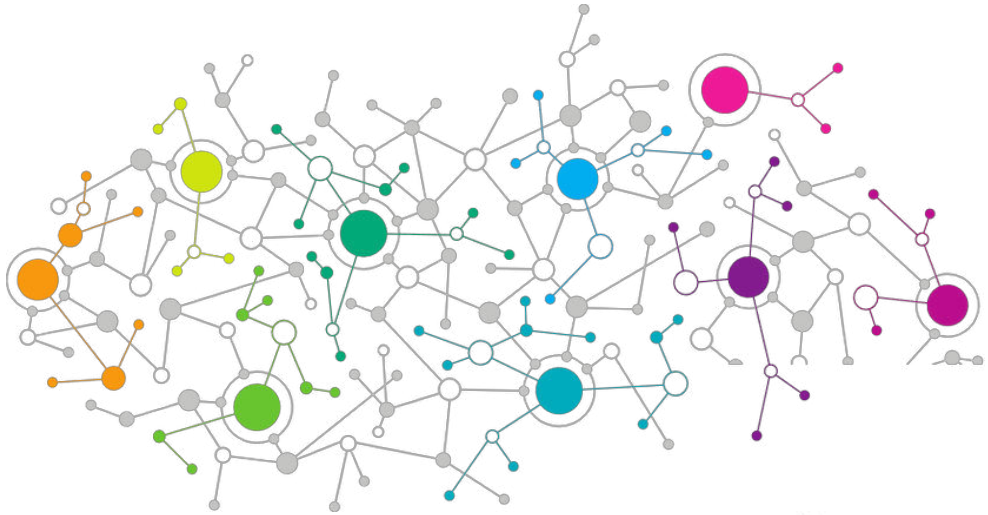
IntelliPaat

# Data Science

## Data Manipulation



# Agenda

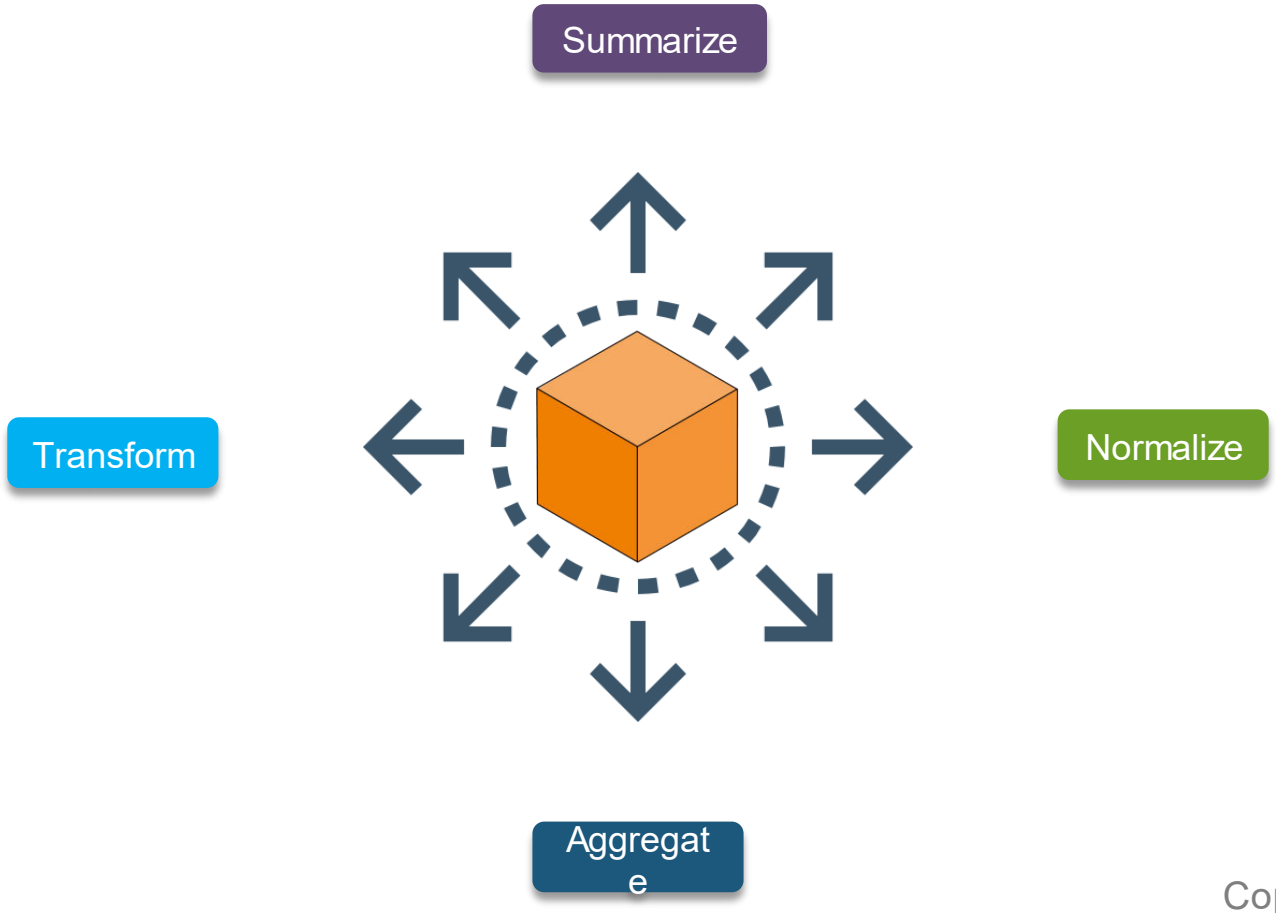
01 Data Manipulation

02 Apply family of Functions

03 DPLYR Package

# Data Manipulation

Data Manipulation is the process of changing data in order to make it more organized and also to find insights from the data



# Apply family of functions

# Apply Family

The apply family consists of vectorized functions which minimize your need to explicitly create loops

| | |
|---|---|
| apply() | For matrices and Dataframes |
| lapply() | For lists. The output is given out as list |
| sapply() | For lists. A simplified output is given |
| tapply() | For Vectors. Useful when we need to break up a vector into groups |
| mapply() | Multivariate version of sapply() |

# apply() Function

The apply() function is most often used to apply a function to the rows or columns (margins) of matrices or data frames

apply(x, MARGIN, FUN, ...)

```
> mat1
     [,1] [,2]
[1,]    1    3
[2,]    2    4
```

apply(m,1,sum)

```
> apply(m,1,sum)
[1] 4 6
```

```
> mat1
     [,1] [,2]
[1,]    1    3
[2,]    2    4
```

apply(m,2,sum)

```
> apply(m,2,sum)
[1] 3 7
```

# lapply() Function

The lapply() function does the following simple series of operations:

- It loops over a list, iterating over each element in that list
- It applies a function to each element of the list (a function that you specify)
