Tasks

Data Cleaning

- → Find & deal with missing Data.
- → Detect & Remove duplicates.
- → Record variable values.
- → Clean & Prepare data for analysis.

Datasets (starwars)

What this data set represent?

The starwars dataset contain information about 87 characters from the starwars movie Universe.

Each row: one character (like Luke skywalker, Darth Vader, Yoda)

Each cloumn: One feature or property of that character (like height, mass, species etc)

→ Information about Characters

Lukeskywalke → Main hero is the

movie. Learns to use the force to fight evil.

Darth Vader → Villain → he turns evil and become servant of the Dark side.

Yoda → Jedi Master → A wise, small green alien who trains Luke skywalker.

---

1. **Data("starwars")**
   ↳ Load the dataset named starwars (from the **dplyr** Package).

2. **View(starwars)**
   ↳ open data set in a spreadsheet-style viewer inside R studio.
   ↳ A Table with rows & columns.
   Each row : a star war character
   Each column: features of characters (like height, weight, species).

3. **glimpse(starwars)**
   ↳ Gives quick summary

ob variable names, datatypes & first few values in compact view.

Example

Rows: 87

Columns: 14

$name <chr> "Luke skywalker", "C-3PO"...

$height <int> 172, 167, 96, 202

variables — variable type — values

4 Class(starwars $gender)

$ → This sign is used to select a specific column from data frame.

→ means the gender column from the starwars dataset. (character).

5 Unique (starwars $ gender)

↳ Different genders in starwars like: "masculine", "feminine", "NA" → (missing data → no gender specific).

6 changing datatype.

```
starwars $gender <- as.factor(starwars $ gender)
// converting the column gender from
"character" type to a "factor" type.
```

→ Factor:

↳ factor is used to store categorical data (data with limited, fixed categories).

↳ Also tells that these types of variable has levels (categories) like "male, "female" etc.

7 levels(starwars $ gender)

checking leveles

"feminine" "masculine"

8 Select Variables:-

```
names(starwars)
starWars %>% (ctrl + shit + M)
    select(name, height, end-with("color"))
```

%>% → Pipe Operator

"Take the dataset starswars, and then ..."

This pipe passes the output of the left-hand side (starwars) as input to the function on the right-hand side.

9 **Filter Observation:**

```
unique (starwars $hair_color)
starwars %>%
    select (name, height,
ends_with ("color")) %>%
filter (hair_color %in% c ("blond",
    "brown") & height < 180).
```

10 **Missing Values:-**

```
mean (starwars $height)
```

↳ It give result **(NA)**

R cannot calculate the mean if there are any NA's it will return **NA**.

so we add

na.rm = TRUE

↳ Not Available values Remove.

→ Remove all NA values before calculating the Mean.

```
starwars %>%
  select (name, gender, height, hair-
color)
```

It shows all the values of these columns with Null values.

```
starwars %>%
  select (name, gender, height,
hair-color) %>%
    na.omit
```

→ It gives 72 values.

↳ It does not show NA values.

But our main problem is we do not know what we lost, we don't know which observations are removed.

To understand where missingness is

```
starwars %>%
    select (name, gender,
height, hair-color) %>%
  filter (! complete.cases (.))
```

↳ (complete.cases (.)) → True for Complete Rows

! complete.cases(.) True for incomplete values/ rows.

This shows NA (Not Available Data).

→ "Not" in R (logical negation).

We are filtering the rows where ! complete.cases(.) is TRUE.

means rows that have missing values.

Why we use this → (.) ?

The . means "the current data set being passed through the Pipe"

After select(....), the data that comes out (a smaller table) is sent to filter(). Inside filter(), The dot (.) represent the data.

Examples:

① x ← starwars %>% select(name, height) %>% filter(x, height > 180).

② starwars %>%
select(name, height) %>%
filter(height > 180)

3

```
starWars %>%
    select(name, height) %>%
    filter(!complete.cases(.)).
```

All mean the same thing.

But the dot lets you use the **Piped-in data** directly, without creating a separate variable like x

11 Drop.na(height) → bar (drop not available value of height).

12 **Replace values**

It filter all not available values and then create column (mutate) hair-color-2 and replace na values with (none) using original hair-color column.

```
starWars %>%
    "          "          "
    "          "          "
    mutate(hair-color-2 = replace-na(
    hair-color, "none")).
```

↳ used to create new column or modify existing one.

13

Duplicates

```
Names <- c("Peter", "John",
"Andrew", "Peter")
Age <- c(22, 33, 44, 22)

friends <- data.frame(Names, Age)
friends
```

→

```
friends[!duplicated(friends),]
```

14

Recoding variables.

```
starWars %>%
    select(name, gender) %>%
    mutate(gender = recode(gende
        "masculine" = 1,
        "feminine" = 2)),
```

Recode function is used to replace specific values inside a column.

```
recode(gender,
    "masculine" = 1,
    "feminine" = 2))
```

15

To find which columns are numeric & categorical or Factor.

In (tidyverse).

Numeric columns

```
Starwars %>%
  select(where(is.numeric)) %>%
  names()
```

categorical columns.

```
starwars %>%
  select(where(~is.character(.) |
    is.factor(.))) %>%
  names().
```

Using sapply

see code

Univariate Analysis

→ Univariate Analysis looks at one variable at a time.

→ Focus on measure of central tendency, dispersion and frequency.

Numeric: Mode, Mean, Median, Min, Max, Range, SD, Variance, Histogram, Boxplot.

Categorical: Frequency tables, Mode, Bar char, Pie chart.