DataScience Made Simple

Select Random Samples in R using Dplyr – (sample\_n() and sample\_frac())

Sample\_n() and Sample\_frac () are the functions used to select random samples in R using Dplyr Package.  Dplyr package in R is provided with sample\_n() function which selects random n rows from a data frame.  Sample\_frac () in R returns the random N% of rows.

We will be using mtcars data to depict the above functions

**sample\_n() Function in Dplyr**

The **sample\_n**function selects random rows from a data frame (or table). First parameter contains the data frame name, the second parameter of the function tells R the number of rows to select.

library(dplyr)

mydata <- mtcars

# select random 4 rows of the dataframe

sample\_n(mydata,4)

#### ****sample\_frac() Function in Dplyr :****

The **sample\_frac()**function selects random n percentage of rows from a data frame (or table). First parameter contains the data frame name, the second parameter tells what percentage of rows to select

library(dplyr)

mydata <- mtcars

# select random 20 percentage rows of the dataframe

sample\_frac(mydata,0.2)

# Remove Duplicate rows in R using Dplyr – distinct () function

Distinct function in R is used to remove duplicate rows in R using Dplyr package.  Dplyr package in R is provided with distinct() function  which eliminate duplicates rows with single variable or with multiple variable.

We will be using mtcars data to depict the above functions

#### ****distinct() Function in Dplyr  –  Remove duplicate rows of a dataframe:****

library(dplyr)

mydata <- mtcars

# Remove duplicate rows of the dataframe

distinct(mydata)

#### ****Remove Duplicate Rows based on a variable****

We will be removing duplicate rows using a particular variable.

library(dplyr)

mydata <- mtcars

# Remove duplicate rows of the dataframe using carb variable

distinct(mydata,carb, .keep\_all= TRUE)

#### ****Remove Duplicate Rows based on multiple variables****

We will be removing duplicate rows using Multiple variables in the below example.

library(dplyr)

mydata <- mtcars

# Remove duplicate rows of the dataframe using cyl and vs variables

distinct(mydata, cyl,vs, .keep\_all= TRUE)

The **.keep\_all**function is used to retain all other variables in the output data frame. So the resultant dataframe will be

# Select variables (columns) in R using Dplyr – select () Function

Select function in R is used to select variables (columns) in R using Dplyr package.  Dplyr package in R is provided with select() function which select the columns based on conditions.

We will be using mtcars data to depict the select() function

#### ****select () Function in Dplyr:****

library(dplyr)

mydata <- mtcars

# Select columns of the dataframe

select(mydata,mpg,cyl,wt)

#### ****Select 3rd and 4th columns of the dataframe:****

library(dplyr)

mydata <- mtcars

# Select 3rd and 4th columns of the dataframe

select(mydata,3:4)

#### ****starts\_with() function:****

Select the column name which starts with mpg

library(dplyr)

mydata <- mtcars

# Select on columns names of the dataframe which starts with

select(mydata,starts\_with("mpg"))

Select the column names which does not starts with mpg

library(dplyr)

mydata <- mtcars

# deselect on columns names of the dataframe which starts with

select(mydata,-starts\_with("mpg"))

#### ****ends\_with() function:****

Select the column name which ends with cyl

library(dplyr)

mydata <- mtcars

# Select on columns names of the dataframe which ends with

select(mydata,ends\_with("cyl"))

#### ****contains() function:****

Select the column name which contains “s”

library(dplyr)

mydata <- mtcars

# Select on columns names of the dataframe which contains

select(mydata,contains("s"))

#### ****matches() function:****

Select the column name which matches with “di”

library(dplyr)

mydata <- mtcars

# Select on columns names of the dataframe which matches

select(mydata,matches("di"))

#### ****everything() function:****

select everything /all columns of the dataframe

library(dplyr)

mydata <- mtcars

# Select everything

select(mydata,everything())

# Drop column in R using Dplyr – drop variables

Drop column in R using Dplyr: Drop column in R can be done by using minus before the select function.  Dplyr package in R is provided with select() function which is used to select or drop the columns based on conditions.

We will be using mtcars data to depict, dropping of the variable.

