Dplyr (“dee-ply-er”)

Data Manipulation in R Using dplyr

- dplyr aims to provide a function for each basic verb of data manipulation. These verbs can be organised into three categories based on the component of the dataset that they work with:
- Rows:
  - filter() chooses rows based on column values.
  - arrange() changes the order of the rows.
- Columns:
  - select() return the subset of the columns of a data frame.
  - mutate() changes the values of columns and creates new columns.
  - rename() renames the column
- Groups of rows:
  - summarise() collapses a group into a single row.

- **The pipe**
- All of the dplyr functions take a data frame (or tibble) as the first argument.
- Rather than forcing the user to either save intermediate objects or nest functions, dplyr provides the %>% operator from magrittr. x %>% f(y) turns into f(x, y) so the result from one step is then “piped” into the next step. You can use the pipe to rewrite multiple operations that you can read left-to-right, top-to-bottom (reading the pipe operator as “then”).

```
install.packages("dplyr") # once per machine
library("dplyr")
```

- library(dplyr)
- ## let's first find some insight into data set
- dim(airquality)
- ## [1] 153   6
- str(airquality)
- air_quality=airquality

- glimpse(airquality)
- mydata=airquality
- sample_n(mydata,3)
- head(mydata)
- sample_frac(mydata,0.1)
- x1 = distinct(mydata)

Filter()

```
## select all rows where Temp is more than 90 degree and Month equal to 9 (September)
filter(air_quality, Month == 9, Temp >90)
## Source: local data frame [4 x 6]
##
##   Ozone Solar.R  Wind  Temp Month   Day
##   (int)   (int) (dbl) (int) (int) (int)
## 1    96     167   6.9    91     9     1
## 2    78     197   5.1    92     9     2
## 3    73     183   2.8    93     9     3
## 4    91     189   4.6    93     9     4
```

Note : You can use comma or ampresand (&) to represent AND condition.

```
## select all rows where day is less than 5 and Solar radiation greater than or equal
filter(air_quality, Day <5 & Solar.R >= 200)
## Source: local data frame [7 x 6]
##
##   Ozone Solar.R  Wind  Temp Month   Day
##   (int)   (int) (dbl) (int) (int) (int)
## 1    18     313  11.5    62     5     4
## 2    NA     286   8.6    78     6     1
## 3    NA     287   9.7    74     6     2
## 4    NA     242  16.1    67     6     3
## 5   135     269   4.1    84     7     1
## 6    49     248   9.2    85     7     2
## 7    32     236   9.2    81     7     3
```

```
## We want get only those rows where Ozone is not missing
head(filter(air_quality, !is.na(Ozone)), 5)
## Source: local data frame [5 x 6]
##
##   Ozone Solar.R  Wind  Temp Month   Day
##   (int)   (int) (dbl) (int) (int) (int)
## 1    41     190   7.4    67     5     1
## 2    36     118   8.0    72     5     2
## 3    12     149  12.6    74     5     3
## 4    18     313  11.5    62     5     4
## 5    28      NA  14.9    66     5     6
```

arrange()

- arrange () function sorting your data either descending or ascending order
- ## arrange the rows in the ascending order of Day and then in the descending order of Month.
- arrange(air_quality, Day, desc(Month))
- ## Source: local data frame [153 x 6]
- ##
- ## Ozone Solar.R Wind Temp Month Day
- ## (int) (int) (dbl) (int) (int) (int)
- ## 1 96 167 6.9 91 9 1
- ## 2 39 83 6.9 81 8 1
- ## 3 135 269 4.1 84 7 1
- ## 4 NA 286 8.6 78 6 1
- ## 5 41 190 7.4 67 5 1
- ## 6 78 197 5.1 92 9 2
- ## 7 9 24 13.8 81 8 2
- ## 8 49 248 9.2 85 7 2
- ## 9 NA 287 9.7 74 6 2
- ## 10 36 118 8.0 72 5 2

Mutate()

mutate() function generally create add new variable that are function of existing variable.

mutate(air_quality, temp_celsius= (Temp - 32)*5/9)

select

- library(dplyr)
- ## let's first find some insight into data set
- dim(airquality)
- ## [1] 153   6
- str(airquality)
- air_quality=airquality
- # select column by name
- head(select(airquality, Ozone,Day, Month), 3)
- ## Source: local data frame [6 x 3]
- select(airquality, Ozone:Wind)

- select(air_quality, -Solar.R)

- select(air_quality,contains("o"))

1. ends_with() = Select columns that end with a character string
2. starts_with() = Select columns that start with a character string
3. matches() = Select columns that match a regular expression
4. one_of() = Select columns names that are from a group of names

rename() syntax :

rename(data , new_name = old_name)

data : Data Frame

new_name : New variable name you want to keep

old_name : Existing Variable Name

Summarise()

summarise() function is mainly useful with data that has been grouped by one or more variable.

```
## compute the max and min temerature
summarise(air_quality, max_temp= max(Temp), min_temp = min(Temp))
## Source: local data frame [1 x 2]
##
##   max_temp min_temp
##      (int)    (int)
## 1       97       56
```

Summarise()

summarise() function is mainly useful with data that has been grouped by one or more variable.

```
## compute the average number of Ozone
summarise(air_quality, median_Oz = median(Ozone, na.rm = TRUE))
## Source: local data frame [1 x 1]
##
##   median_Oz
##       (dbl)
## 1      31.5
```

Pipe - Chaining syntax (%>%):

The important part of dplyr is when you start to “chain” different verbs together. The magrittr R package contain the pipe function %>%.

We use %>% operator to connect one command to another. The output of one command becomes the input for the next command.

- ## compute mean temperature of month where month starts from May to August.

```
> air_quality %>%
+   group_by(Month) %>%
+   filter(Month > 4 & Month <=8) %>%
+   summarise(mean=mean(Temp, na.rm=TRUE))
# A tibble: 4 x 2
  Month  mean
  <int> <dbl>
1     5  65.5
2     6  79.1
3     7  83.9
4     8  84.0
```