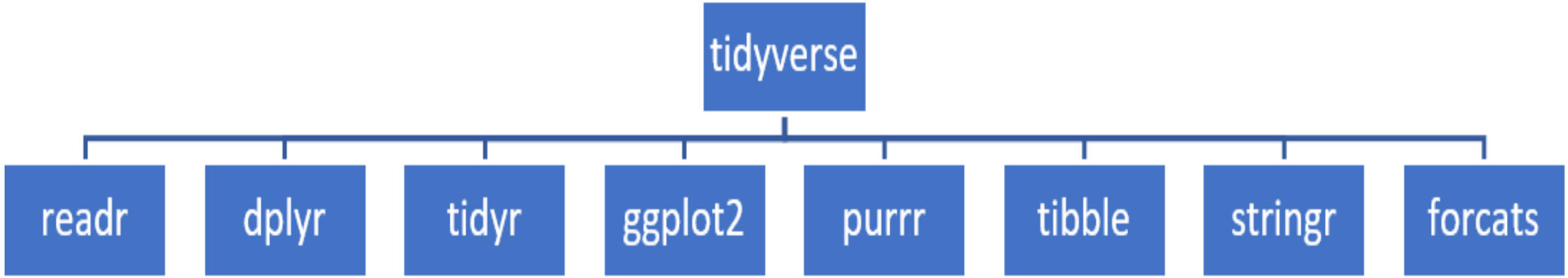
Data Wrangling – Part 1

tidyverse Package

Tidyverse: is a collection of R packages designed to facilitate data manipulation, exploration, visualization, and modeling.



Reading Data

From an Excel file in your computer

From a dataset that comes from a specific package.

I will illustrate this with RStudio. Specific and detailed steps on how to do it can be found in your student notes.

Examining the Data

- `slice()`: Display the first few rows

Command: `slice(name_of_dataframe, 1:5)`

Example 2: `slice(penguins, 1:3)`

Output you will see in your console:

```
> slice(penguins, 1:3)
# A tibble: 3 × 8
  species island    bill_length_mm bill_depth_mm
  <fct>   <fct>              <dbl>         <dbl>
1 Adelie  Torgersen           39.1          18.7
2 Adelie  Torgersen           39.5          17.4
3 Adelie  Torgersen           40.3          18
# ℹ 4 more variables: flipper_length_mm <int>,
#   body_mass_g <int>, sex <fct>, year <int>
```

- `glimpse()`: Get summary of the variables

Command: `glimpse(name_of_dataframe)`

Example 3: `glimpse(penguins)`

Output you will see in your console:

```
> glimpse(penguins)
Rows: 344
Columns: 8
$ species           <fct> Adelie, Adelie, Adelie, Adelie…
$ island            <fct> Torgersen, Torgersen, Torgerse…
$ bill_length_mm    <dbl> 39.1, 39.5, 40.3, NA, 36.7, 39…
$ bill_depth_mm     <dbl> 18.7, 17.4, 18.0, NA, 19.3, 20…
$ flipper_length_mm <int> 181, 186, 195, NA, 193, 190, 1…
$ body_mass_g       <int> 3750, 3800, 3250, NA, 3450, 36…
$ sex               <fct> male, female, female, NA, fema…
$ year              <int> 2007, 2007, 2007, 2007, 2007, …
```

- `str()`: Display the structure of the data

Command: `str(name_of_dataframe)`

Example 4: `str(penguins)`

Output you will see in your console:

```
> str(penguins)
tibble [344 × 8] (S3: tbl_df/tbl/data.frame)
 $ species          : Factor w/ 3 levels "Adelie","Chinstrap",..: 1 1 1 1 1
 $ island           : Factor w/ 3 levels "Biscoe","Dream",...: 3 3 3 3 3 3 3
 $ bill_length_mm   : num [1:344] 39.1 39.5 40.3 NA 36.7 39.3 38.9 39.2 34.1
 $ bill_depth_mm    : num [1:344] 18.7 17.4 18 NA 19.3 20.6 17.8 19.6 18.1 2
 $ flipper_length_mm: int [1:344] 181 186 195 NA 193 190 181 195 193 190 ...
 $ body_mass_g      : int [1:344] 3750 3800 3250 NA 3450 3650 3625 4675 347
 $ sex              : Factor w/ 2 levels "female","male": 2 1 1 NA 1 2 1 2 N
 $ year             : int [1:344] 2007 2007 2007 2007 2007 2007 2007 2007 20
```

- `table()`: it's typically used for one or two columns. For one column, it tabulates each observation, and shows how many times each observation appeared (or repeats) in that column.

Command:

```
table(name_of_dataframe$name_of_column)
```

```
table(name_of_dataframe$name_of_column1,
name_of_dataframe$name_of_column2)
```

Example 5: `table(penguins$sex)`

```
> table(penguins$sex)

female   male
   165    168
```

```
table(penguins$sex, useNA = "ifany")
```

```
> table(penguins$sex, useNA = "ifany")

female   male   <NA>
   165    168     11
```

Tells you how many missing values (NA's) there are in the column.