#### ****Drop by column names in Dplyr:****

select() function along with minus which is used to drop the columns by name

library(dplyr)

mydata <- mtcars

# Drop the columns of the dataframe

select (mydata,-c(mpg,cyl,wt))

#### ****Drop 3rd, 4th and 5th columns of the dataframe:****

library(dplyr)

mydata <- mtcars

# Drop 3rd,4th and 5th columns of the dataframe

select(mydata,-c(3,4,5))

#### ****starts\_with() function:****

Drop the column name which starts with mpg

library(dplyr)

mydata <- mtcars

# Drop column names of the dataframe which starts with

select(mydata,-starts\_with("mpg"))

#### ****ends\_with() function:****

Drop the column name which ends with cyl

library(dplyr)

mydata <- mtcars

# Drop column names of the dataframe which ends with

select(mydata,-ends\_with("cyl"))

#### ****contains() function:****

Drop the column name which contains “s”

library(dplyr)

mydata <- mtcars

# drop the column names of the dataframe which contains

select(mydata,-contains("s"))

#### ****matches() function:****

Drop the column name which matches with “di”

library(dplyr)

mydata <- mtcars

# Drop the columns names of the dataframe which matches

select(mydata,-matches("di"))

# Re arrange or Re order the column of dataframe in R using Dplyr

Re Arranging or Re order the column of dataframe in R using Dplyr. Dplyr package in R is provided with select() function which re orders the columns.

We will be using mtcars data to depict the re order of variable.

#### ****Re arrange or Re order  the column of dataframe in R using Dplyr:****

Re order the column using select function with gear,hp,qsec,vs columns arranged in order.

library(dplyr)

mydata <- mtcars

# Reorder the columns of the dataframe

Mydata1 = select(mydata, gear,hp,qsec,vs, everything())

Mydata1

Re order only with carb and cyl columns

library(dplyr)

mydata <- mtcars

# Reorder the columns of the dataframe

Mydata1 = select(mydata, carb,cyl)

Mydata1

# Rename the column name in R using Dplyr

Rename the column name in R using Dplyr. Dplyr package in R is provided with rename() function which re names the column name or column variable.

We will be using mtcars data to depict the re name of column variable.

#### ****Rename the column name in R using Dplyr:****

Rename the column name using rename function in dplyr.

library(dplyr)

mydata <- mtcars

# Rename the column name of the dataframe

Mydata1 = rename(mydata, displacement=disp, cylinder=cyl)

Mydata1

# Filter or subsetting rows in R using Dplyr

Filter or subsetting rows in R using Dplyr can be easily achieved. Dplyr package in R is provided with filter() function which subsets the rows with multiple conditions.

We will be using mtcars data to depict the example of filtering or subsetting.

#### ****Filter or subsetting the rows in R using Dplyr:****

Subset using filter() function.

library(dplyr)

mydata <- mtcars

# subset the rows of dataframe with condition

Mydata1 = filter(mydata,cyl==6)

Mydata1

Only the rows with cyl =6 is filtered

#### ****Filter or subsetting the rows in R with multiple conditions using Dplyr:****

library(dplyr)

mydata <- mtcars

# subset the rows of dataframe with multiple conditions

Mydata1 = filter(mydata, gear %in% c(4,5))

Mydata1

The rows with gear=4 or 5 are filtered

#### ****Filter or subsetting the rows in R with multiple conditions (AND) using Dplyr****

library(dplyr)

mydata <- mtcars

# subset the rows of dataframe with multiple conditions

Mydata1 = filter(mydata, gear %in% c(4,5) & carb==2)

Mydata1

The rows with gear= (4 or 5) and carb=2 are filtered

#### ****Filter or subsetting the rows in R with multiple conditions (OR) using Dplyr:****

library(dplyr)

mydata <- mtcars

# subset the rows of dataframe with multiple conditions

Mydata1 = filter(mydata, gear %in% c(4,5) | mpg==21.0)

Mydata1

The rows with gear= (4 or 5)  or mpg=21 are filtered

#### ****Filter or subsetting the rows in R with multiple conditions (NOT) using Dplyr:****

library(dplyr)

mydata <- mtcars

# subset the rows of dataframe with multiple conditions

Mydata1 = filter(mydata, !gear %in% c(4,5))

Mydata1

The rows with gear!=4 or gear!=5 are filtered

#### ****Filter or subsetting the rows in R with Contains condition using Dplyr:****

library(dplyr)

mydata <- mtcars

# subset the rows of dataframe with multiple conditions

Mydata1 = filter(mydata, grepl(0,hp))