- and returns a list (the l is for “list”)

apply(x, FUN, ...)

```
> my_list
$`a`
[1] 1 1

$b
[1] 2 2

$c
[1] 3 3
```

lapply(my_list, mean)

```
> lapply(my_list, mean)
$`a`
[1] 1

$b
[1] 2

$c
[1] 3
```

# sapply() Function

sapply() is the same as lapply, but returns a vector instead of a list

sapply(x, FUN, ...)

```
> my_list
$`a`
[1] 1 1

$b
[1] 2 2

$c
[1] 3 3
```

sapply(my_list, mean)

```
> sapply(my_list, mean)
a b c
1 2 3
```

# tapply() Function

tapply() splits the array based on specified data, usually factor levels and then applies the function to it

```
tapply(x, INDEX, FUN, ...)
```

| Dependents | tenure |
|---|---|
| No | 1 |
| No | 34 |
| No | 2 |
| No | 45 |
| No | 2 |
| No | 8 |
| Yes | 22 |
| No | 10 |

```
tapply(customer_churn$tenure, customer_churn$Partner, mean)
```

```
      No      Yes
23.35787 42.01764
```

# mapply() Function

mapply is a multivariate version of sapply. It will apply the specified function to the first element of each argument first, followed by the second element, and so on

mapply(FUN, ...)

```
> x
[1] 1 2 3 4 5
> b
[1]  6  7  8  9 10
```

mapply(sum, x, b)

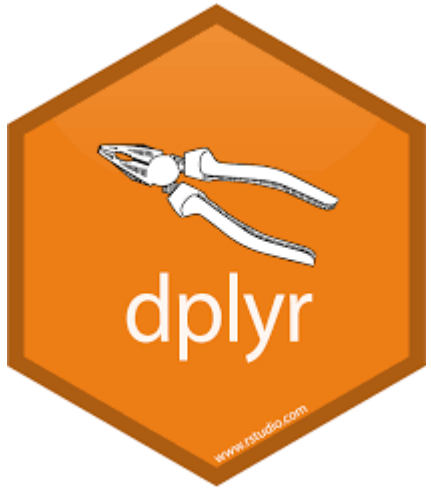
```
> mapply(sum, x, b)
[1]  7  9 11 13 15
```

It adds 1 with 6, 2 with 7, and so on

# dplyr Package

# dplyr Package

The R package *dplyr* is an extremely useful resource for data cleaning, manipulation and analysis

