-\\d{4}")

# Output: "(123) 456-7890"

* **Replacing Multiple Spaces with a Single Space**:

R

Copy code

text <- "This is a test"

clean\_text <- gsub("\\s+", " ", text)

# Output: "This is a test"

**2.2 Splitting and Joining Text**

* **Splitting Strings**: Use str\_split() to divide text based on a delimiter.

R

Copy code

text <- "apple,banana,cherry"

split\_text <- str\_split(text, ",")[[1]]

# Output: c("apple", "banana", "cherry")

* **Joining Strings**: Use str\_c() to concatenate strings.

R

Copy code

words <- c("Hello", "World")

joined\_text <- str\_c(words, collapse = " ")

# Output: "Hello World"

**2.3 Handling Encoding Issues**

Sometimes, text may have encoding issues (e.g., unusual symbols).

* **Function**: iconv() converts text to a specified encoding.

R

Copy code

text <- "Müller"

clean\_text <- iconv(text, from = "UTF-8", to = "ASCII//TRANSLIT")

# Output: "Muller"

**3. Workflow for Cleaning Text Data**

Here’s a step-by-step example:

**Raw Data**

R

Copy code

text\_data <- c(

" HELLO, my name is John!!! ",

"You can contact me at john.doe@example.com.",

"123 Main Street"

)

**Cleaning Steps**

R

Copy code

library(stringr)

cleaned\_text <- text\_data %>%

str\_to\_lower() %>% # Convert to lowercase

str\_squish() %>% # Remove extra spaces

str\_remove\_all("[[:punct:]]") %>% # Remove punctuation

str\_remove\_all("\\d+") # Remove numbers

# Output: "hello my name is john", "you can contact me at", "main street"

**4. Study Focus**

* **Regex Mastery**: Learn the syntax for pattern matching ([a-z], \\s, +, ^, $).
* **Text Packages**:
  + stringr for flexible string manipulation.
  + tidytext for tokenization and sentiment analysis.
  + tm (Text Mining) for advanced cleaning and preprocessing.