Mydata1

hp  which contains value 0 are filtered

# summarise, summarise\_at, summarise\_if, summarise\_all in R- Get the summary of dataset in R using Dplyr

summarise, summarise\_at, summarise\_if, summarise\_all in R – Summary of the dataset (Mean, Median and Mode) in R can be done using Dplyr. Dplyr package in R is provided with summarise() function which gets the summary of dataset in R. Dplyr package has summarise(), summarise\_at(), summarise\_if(), summarise\_all()

We will be using mtcars data to depict the example of summarise function.

#### ****Summary of column in dataset in R using Dplyr – summarise()****

library(dplyr)

mydata <- mtcars

# summarise the columns of dataframe

summarise(mydata, mpg\_mean=mean(mpg),mpg\_median=median(mpg))

#### ****Summary of multiple column of dataset in R using Dplyr – summarise\_at()****

library(dplyr)

mydata <- mtcars

# summarise the list of columns of dataframe

summarise\_at(mydata, vars(mpg, hp), funs(n(), mean, median))

#### ****summarise all numeric variable with summarise\_if():****

The summarise\_if function allows you to summarise conditionally.

library(dplyr)

mydata <- mtcars

# summarise all the list of numeric variable of dataframe

summarise\_if(mydata, is.numeric, funs(n(),mean,median))

summarise\_if() function that gets the number of rows, mean and median of  all the numeric columns.

#### ****summarise\_all()****

The summarise\_all function allows you to summarise all the variables.

|  |
| --- |
| library(dplyr)  mydata <- mtcars    # summarise all the column of dataframe  summarise\_all(mydata,funs(n(),mean,median)) |

summarise\_all() function that gets the number of rows, mean and median of  all the columns.

#### ****Summarize categorical or factor Variable:****

We will be summarizing the number of levels/categories and count of missing observations in a categorical (factor) variable. Let’s use iris dataset for example

library(dplyr)

mydata2 <- iris

summarise\_all(mydata2["Species"], funs(nlevels(.), nmiss=sum(is.na(.))))

# Sorting DataFrame in R using Dplyr – arrange function

Sorting dataframe in R can be done using Dplyr. Dplyr package in R is provided with arrange() function which sorts the dataframe by multiple conditions.

We will be using mtcars data to depict the example of sorting with arrange() function.

library(dplyr)

mydata <- mtcars

# sort the dataframe in R

arrange(mydata,mpg)

#### ****Sorting dataframe in R using multiple variables with Dplyr:****

In this example, we are sorting data by multiple variables. One in descending and one in ascending the example is shown below. We will be using pipe operator  (%>%).

|  |
| --- |
| # sort the dataframe in R using multiple variable with Dplyr  mydata %>% arrange(desc(mpg)) %>% arrange(gear) |

pipe operator first sorts the mpg in descending order format and within that it sorts gear in ascending order format.

# Groupby function in R using Dplyr – group\_by

Groupby Function in R – group\_by is used to group the dataframe in R.  Dplyr package in R is provided with group\_by() function which groups the dataframe by multiple columns with mean, sum or any other functions.

We will be using iris data to depict the example of group\_by() function

library(dplyr)

mydata2 <-iris

# Groupby function for dataframe in R

summarise\_at(group\_by(mydata2,Species),vars(Sepal.Length),funs(mean(.,na.rm=TRUE)))

Mean of Sepal.Length is grouped by Species variable.

#### ****Groupby function in R with dplyr pipe operator %>%:****

library(dplyr)

mydata2 <-iris # Group by function for dataframe in R using pipe operator mydata2 %>% group\_by(Species) %>% summarise\_at(vars(Sepal.Length),funs(sum(.,na.rm=TRUE)))

Sum of Sepal.Length is grouped by Species variable with the help of pipe operator (%>%) in dplyr package.