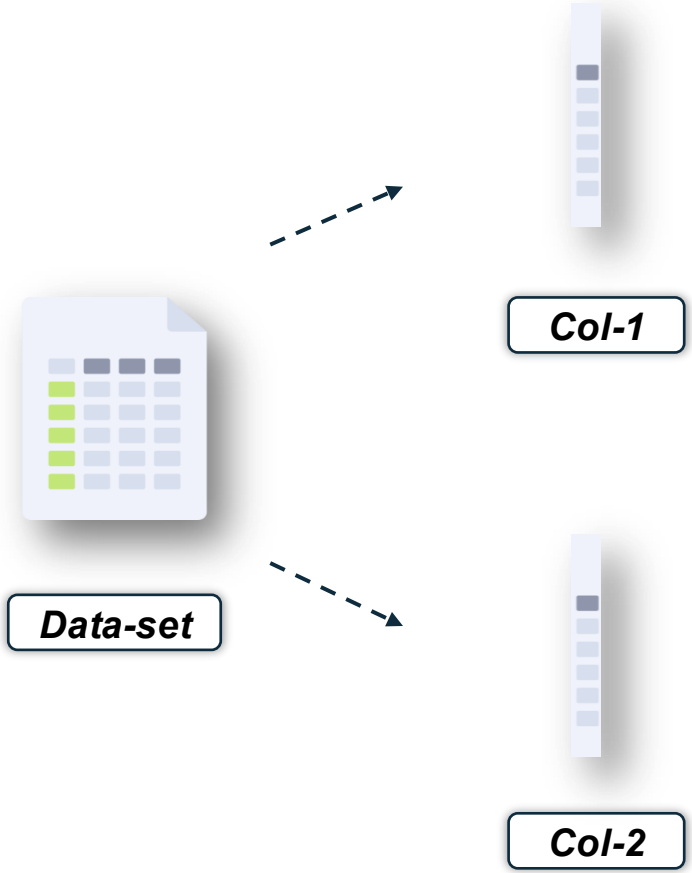



- select()
- filter()
- mutate()
- Sample_n()
- Sample_frac()
- summarise()

# select()

# Select()

select() function helps in selecting individual columns from the data.frame



# select() syntax

select(dataframe_name,col_num1,col_num2…..col_num_n)

select(customer_churn,1,3,6)

| customerID | gender | SeniorCitizen | Partner | Dependents | tenure | PhoneService |
|---|---|---|---|---|---|---|
| 7590-VHVEG | Female | 0 | Yes | No | 1 | No |
| 5575-GNVDE | Male | 0 | No | No | 34 | Yes |
| 3668-QPYBK | Male | 0 | No | No | 2 | Yes |
| 7795-CFOCW | Male | 0 | No | No | 45 | No |
| 9237-HQITU | Female | 0 | No | No | 2 | Yes |
| 9305-CDSKC | Female | 0 | No | No | 8 | Yes |
| 1452-KIOVK | Male | 0 | No | Yes | 22 | Yes |
| 6713-OKOMC | Female | 0 | No | No | 10 | No |
| 7892-POOKP | Female | 0 | Yes | No | 28 | Yes |
| 6388-TABGU | Male | 0 | No | Yes | 62 | Yes |
| 9763-GRSKD | Male | 0 | Yes | Yes | 13 | Yes |

| customerID | tenure | SeniorCitizen |
|---|---|---|
| 7590-VHVEG | 1 | 0 |
| 5575-GNVDE | 34 | 0 |
| 3668-QPYBK | 2 | 0 |
| 7795-CFOCW | 45 | 0 |
| 9237-HQITU | 2 | 0 |
| 9305-CDSKC | 8 | 0 |
| 1452-KIOVK | 22 | 0 |
| 6713-OKOMC | 10 | 0 |
| 7892-POOKP | 28 | 0 |
| 6388-TABGU | 62 | 0 |
| 9763-GRSKD | 13 | 0 |

# select() syntax

select(dataframe_name,col_num1:col_num_n)

select(customer_churn,2:5)

| customerID | gender | SeniorCitizen | Partner | Dependents | tenure | PhoneService |
|---|---|---|---|---|---|---|
| 7590-VHVEG | Female | 0 | Yes | No | 1 | No |
| 5575-GNVDE | Male | 0 | No | No | 34 | Yes |
| 3668-QPYBK | Male | 0 | No | No | 2 | Yes |
| 7795-CFOCW | Male | 0 | No | No | 45 | No |
| 9237-HQITU | Female | 0 | No | No | 2 | Yes |
| 9305-CDSKC | Female | 0 | No | No | 8 | Yes |
| 1452-KIOVK | Male | 0 | No | Yes | 22 | Yes |
| 6713-OKOMC | Female | 0 | No | No | 10 | No |
| 7892-POOKP | Female | 0 | Yes | No | 28 | Yes |
| 6388-TABGU | Male | 0 | No | Yes | 62 | Yes |
| 9763-GRSKD | Male | 0 | Yes | Yes | 13 | Yes |

| gender | SeniorCitizen | Partner | Dependents |
|---|---|---|---|
| Female | 0 | Yes | No |
| Male | 0 | No | No |
| Male | 0 | No | No |
| Male | 0 | No | No |
| Female | 0 | No | No |
| Female | 0 | No | No |
| Male | 0 | No | Yes |
| Female | 0 | No | No |
| Female | 0 | Yes | No |
| Male | 0 | No | Yes |
| Male | 0 | Yes | Yes |