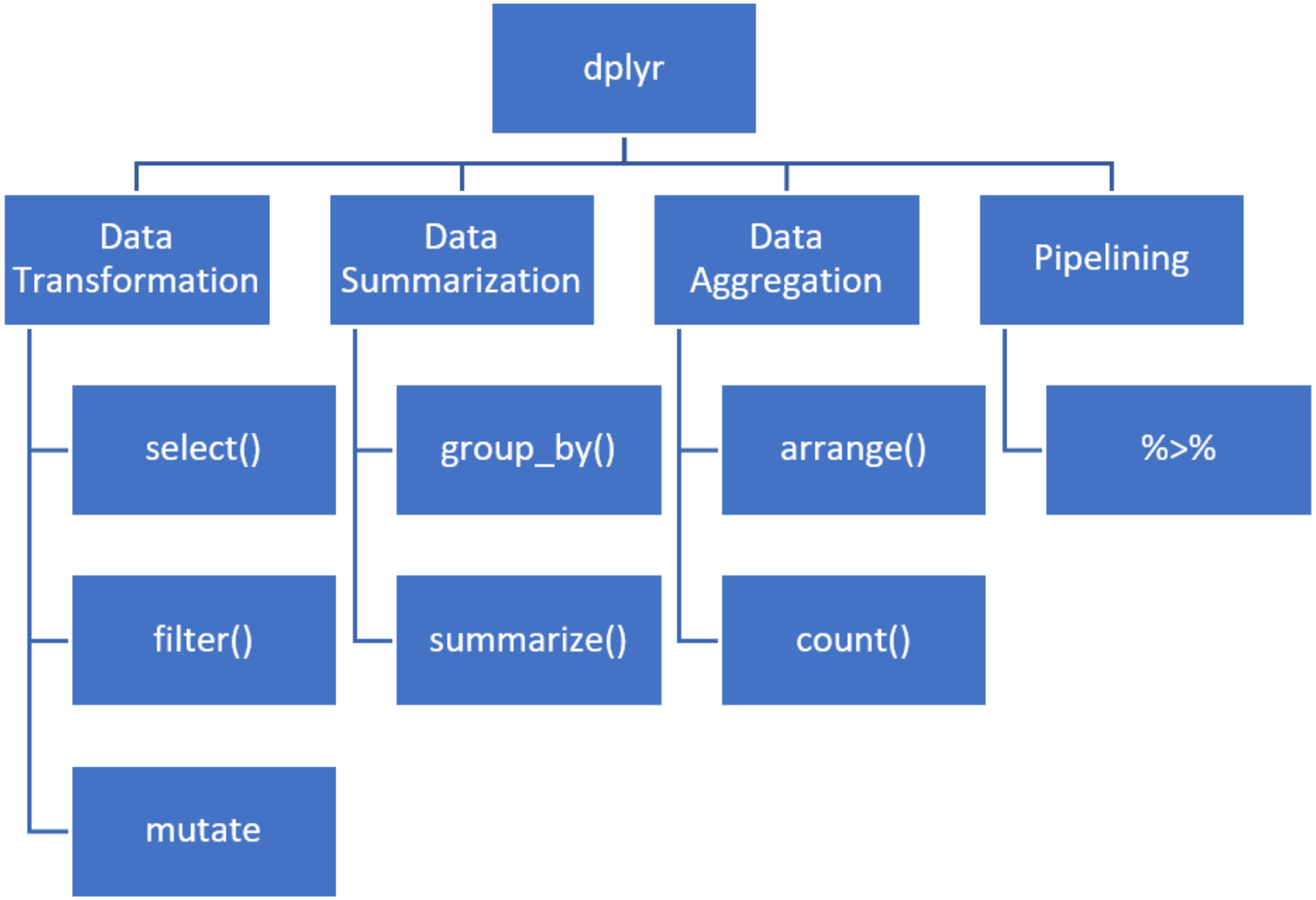
- `colnames()`: Display the structure of the data

Command: `colnames(name_of_dataframe)`

Example 6: `colnames(penguins)`

```
> colnames(penguins)
[1] "species"           "island"
[3] "bill_length_mm"    "bill_depth_mm"
[5] "flipper_length_mm" "body_mass_g"
[7] "sex"               "year"
```

dplyr Package



name you will choose to save all the steps you are doing.

name of the dataframe you will be using to perform your analysis

Command Illustration

```
new_dataframe_name <- dataframe_name %>%
    select(column_name1) %>%
    select(-column_name2)
```

`select()` Function

Illustration_Data

you can find this from the repository inside the data file

Name	Age	total_Income	var_1	var_2	var_3	zipcode	honesty	cat_total
Val	18	18000	apple	carrots	elephant	60001	agree	0
Derek	25	25000	grapes	carrots	tiger	60073	disagree	1
Whitney	30	30000	bananas	carrots	lion	60109	disagree	2
Sasha	40	40000	peaches	carrots	rabbit	60111	disagree	1
Daniella	45	45000	bananas	carrots	shark	60155	agree	1