#### ****Groupby in R without dplyr using aggregate function:****

In this example we will be using aggregate function in R to do group by operation as shown below

mydata2 <-iris

# Group by in R using aggregate function

aggregate(mydata2$Sepal.Length, by=list(Species=mydata2$Species), FUN=sum)

# Windows Function in R using Dplyr

Like SQL, dplyr uses windows function in R that are used to subset data within a group. It returns a vector of values. We could use **min\_rank() function** that calculates rank in the preceding example

We will be using iris data to depict the example of group\_by() function

library(dplyr)

mydata2 <-iris # windows function in R using Dplyr mydata2 %>% select(Species,Sepal.Length) %>%

    group\_by(Species) %>%

    filter(min\_rank(desc(Sepal.Length))<=5)

Let’s say you have a question like, “What are the top 5 sepal lengths based on the species?” To answer this question, we can simply use one of the rank functions called ‘min\_rank()’ from dplyr and call it directly inside the ‘filter()’ function

By the way, I’m using ‘min\_rank()’ function here but there is another rank function called ‘dense\_rank()’ from dplyr. Both functions return the ranking number based on a given measure column (e.g. Sepal.Length), and only the difference is when there are ties like below.



As you can see when two same rank (tie) occurs min\_rank() function skips one rank and assigns next to next rank wheres the dense\_rank() Function assigns subsequent ranks without skipping any rank as shown above

#### ****Types of window functions:****

Ranking and ordering functions: row\_number(), min\_rank (RANK in SQL), dense\_rank(), cume\_dist(), percent\_rank(), and ntile(). These functions all take a vector to order by, and return various types of ranks.

# Mutate Function in R using dplyr – mutate, mutate\_all and mutate\_at – Create new variable

Mutate Function in R (mutate, mutate\_all and mutate\_at) is used to create new variable or column to  the dataframe in R.  Dplyr package in R is provided with mutate(), mutate\_all() and mutate\_at() function which creates the new variable to the dataframe.

**Syntax of mutate function in dplyr:**

mutate(data\_frame, expression(s) )  
or  
data\_frame %>% mutate(expression(s)

library(dplyr)

mydata2 <-iris

# Mutate function for creating new variable to the dataframe in R

mydata3 = mutate(mydata2, sepal\_length\_width\_ratio=Sepal.Length/Sepal.Width)

head(mydata3)

#### ****mutate\_all() Function in R****

mutate\_all() function in R creates new columns for all the available columns here in our example. mutate\_all() function creates 4 new column and get the percentage distribution of sepal length and width, petal length and width.

library(dplyr)

mydata2 <-iris

# Mutate\_all function for creating new variable to the dataframe in R

mydata3 = mutate\_all(mydata2[,-5], funs("percent"= ./100))

head(mydata3)

#### ****mutate\_at() Function in R****

mutate\_at() function in R creates new columns for the specified columns here in our example. mutate\_at() Function get the min\_rank() of sepal length and sepal width .

library(dplyr)

mydata2 <-iris

# mutate\_at() function for creating new variable to the dataframe in R

mydata4 = mutate\_at(mydata2, vars(Sepal.Length,Sepal.Width), funs(Rank=min\_rank(desc(.))))

head(mydata4)

# Union and union\_all Function in R using Dplyr (union of data frames):

**Union and union\_all Function in R :** Union of two data frames in R can be easily achieved by using union Function and union all function in Dplyr package . Dplyr package in R is provided with union(), union\_all() function.

#### ****Union function in R:****

UNION function in R combines all rows from both the tables and removes duplicate records from the combined dataset

#### ****union\_all function in R:****

UNION\_ALL function in R combines all rows from both the tables without removing the duplicate records from the combined dataset.

**Union Function in R example:** First lets create two data frames

#Create two data frames

df1 = data.frame(CustomerId = c(1:6), Product = c(rep("Oven", 3), rep("Television", 3)))

df2 = data.frame(CustomerId = c(4:7), Product = c(rep("Television", 2), rep("Air conditioner", 2)))

**df1 will be**

    CustomerId  Product  
1        1                 Oven  
2        2                 Oven  
3        3                 Oven  
4        4                 Television  
5        5                 Television  
6        6                 Television

**df2 will be**

CustomerId         Product

1          4           Television  
2          5           Television  
3          6          Air conditioner  
4          7          Air conditioner

**Union Function in R :**UNION function in R combines all rows from both the tables and removes duplicate records from the combined dataset. So the resultant dataframe will not have any duplicates.

library(dplyr)

#  union two dataframes  without duplicates

union(df1,df2)