# select() syntax

select(dataframe_name,column_name)

select(customer_churn,Partner)

| customerID | gender | SeniorCitizen | Partner | Dependents | tenure | PhoneService |
|---|---|---|---|---|---|---|
| 7590-VHVEG | Female | 0 | Yes | No | 1 | No |
| 5575-GNVDE | Male | 0 | No | No | 34 | Yes |
| 3668-QPYBK | Male | 0 | No | No | 2 | Yes |
| 7795-CFOCW | Male | 0 | No | No | 45 | No |
| 9237-HQITU | Female | 0 | No | No | 2 | Yes |
| 9305-CDSKC | Female | 0 | No | No | 8 | Yes |
| 1452-KIOVK | Male | 0 | No | Yes | 22 | Yes |
| 6713-OKOMC | Female | 0 | No | No | 10 | No |
| 7892-POOKP | Female | 0 | Yes | No | 28 | Yes |
| 6388-TABGU | Male | 0 | No | Yes | 62 | Yes |
| 9763-GRSKD | Male | 0 | Yes | Yes | 13 | Yes |

| Partner |
|---|
| Yes |
| No |
| No |
| No |
| No |
| No |
| No |
| No |
| Yes |
| No |
| Yes |

# starts_with() & ends_with()

select(dataframe_name,starts_with(“”))

select(customer_churn,starts_with(“P”))



Extract all columns which start with “P”

select(dataframe_name,ends_with(“”))

select(customer_churn,ends_with(“e”))

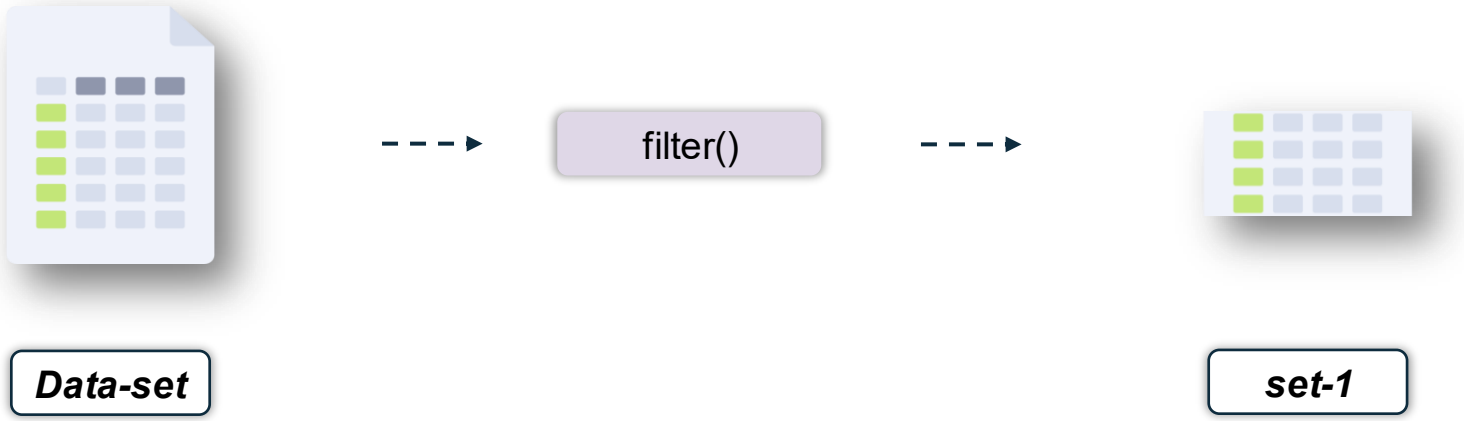


Extract all columns which end with “e”