Different ways to use `select()`

1. Choose specific column name from the data frame

Example 7: Select columns "Age" and "total_Income"

```
example_7 <- Illustration_Data %>%
     select(Age, total_Income)
```

Output: The output is a dataframe that looks like this

Name	Age	total_Income	var_1	var_2	var_3	zipcode	honesty	cat_total
Val	18	18000	apple	carrots	elephant	60001	agree	0
Derek	25	25000	grapes	carrots	tiger	60073	disagree	1
Whitney	30	30000	bananas	carrots	lion	60109	disagree	2
Sasha	40	40000	peaches	carrots	rabbit	60111	disagree	1
Daniella	45	45000	bananas	carrots	shark	60155	agree	1

Output: The output is a dataframe that looks like this

Age	total_Income
18	18000
25	25000
30	30000
40	40000
45	45000

2. Select Columns by Name Patterns: You can use special helper functions like:

- `starts_with()`
- `ends_with()`
- `contains()`
- `matches()`
- `everything()`

to select columns based on their names.

Example 8: Select columns that start with "var"

```
example_8 <- Illustration_Data %>%
    select(starts_with("var"))
```

Output: The output is a dataframe that looks like this

Name	Age	total_Income	var_1	var_2	var_3	zipcode	honesty	cat_total
Val	18	18000	apple	carrots	elephant	60001	agree	0
Derek	25	25000	grapes	carrots	tiger	60073	disagree	1
Whitney	30	30000	bananas	carrots	lion	60109	disagree	2
Sasha	40	40000	peaches	carrots	rabbit	60111	disagree	1
Daniella	45	45000	bananas	carrots	shark	60155	agree	1

Output: The output is a dataframe that looks like this

var_1	var_2	var_3
apple	carrots	elephant
grapes	carrots	tiger
bananas	carrots	lion
peaches	carrots	rabbit
bananas	carrots	shark

Example 9: Select columns that contains "total" in their names

```
example_9 <- Illustration_Data %>%
    select(contains("total"))
```

Output: The output is a dataframe that looks like this

Name	Age	total_Income	var_1	var_2	var_3	zipcode	honesty	cat_total
Val	18	18000	apple	carrots	elephant	60001	agree	0
Derek	25	25000	grapes	carrots	tiger	60073	disagree	1
Whitney	30	30000	bananas	carrots	lion	60109	disagree	2
Sasha	40	40000	peaches	carrots	rabbit	60111	disagree	1
Daniella	45	45000	bananas	carrots	shark	60155	agree	1

Output: The output is a dataframe that looks like this

total_Income	cat_total
18000	0
25000	1
30000	2
40000	1
45000	1

3. Exclude Columns: To exclude specific columns, you can use the "-" (minus) sign before the column name.

Example 10: Exclude columns "honesty" and "zipcode"

```
example_10 <- Illustration_Data %>%
    select(-honesty, -zipcode)
```

Output: The output is a dataframe that looks like this

Name	Age	total_Income	var_1	var_2	var_3	zipcode	honesty	cat_total
Val	18	18000	apple	carrots	elephant	60001	agree	0
Derek	25	25000	grapes	carrots	tiger	60073	disagree	1
Whitney	30	30000	bananas	carrots	lion	60109	disagree	2
Sasha	40	40000	peaches	carrots	rabbit	60111	disagree	1
Daniella	45	45000	bananas	carrots	shark	60155	agree	1

Output: The output is a dataframe that looks like this

Name	Age	total_Income	var_1	var_2	var_3	cat_total
Val	18	18000	apple	carrots	elephant	0
Derek	25	25000	grapes	carrots	tiger	1
Whitney	30	30000	bananas	carrots	lion	2
Sasha	40	40000	peaches	carrots	rabbit	1
Daniella	45	45000	bananas	carrots	shark	1

4. Select Columns by Index and Range: you can use a numeric vector to select the columns according to their index position.

Example 11: Exclude columns "honesty" and "zipcode"

```
example_11 <- Illustration_Data %>%
    select(c(1,3,5,6))
```

Output: The output is a dataframe that looks like this

1	2	3	4	5	6	7	8	9
Name	Age	total_Income	var_1	var_2	var_3	zipcode	honesty	cat_total
Val	18	18000	apple	carrots	elephant	60001	agree	0
Derek	25	25000	grapes	carrots	tiger	60073	disagree	1
Whitney	30	30000	bananas	carrots	lion	60109	disagree	2
Sasha	40	40000	peaches	carrots	rabbit	60111	disagree	1
Daniella	45	45000	bananas	carrots	shark	60155	agree	1

Output: The output is a dataframe that looks like this

Name	total_Income	var_2	var_3
Val	18000	carrots	elephant
Derek	25000	carrots	tiger
Whitney	30000	carrots	lion
Sasha	40000	carrots	rabbit
Daniella	45000	carrots	shark