**union\_all Function in R example:**UNION\_ALL function in R combines all rows from both the tables without removing the duplicate records from the combined dataset. So the resultant dataframe will have duplicates.

library(dplyr)

#  union two dataframes  with duplicates

union\_all(df1,df2)

# Intersect Function in R using Dplyr (intersection of data frames)

Intersection of two data frames can be easily achieved by using intersect Function in R Dplyr package . Dplyr package in R is provided with intersect() function

#### ****intersect function in R:****

intersect function in R takes the rows that appear in both the tables

**R intersect Function  example:** First lets create two data frames

#Create two data frames

df1 = data.frame(CustomerId = c(1:6), Product = c(rep("Oven", 3), rep("Television", 3)))

df2 = data.frame(CustomerId = c(4:7), Product = c(rep("Television", 2), rep("Air conditioner", 2)))

library(dplyr)

#  intersect two dataframes

intersect(df1,df2)

# Setdiff() Function in R using Dplyr (get difference of dataframes)

To get the difference of two data frames  i.e. To get the row present in one table which is not in other table we will be using setdiff() function in R ‘s Dplyr package . Dplyr package in R is provided with setdiff() function which gets the difference of two dataframe.

#### ****setdiff() function in R:****

setdiff function in R takes the rows that appear in one tables but not in other

**setdiff Function in R example:** First lets create two data frames

#Create two data frames

df1 = data.frame(CustomerId = c(1:6), Product = c(rep("Oven", 3), rep("Television", 3)))

df2 = data.frame(CustomerId = c(4:7), Product = c(rep("Television", 2), rep("Air conditioner", 2)))

Setdiff function  in R takes the rows that appear in first table but not in second table and creates the dataframe.

library(dplyr)

#  difference of two dataframes

setdiff(df1,df2)

# Case when in R using case\_when() Dplyr – case\_when in R

Case when in R can be executed with case\_when() function in dplyr package. Dplyr package is provided with case\_when() function which is similar to case when statement in SQL.

We will be using iris data to depict the example of case\_when() function.

library(dplyr)

mydata2 <-iris

head(mydata2)

We will be creating additional variable **species\_new** using mutate function and case when statement.

mydata2 %>% mutate(species\_new = case\_when(is.na(Species) ~ "missing",

                                           Species=="setosa" ~ "setosa\_new",

                                           Species=="versicolor" ~ "versicolor\_new",

                                           Species=="virginica" ~ "virginica\_new",

                                           TRUE ~ "others"))

* you can use variables directly within case\_when() wrapper.
* **TRUE**equivalent to ELSE statement .

**NOTE:**  Make sure you set **is.na()** condition at the beginning of R case\_when to handle the missing values.

# Row wise operation in R using Dplyr

Row wise operation in R can be performed using rowwise() function in dplyr package. Dplyr package is provided with rowwise() function with which we will be doing row wise maximum or row wise minimum operations.

We will be using iris data to depict the example of rowwise() function.

|  |
| --- |
| library(dplyr)    mydata2 <-iris  head(mydata2) |

iris data will be looking l

We will be creating additional variable **row\_max** using mutate function and rowwise() function to store the row wise maximum variable.

df1 = mydata2 %>%

  rowwise() %>% mutate(row\_max= max(Sepal.Length:Petal.Width))

head(df1)

# Calculate percentile, quantile, N tile of dataframe in R using dplyr (create column with percentile rank)

Quantile, Decile and Percentile can be calculated using ntile() Function in R. Dplyr package is provided with mutate() function and ntile() function. The **ntile()**function is used to divide the data into N bins. In this example we will be creating the column with percentile, decile and quantile rank

#### ****Quantile rank in R:****

We will be using mtcars data to depict the example of ntile() function.

library(dplyr)

mydata<-mtcars

df1 = mutate(mydata, quantile\_rank = ntile(mydata$mpg,4))

df1

#### ****Decile rank in R:****

|  |
| --- |
| library(dplyr)    mydata<-mtcars  df1 = mutate(mydata, decile\_rank = ntile(mydata$mpg,10))  df1 |

So in the resultant data frame decile rank is calculated and populated across

#### ****Percentile rank in R:****

library(dplyr)

mydata<-mtcars

df1 = mutate(mydata, percentile\_rank = ntile(mydata$mpg,100))

df1