# filter()

# filter()

filter() function helps in filtering out records on the basis of a condition



# filter() syntax

filter(data-frame_name, condition)

| customerID | gender | SeniorCitizen | Partner | Dependents | tenure | PhoneService |
|---|---|---|---|---|---|---|
| 7590-VHVEG | Female | 0 | Yes | No | 1 | No |
| 5575-GNVDE | Male | 0 | No | No | 34 | Yes |
| 3668-QPYBK | Male | 0 | No | No | 2 | Yes |
| 7795-CFOCW | Male | 0 | No | No | 45 | No |
| 9237-HQITU | Female | 0 | No | No | 2 | Yes |
| 9305-CDSKC | Female | 0 | No | No | 8 | Yes |
| 1452-KIOVK | Male | 0 | No | Yes | 22 | Yes |
| 6713-OKOMC | Female | 0 | No | No | 10 | No |
| 7892-POOKP | Female | 0 | Yes | No | 28 | Yes |
| 6388-TABGU | Male | 0 | No | Yes | 62 | Yes |
| 9763-GRSKD | Male | 0 | Yes | Yes | 13 | Yes |

Customer_churn dataframe

filter(customer_churn,gender==“female”)

# filter() syntax

filter(data-frame_name, condition1 & condition2)

filter(data-frame_name, condition1 | condition2)

filter(customer_churn,gender==“female” & tenure>50)

filter(customer_churn,gender==“male” | SeniorCitizen==1)

| customerID | gender | SeniorCitizen | Partner | Dependents | tenure | PhoneService |
|---|---|---|---|---|---|---|
| 7590-VHVEG | Female | 0 | Yes | No | 1 | No |
| 5575-GNVDE | Male | 0 | No | No | 34 | Yes |
| 3668-QPYBK | Male | 0 | No | No | 2 | Yes |
| 7795-CFOCW | Male | 0 | No | No | 45 | No |
| 9237-HQITU | Female | 0 | No | No | 2 | Yes |
| 9305-CDSKC | Female | 0 | No | No | 8 | Yes |
| 1452-KIOVK | Male | 0 | No | Yes | 22 | Yes |
| 6713-OKOMC | Female | 0 | No | No | 10 | No |
| 7892-POOKP | Female | 0 | Yes | No | 28 | Yes |
| 6388-TABGU | Male | 0 | No | Yes | 62 | Yes |
| 9763-GRSKD | Male | 0 | Yes | Yes | 13 | Yes |

| customerID | gender | SeniorCitizen | Partner | Dependents | tenure | PhoneService |
|---|---|---|---|---|---|---|
| 7590-VHVEG | Female | 0 | Yes | No | 1 | No |
| 5575-GNVDE | Male | 0 | No | No | 34 | Yes |
| 3668-QPYBK | Male | 0 | No | No | 2 | Yes |
| 7795-CFOCW | Male | 0 | No | No | 45 | No |
| 9237-HQITU | Female | 0 | No | No | 2 | Yes |
| 9305-CDSKC | Female | 0 | No | No | 8 | Yes |
| 1452-KIOVK | Male | 0 | No | Yes | 22 | Yes |
| 6713-OKOMC | Female | 0 | No | No | 10 | No |
| 7892-POOKP | Female | 0 | Yes | No | 28 | Yes |
| 6388-TABGU | Male | 0 | No | Yes | 62 | Yes |
| 9763-GRSKD | Male | 0 | Yes | Yes | 13 | Yes |

# mutate()

# filter()

mutate() function helps in adding new columns on the basis of a condition

# mutate() syntax

mutate(data-frame_name, new_column_name = condition)

Student data-frame

| Name | Sub1 | Sub2 | Sub3 |
|---|---|---|---|
| Sam | 23 | 86 | 78 |
| Bob | 67 | 45 | 65 |
| Matt | 21 | 43 | 83 |

mutate(student, Total = Sub1+Sub2+Sub3) -> student

Student data-frame with added column

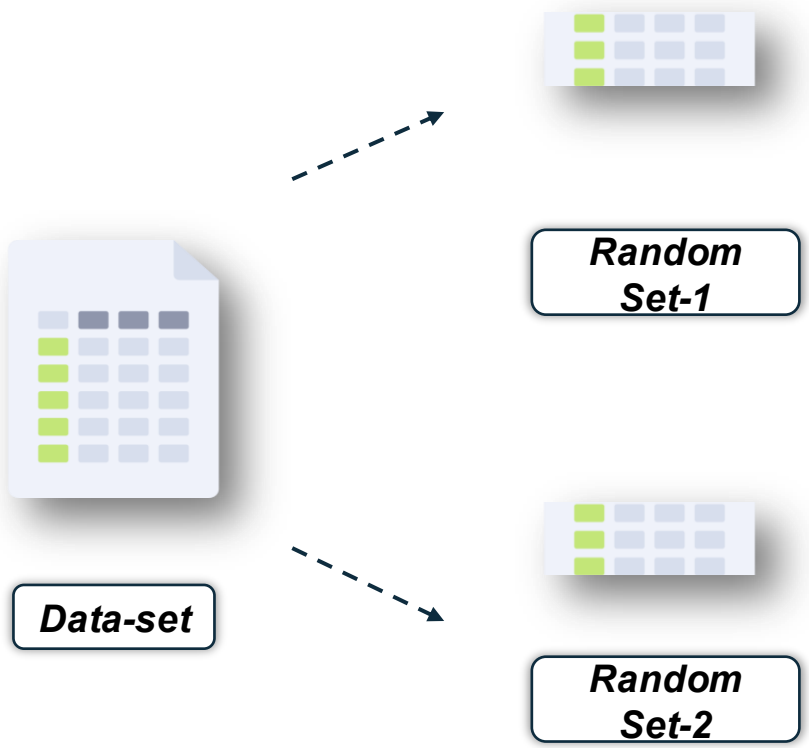
| Name | Sub1 | Sub2 | Sub3 | Total |
|---|---|---|---|---|
| Sam | 23 | 86 | 78 | 187 |
| Bob | 67 | 45 | 65 | 177 |
| Matt | 21 | 43 | 83 | 147 |

