# Load Package

First, we need to load basis three packages into R.

```
#install.packages("tidyverse")
#install.packages("lubridate")
#install.packages("nycflights13")
library(tidyverse)
library(lubridate)
library(nycflights13)
```

Proportion test in R

# Getting Data

Based on **nycflights13** data just load the data in o R environment.

head(flights)

## tidyverse in r

### 1. Create a new column basis count option

```
flights %>%
  mutate(long_flight = (air_time >= 6 * 60)) %>%
  View()
```

You can create new column long flights based on above scripts.

Now need to count the number of long flights

```
flights %>%
  mutate(long_flight = (air_time >= 6 * 60)) %>%
  count(long_flight)
```

The above two steps you can execute in a single line.

```
flights %>%
  count(long_flight = air_time >= 6 * 60)
```

Same way all different column count can calculate, one example is here.

```
flights %>%
  count(flight_path = str_c(origin, " -> ", dest), sort = TRUE)
```

### 2. Create a new column basis group by

You can create group by summary based on below script.

```
flights %>%
  group_by(date = make_date(year, month, day)) %>%
  summarise(flights_n = n(), air_time_mean = mean(air_time, na.rm =
TRUE)) %>%
  ungroup()
```

**3. Randomly Shuffle the data**

Suppose you want to randomly slice the data with 15 rows, can execute the same basis below command.

```
flights %>%
  slice_sample(n = 15)
```

Using prop command also you can slice the data set.

```
flights %>%
  slice_sample(prop = 0.15)
```

**4. Date column creation**

In the original data set year, month and date contained as separate columns-based make_date command can create new date column.

```
flights %>%
  select(year, month, day) %>%
  mutate(date = make_date(year, month, day))
```

**5. Number Parsing**

Suppose you want extract only numbers then you can you parse_number option.

```
numbers_1 <- tibble(number = c("#1", "Number8", "How are you 3"))
numbers_1 %>% mutate(number = parse_number(number))
```

**6. Select columns with starts_with and ends_with**

You can select the columns based on start_with and end_with option, here is the example

```
flights %>%
  select(starts_with("dep_"))
flights %>%
  select(ends_with("hour"))
flights %>%
  select(contains("hour"))
```

This is one of the useful code for our day to day life.

**7. case_when to create when conditions are met**

Create a new columns when conditions are met. case_when is one of the handy tool for conditions identification.

one sample analysis in R

```
flights %>%
  mutate(origin = case_when(
    (origin == "EWR") & dep_delay > 20 ~ "Newark International Airport
- DELAYED",
    (origin == "EWR") & dep_delay <= 20 ~ "Newark International Airport
- ON TIME DEPARTURE",
  )) %>%
  count(origin)
```

**8. str_replace_all to find and replace multiple options at once**

Every one aware about str_replace in string r pacakage, here we can execute replace multiple options at a once.

```
flights %>%
  mutate(origin = str_replace_all(origin, c(
    "^EWR$" = "Newark International",    "^JFK$" = "John F. Kennedy
International"
  ))) %>%
  count(origin)
```

**9. Filter groups without making a new column**

Filtering is one of the essential function for cleaning and checking data sets.

```
flights_top_carriers <- flights %>%
  group_by(carrier) %>%
  filter(n() >= 10000) %>%
  ungroup()
```

**10. Extract rows from the first table which are matched in the second table**

You can extract the row information's based on str_detect function

```
beginning_with_am<- airlines %>%
filter(name %>% str_detect("^Am"))
```

Non Parametric tests

**11. Extract rows from the first table which are not matched in the second table**

Same way you can remove row information's from the data frame while using anti_join function

```
flights %>%
  anti_join(airways_beginning_with_a, by = "carrier")
```

**12. fct_reorder to sort for charts creation**

When you are creating graphs reordering one of the key function, tidyverse will handle such kind of situations.

```
airline_names <- flights %>%
  left_join(airlines, by = "carrier")
airline_names %>%
```

```
  count(name) %>%
  ggplot(aes(name, n)) +
  geom_col()
airline_names %>%
  count(name) %>%
  mutate(name = fct_reorder(name, n)) %>%
  ggplot(aes(name, n)) +
  geom_col()
```

### 13. coord_flip to display counts more accurately

To change x and y axis and make a beautiful display

```
flights_with_airline_names %>%
count(name) %>%
mutate(name = fct_reorder(name, n)) %>%
ggplot(aes(name, n)) +
geom_col() +
coord_flip()
```

Types of Data Visualization

### 14. Generate all combinations using crossing

Like expand grid in R, you can create all possible combinations based on crossing function in tidyverse.

```
crossing(
  customer_channel = c("Bus", "Car"),
  customer_status = c("New", "Repeat"),
  spend_range = c("$0-$10", "$10-$20", "$20-$50", "$50+"))
```

### 15. Group by based on function

Write the function based on your requirements and group by accordingly.

```
summary <- function(data, col_names, na.rm = TRUE) {
  data %>%
    summarise(across({{ col_names }},
                     list(
                       min = min,
                       max = max,
                       median = median,
                       mean = mean
                     ),
                     na.rm = na.rm,
                     .names = "{col}_{fn}"
    ))
}
flights_with_airline_names %>%
  summary(c(air_time, arr_delay))
flights_with_airline_names %>%
  group_by(carrier) %>%
```

```
summary(c(air_time, arr_delay))
```

Uses of Index Numbers