dplyr : A Grammar of Data Manipulation

Basic functions of “dplyr”

select()	Selection of specific columns
filter()	Filtering data based on condition
mutate()	Creating new variables/columns
arrange()	Sorting dataset
na.omit()	Remove rows with missing value.

- Loading “dplyr” package and importing data.

```
library(dplyr)

classf <- read.csv('http://s.anilz.net/wb_class') #World Bank country classification
energy <- read.csv('http://s.anilz.net/wb_energy') #World Bank energy dataset
var_def <- read.csv('http://s.anilz.net/wb_var_def') #Variables definition
```

- Examples of basic functions of “dplyr” package.
  a. Selecting columns [Syntax : **select(data, column1, column2, …..)**

```
View(energy)
data_select <- select(energy, country, ccode, year, tfec)
View(data_select)
```

  b. Filtering data [Syntax : **filter(data, condition1, condition2, ...)]**

```
data_nepal <- filter(energy, country == "Nepal")
View(data_nepal)
```

  c. Create new columns [Syntax : **mutate(data, new_col = expression, ...)]**

```
data_ren_ele_share <- mutate(energy, ren_ele_share = ren_ele/tot_ele*100)
View(data_ren_ele_share)
```

  d. Sorting data [Syntax : **arrange(data, col1, col2, ….)]**

```
data_sort <- arrange(energy,year,desc(country))
View(data_sort)
```

  e. Remove rows with missing values [Syntax : **na.omit(data)]**

```
data_na_omit <- na.omit(energy)
View(data_na_omit)
```

Advance functions of “dplyr”

- **Piping (%>%)** : Piping is used for chaining multiple operations together in a clean way.

  Example: Suppose you are interested in renewable electricity output data in Nepal and India. Now, you want to perform the following operations with the help of piping (%>%).

  - Select columns **year, country, ren_ele, tot_ele** from **energy** dataframe.
  - Keep data of Nepal and India only.
  - Sort the dataframe according to **country** and **year** columns.
  - Create a new column **ren_ele_share** by calculating share of renewable electricity output in total output (i.e. **ren_ele/tot_ele*100**).
  - Save the new dataframe as **energy_np_in**

```
energy_np_in <- energy %>%
  select(year, country, ren_ele, tot_ele) %>%
  filter(country == 'Nepal' | country =='India') %>%
  arrange(country, year) %>%
  mutate(ren_ele_share = ren_ele/tot_ele*100)
View(energy_np_in)
```

- Summarizing by categories using **group_by()** and **summarize()** functions.

  Example: Suppose now you want to summarize the dataframe **energy_np_in** by calculating max, min, and average values of **ren_ele_share** in Nepal and India and save summarized dataframe as **energy_summary**.

```
energy_summary <- energy_np_in %>%
  na.omit() %>%
  group_by(country) %>%
  summarize(max = max(ren_ele_share),
            min = min(ren_ele_share),
            mean = mean(ren_ele_share))

View(energy_summary)
```

Practice : Let’s summarize the dataframe **energy** by calculating max, min, and average values of **ele_total** [Access to electricity (% of total population)] for each country.

- **Merging dataframes** using join functions.

inner_join()	Return rows with matching keys in both data frames
left_join()	Return all rows from first data frame, matching rows from second
right_join()	Return all rows from second data frame, matching rows from first
full_join()	Return all rows from both data frames, matching by keys
semi_join()	Return rows from first data frame with matching keys in second
anti_join()	Return rows from first data frame without matching keys in second

```
df1 <- data.frame(id = c(1, 2, 3), colA = c("A", "B", "C"))
df2 <- data.frame(id = c(1, 3, 5), colB = c("X", "Y", "Z"))
print(df1)
print(df2)

inner_join(df1, df2, by = 'id')
left_join(df1, df2, by = 'id')
right_join(df1, df2, by = 'id')
full_join(df1, df2, by = 'id')
semi_join(df1, df2, by = 'id')
anti_join(df1, df2, by = 'id')
```

Practice :

- Let's left_join dataframes **energy** and **classf** by common column **ccode**.
- Summarize by calculating average values of **ele_total** [Access to electricity (% of total population)] for each **year** and **country** group (i.e., H, UM, LM, L).
- Save the summarized dataframe as **wb_energy**.

```
wb_energy <- left_join(energy, classf, by = 'ccode') %>%
  na.omit() %>%
  group_by(wb_class) %>%
  summarize(average = mean(ele_total))
View(wb_energy)
```