Add new column "***Total***"

# sample_n() & sample_frac()

# sample_n() & sample_frac()

sample_n() & sample_frac() help in random sampling of the data



# sample_n()

sample_n() function is used to randomly sample "n" rows from the entire data-frame

sample_n(dataframe_name, sample_size )

| customerID | gender | SeniorCitizen | Partner | Dependents | tenure | PhoneService |
|---|---|---|---|---|---|---|
| 7590-VHVEG | Female | 0 | Yes | No | 1 | No |
| 5575-GNVDE | Male | 0 | No | No | 34 | Yes |
| 3668-QPYBK | Male | 0 | No | No | 2 | Yes |
| 7795-CFOCW | Male | 0 | No | No | 45 | No |
| 9237-HQITU | Female | 0 | No | No | 2 | Yes |
| 9305-CDSKC | Female | 0 | No | No | 8 | Yes |
| 1452-KIOVK | Male | 0 | No | Yes | 22 | Yes |
| 6713-OKOMC | Female | 0 | No | No | 10 | No |
| 7892-POOKP | Female | 0 | Yes | No | 28 | Yes |
| 6388-TABGU | Male | 0 | No | Yes | 62 | Yes |
| 9763-GRSKD | Male | 0 | Yes | Yes | 13 | Yes |

Customer_churn dataframe

sample_n(customer_churn, 2)

| | | | | | | |
|---|---|---|---|---|---|---|
| 9237-HQITU | Female | 0 | No | No | 2 | Yes |
| 7892-POOKP | Female | 0 | Yes | No | 28 | Yes |

# sample_frac()

sample_frac() function is used to randomly sample a ***fraction of rows*** from the entire data-frame

sample_frac(dataframe_name, fraction_value )

| customerID | gender | SeniorCitizen | Partner | Dependents | tenure | PhoneService |
|---|---|---|---|---|---|---|
| 7590-VHVEG | Female | 0 | Yes | No | 1 | No |
| 5575-GNVDE | Male | 0 | No | No | 34 | Yes |
| 3668-QPYBK | Male | 0 | No | No | 2 | Yes |
| 7795-CFOCW | Male | 0 | No | No | 45 | No |
| 9237-HQITU | Female | 0 | No | No | 2 | Yes |
| 9305-CDSKC | Female | 0 | No | No | 8 | Yes |
| 1452-KIOVK | Male | 0 | No | Yes | 22 | Yes |
| 6713-OKOMC | Female | 0 | No | No | 10 | No |
| 7892-POOKP | Female | 0 | Yes | No | 28 | Yes |
| 6388-TABGU | Male | 0 | No | Yes | 62 | Yes |
| 9763-GRSKD | Male | 0 | Yes | Yes | 13 | Yes |

Customer_churn dataframe

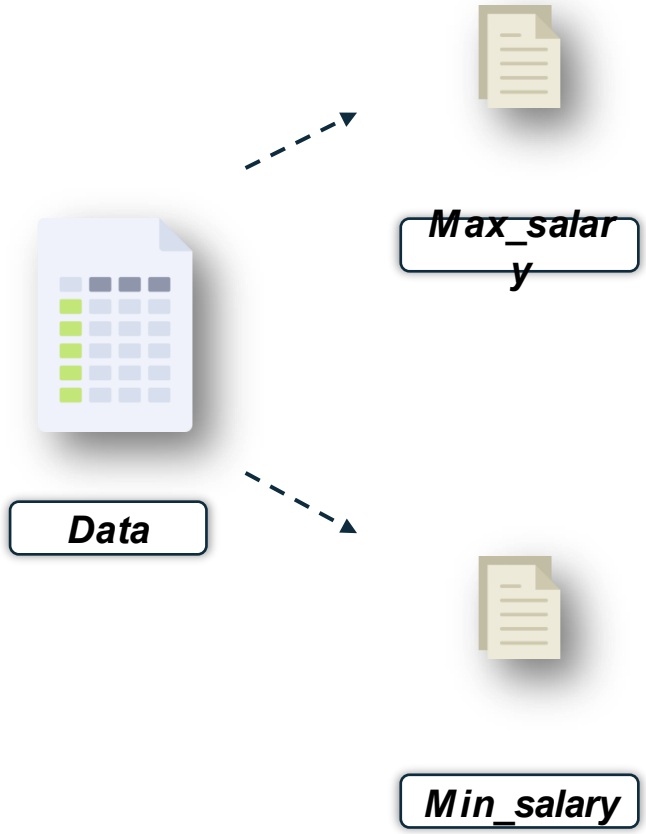
sample_frac(customer_churn, .3)

