#dplyr package

#Dr.Trilok Nath Pandey

#Asst.Prof.

#VIT,SCOPE

#dplyr contains mainly 5 verbs and these verbs

#make up the majority of data manipulation which

#corresponds to common tasks you might to

#perform on a table of data.

##

#Select: return the subset of the columns of a

#data frame. It is fast when you are working

#with large dataset with many columns but

#few are actually of interest to you.

#See ?select for more details.

##

#Filter: filter() function allows to extract a

#subset of rows from a data frame based on

#logical condition. The first argument of filter

#function takes as data frame and the second

#argument as well as subsequent argument takes

#as logical expressions that filter the data

#frame.

##

#Mutate: mutate () function allow to adding new

#variable or transform variable in the data frame.

##

#Arrange: arrange () function allow you to order

#the rows by one or more columns in ascending or

#descending order.

```
################################################
#Summarise: summarise() function allow to
#generate or calculate summary statistics for a
#given columns in the data frame such as finding
#mean, median etc.
################################################
rm(list=ls())
library(dplyr)
library(MASS)
## let's first find some insight into data set
dim(airquality)
help(airquality)
str(airquality)
air_quality=airquality
glimpse(air_quality)
sample_n(air_quality,5)
head(air_quality)
sample_frac(air_quality,0.1)
x1 = distinct(air_quality)
View(air_quality)
View(x1)
dim(x1)
dim(air_quality)
mydata=na.omit(air_quality)
dim(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)

head(filter(air_quality, !is.na(Ozone)), 5)

select all rows where day is less than 5 and

#Solar radiation greater than or equal to 200.

filter(air_quality, Day <5 & Solar.R >= 200)

select all rows where days equal to 1 and 2

#Display five records

head (filter(air_quality, Day %in% c(1,2)),5)

select all rows where Month is August OR

#average Wind speed less than 5

```
head(filter(air_quality, Month==8 | Wind < 5), 5)
```

##

#Arrange

##

arrange the rows in the ascending order of

#Day and then in the descending order of Month.

```
arrange(air_quality, Day, desc(Month))
```

arrange rows in the descending order of Temp

#and display 15 records

```
head (arrange(air_quality, desc(Temp)), 15)




## adds a new column that displays the
#temperature in Celsius




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




## add new column that display the deviation of
#Ozone from mean




head (mutate(air_quality,
```

```
          Dev_Ozone= Ozone-
            mean(Ozone, na.rm = TRUE)))
################################################
#Select
#Use of : (colon) operator offer selecting a
#range of columns by name.



head(select(air_quality, Ozone,Day, Month), 3)



##############################
#ambiguity in select function

#detach MASS package
detach("package:MASS",TRUE)

# # select column by name
#Use of : (colon) operator offer selecting
#a range of columns by name.
head(select(air_quality, Ozone:Wind), 3)




head(select(air_quality, Ozone,Day, Month), 10)
```

```
select(airquality, Ozone:Wind)



## omit multiple column
## select all the columns except specific
#column (you can omit specific variable using
#minus (-) sign in front of variable tells
#select() that we DON'T want the variable column)


select(air_quality, -Solar.R)
head(select(air_quality, -(Temp:Day)), 3)
head(select(air_quality, -c(Ozone,Day)), 3)


#select columns that contains "o" charater
#such as Ozone, Month etc.
select(air_quality,contains("o"))
##############################################
dplyr::select(air_quality, Ozone,Day, Month)
##############################################



#groupby()
##################################################
#A group_by function allow to split
#the data set according to categorical variable.
```

```
#we are going to create temperature categorical
#variable which indicate whether teperature is
#hot or cold depending the temperature was over
#70 degree or not.




air_quality1 <- mutate(air_quality,
                       TempCat = factor
                       ((Temp > 70),
                         labels = c("cold","hot")))



View(air_quality1)
hot_cold <- group_by(air_quality1, TempCat)
head (hot_cold)
##################################################
#Summarizing Data:
#summarise() function is mainly useful with data
#that has been grouped by one or more variable.
#In this process, group_by() function creates the
#group and summarise() function provides
#aggregation to summarise each group.
## compute the average number of Ozone
```

```
summarise(air_quality, median_Oz =
          median(Ozone,na.rm = TRUE))

## compute the max and min temerature




summarise(air_quality, max_temp= max(Temp),
         min_temp = min(Temp))

## summarize hot_cold object which split
#according to tempreture.
#So we are going to show what is mean Ozone,
#what is median Solar radiation and what is
#max Wind during hot and cold day.



summarise(hot_cold, mean_Ozone=
           mean(Ozone, na.rm = TRUE),
         med_solar =
           median(Solar.R, na.rm = TRUE),
         max_wind = max(Wind, na.rm = TRUE))


## compute the average number of  Fahrenheit
#degree and Celsius degree of Temprature
```

```
#according to month, apply the mean()
#function to the column Temp and call the
#summary value mean_temp and mean_in_celsius .
```

```
Month_Cat <- group_by(air_quality, Month)
View(Month_Cat)
summarise (Month_Cat, mean_temp =
             mean(Temp, na.rm = TRUE),
           mean_in_celsius = (mean_temp-32)*5/9)
```

```
#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. It is very useful when you are
#performing several operations on data,
#and don't want to save the output at each
#intermediate step.


## select Ozone and wind from airquality
#of data set.

air_quality %>%
  select(Ozone, Wind) %>%
  head

## we want to see, what is the average number
#of Ozone in last 6 days of May Month

air_quality %>%
  filter(Month== 5 & Day > 25) %>%
  summarise(Ozone = mean(Ozone, na.rm=TRUE)) %>%
  head

## 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)) %>%
```

head()