| | | | | | | |
|---|---|---|---|---|---|---|
| 9237-HQITU | Female | 0 | No | No | 2 | Yes |
| 7892-POOKP | Female | 0 | Yes | No | 28 | Yes |
| 7892-POOKP | Female | 0 | Yes | No | 28 | Yes |
| 7892-POOKP | Female | 0 | Yes | No | 28 | Yes |

# summarise()

# summarise()

summarise() function helps in getting a summarized result



# summarise()

summarise(dataframe_name, function )

| SeniorCitizen | Partner | Dependents | tenure | PhoneService |
|---|---|---|---|---|
| 0 | Yes | No | 1 | No |
| 0 | No | No | 34 | Yes |
| 0 | No | No | 2 | Yes |
| 0 | No | No | 45 | No |
| 0 | No | No | 2 | Yes |
| 0 | No | No | 8 | Yes |
| 0 | No | Yes | 22 | Yes |
| 0 | No | No | 10 | No |

Customer_churn dataframe

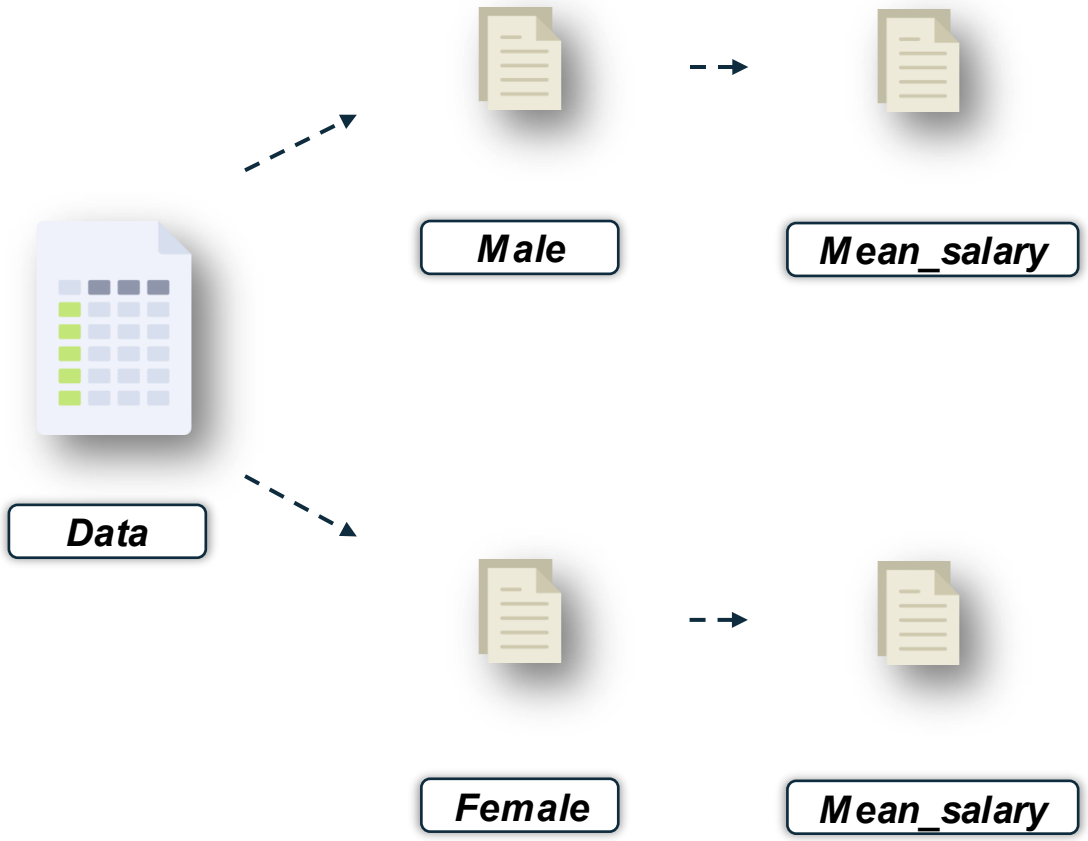
summarise(customer_churn,mean_tenure=mean(tenure))

```
mean_tenure
   32.37115
```

# group_by()

# group_by()

group_by() helps in getting a summarized result with respect to the segregated groups



# group_by()

summarise(group_by(data.frame_name,column_name),function)

| SeniorCitizen | Partner | Dependents | tenure | PhoneService |
|---|---|---|---|---|
| 0 | Yes | No | 1 | No |
| 0 | No | No | 34 | Yes |
| 0 | No | No | 2 | Yes |
| 0 | No | No | 45 | No |
| 0 | No | No | 2 | Yes |
| 0 | No | No | 8 | Yes |
| 0 | No | Yes | 22 | Yes |
| 0 | No | No | 10 | No |

Customer_churn dataframe

summarise(group_by(customer_churn,InternetService),mean(tenure))

| InternetService | `mean(tenure)` |
|---|---|
| <fct> | <dbl> |
| DSL | 32.8 |
| Fiber optic | 32.9 |
| No | 30.5 |

# Pipe Operator (%>%)

# Pipe Operator %>%

Pipe Operator helps to chain a sequence

# Pipe Operator Examples

Using the pipe operator extract only the first 5 columns and only ‘Male’ customers from the customer_churn dataset

```
customer_churn %>% select(1:5) %>% filter( gender =="Male")
```

| customerID | gender | SeniorCitizen | Partner | Dependents |
|---|---|---|---|---|
| 5575-GNVDE | Male | 0 | No | No |
| 3668-QPYBK | Male | 0 | No | No |
| 7795-CFOCW | Male | 0 | No | No |
| 1452-KIOVK | Male | 0 | No | Yes |
| 6388-TABGU | Male | 0 | No | Yes |
| 9763-GRSKD | Male | 0 | Yes | Yes |
| 7469-LKBCI | Male | 0 | No | No |
| 8091-TTVAX | Male | 0 | Yes | No |
| 0280-XJGEX | Male | 0 | No | No |

# Pipe Operator Examples

Using the pipe operator get summarized result of mean “MonthlyCharges” of only those records where “InternetService” is *DSL* grouped w.r.t “gender”

```
customer_churn %>%
filter(InternetService=="DSL") %>%
group_by(gender) %>%
summarise(mean_mc=mean(MonthlyCharges))
```

```
# A tibble: 2 x 2
  gender mean_mc
  <fct>     <dbl>
1 Female     58.6
2 Male       57.6
```

# Pipe Operator Examples

Using the pipe operator select only 1st,2nd, 18th & 19th columns. After that filter out those records where 'MonthlyCharges' are greater than 100 AND gender is 'Male'

```
customer_churn %>%
select(1,2,18,19) %>%
filter(MonthlyCharges>100 & gender =="Male")
```

| customerID | gender | MonthlyCharges | PaymentMethod |
|---|---|---|---|
| 8091-TTVAX | Male | 100.35 | Credit card (automatic) |
| 0280-XJGEX | Male | 103.70 | Bank transfer (automatic) |
| 5129-JLPIS | Male | 105.50 | Electronic check |
| 9959-WOFKT | Male | 106.70 | Bank transfer (automatic) |
| 5380-WJKOV | Male | 106.35 | Electronic check |
| 5067-XJQFU | Male | 108.45 | Electronic check |
| 1891-QRQSA | Male | 111.60 | Bank transfer (automatic) |
| 9848-JQJTX | Male | 100.90 | Bank transfer (automatic) |
| 3192-NQECA | Male | 110.00 | Bank transfer (automatic) |

# Quiz

# Quiz

Which of the following function is used to add a new column in the data?

a. summary()

b. head()

c. tail()

d. mutate()

# Quiz

Which of the following statements describe dplyr in R ?

a. The dplyr is a powerful R-package to manipulate, clean and summarize unstructured data

b. The dply makes data exploration and data manipulation easy and fast in R

c. All of the above

d. None of the above

# Quiz

Which of the following statements describes pipe operator?

a. It lets to wrap multiple functions together with the use of %>%

b. It helps in installing the package in R

c. It helps in reading data in R

d. None of the above

# Quiz

Which of the following statements defines lapply() function ?

a. It loops over a list, iterating over each element in that list

b. It applies a function to each element of the list
(a function that you specify)

c. and returns a list (the l is for “list”).

d. All of the above

# Quiz

Which of the following statement describes mapply() function?

a. multivariate version of sapply

b. multivariate version of lapply

c. multivariate version of tapply

d. None of the above


# Thank You