Class Work

Sayan Das

2022-10-31

tibble, dplyr, tidyverse tutorial (continued. . .)

Calling the 'flights' dataset from the nycflights13 library

flights dataset is available in the *nycflights13* library. To use it, it should be installed into the system.

```
library(nycflights13)
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
flights
```

```
## # A tibble: 336,776 x 19
##     year month   day dep_time sched_de~1 dep_d~2 arr_t~3 sched~4 arr_d~5 carrier
##    <int> <int> <int>    <int>      <int>   <dbl>   <int>   <int>   <dbl> <chr>
##  1  2013     1     1      517        515       2     830     819      11 UA
##  2  2013     1     1      533        529       4     850     830      20 UA
##  3  2013     1     1      542        540       2     923     850      33 AA
##  4  2013     1     1      544        545      -1    1004    1022     -18 B6
##  5  2013     1     1      554        600      -6     812     837     -25 DL
##  6  2013     1     1      554        558      -4     740     728      12 UA
##  7  2013     1     1      555        600      -5     913     854      19 B6
##  8  2013     1     1      557        600      -3     709     723     -14 EV
##  9  2013     1     1      557        600      -3     838     846      -8 B6
## 10  2013     1     1      558        600      -2     753     745       8 AA
## # ... with 336,766 more rows, 9 more variables: flight <int>, tailnum <chr>,
## #   origin <chr>, dest <chr>, air_time <dbl>, distance <dbl>, hour <dbl>,
## #   minute <dbl>, time_hour <dttm>, and abbreviated variable names
## #   1: sched_dep_time, 2: dep_delay, 3: arr_time, 4: sched_arr_time,
## #   5: arr_delay
```

```r
small_flight = select(flights, arr_delay, dep_delay, distance, air_time)
small_flight = mutate(small_flight, gain = arr_delay-dep_delay, speed = distance/air_time*60)
```

Query: On an average what is The departure delay? And also report the average speed

Using summarize() function we can do operation on the rows. Like we can apply average on a particular columns.

syntax
summaraise(dataset name, summarization factor name = definition)

```r
summarise(small_flight, avg_delay = mean(dep_delay, na.rm=TRUE), avg_speed = mean(speed, na.rm=TRUE))
```

```
## # A tibble: 1 x 2
##   avg_delay avg_speed
##       <dbl>     <dbl>
## 1      12.6      394.
```

Similarity between mutate() and summarize()
Both of them returns a dataframe in a tibble format
Both of are doing some arithmatic operation

Dis-similarirty between mutate() and summarize()
mutate() is taking vectorize function only…That is it takes a vector input and it also returns a vector output.

But in summarize() we are condensing the data, here the result need not to be vectorize.

Query: Report the average departure delay and total departure delay for eachday

group_by() function is going to help us in this case. It helps to apply aggregate funtion for each of the values available in some column

syntax
group_by(dataset name, column1, column2, column3,…) column1, column2, column3 are those column according to whose values the data should be grouped.

```r
day_wise = group_by(flights, year, month, day)
summarise(day_wise, avg_dep_delay = mean(dep_delay, na.rm=TRUE), tot_dep_delay = sum(dep_delay, na.rm=T
```

```
## `summarise()` has grouped output by 'year', 'month'. You can override using the
## `.groups` argument.
```

```
## # A tibble: 365 x 5
## # Groups:   year, month [12]
##     year month   day avg_dep_delay tot_dep_delay
##    <int> <int> <int>         <dbl>         <dbl>
##  1  2013     1     1         11.5           9678
##  2  2013     1     2         13.9          12958
##  3  2013     1     3         11.0           9933
##  4  2013     1     4          8.95          8137
##  5  2013     1     5          5.73          4110
##  6  2013     1     6          7.15          5940
##  7  2013     1     7          5.42          5038
##  8  2013     1     8          2.55          2285
##  9  2013     1     9          2.28          2042
## 10  2013     1    10          2.84          2643
## # ... with 355 more rows
```

Query: Compute the average departure delay grouped by day , then over month and then by year

```
(per_day = summarise(day_wise, avg_delay = mean(dep_delay, na.rm=TRUE)))
```

```
## `summarise()` has grouped output by 'year', 'month'. You can override using the
## `.groups` argument.

## # A tibble: 365 x 4
## # Groups:   year, month [12]
##     year month   day avg_delay
##    <int> <int> <int>     <dbl>
##  1  2013     1     1     11.5
##  2  2013     1     2     13.9
##  3  2013     1     3     11.0
##  4  2013     1     4      8.95
##  5  2013     1     5      5.73
##  6  2013     1     6      7.15
##  7  2013     1     7      5.42
##  8  2013     1     8      2.55
##  9  2013     1     9      2.28
## 10  2013     1    10      2.84
## # ... with 355 more rows
```

```
(per_month = summarise(per_day, avg_delay = mean(avg_delay, na.rm=TRUE)))
```

```
## `summarise()` has grouped output by 'year'. You can override using the
## `.groups` argument.

## # A tibble: 12 x 3
## # Groups:   year [1]
##     year month avg_delay
##    <int> <int>     <dbl>
##  1  2013     1     10.0
##  2  2013     2     11.1
##  3  2013     3     13.6
##  4  2013     4     13.9
##  5  2013     5     13.2
##  6  2013     6     20.9
##  7  2013     7     21.8
##  8  2013     8     12.6
##  9  2013     9      6.92
## 10  2013    10      6.19
## 11  2013    11      5.30
## 12  2013    12     16.9
```

```
(per_year = summarise(per_month, avg_delay = mean(avg_delay, na.rm=TRUE)))
```

```
## # A tibble: 1 x 2
##    year avg_delay
##   <int>     <dbl>
## 1  2013      12.7
```

Query:

Suppose I want to study the relationship between the distance and average arrival delay for each destination. For this I have to do the following steps 1. Group all flights by destinations 2. Summarise the data to compute the average distance, average delay and no. of flights arriving in each destination 3. Filter to remove noisy points and Honolulu air which is twice as far away as the next closest airport 4. Prepare a scatter plot of average distance and average arrival delay

Here we are going to need n() function that will be helping us to count the number of flights.

syntax
n(): (no argument needed) see example

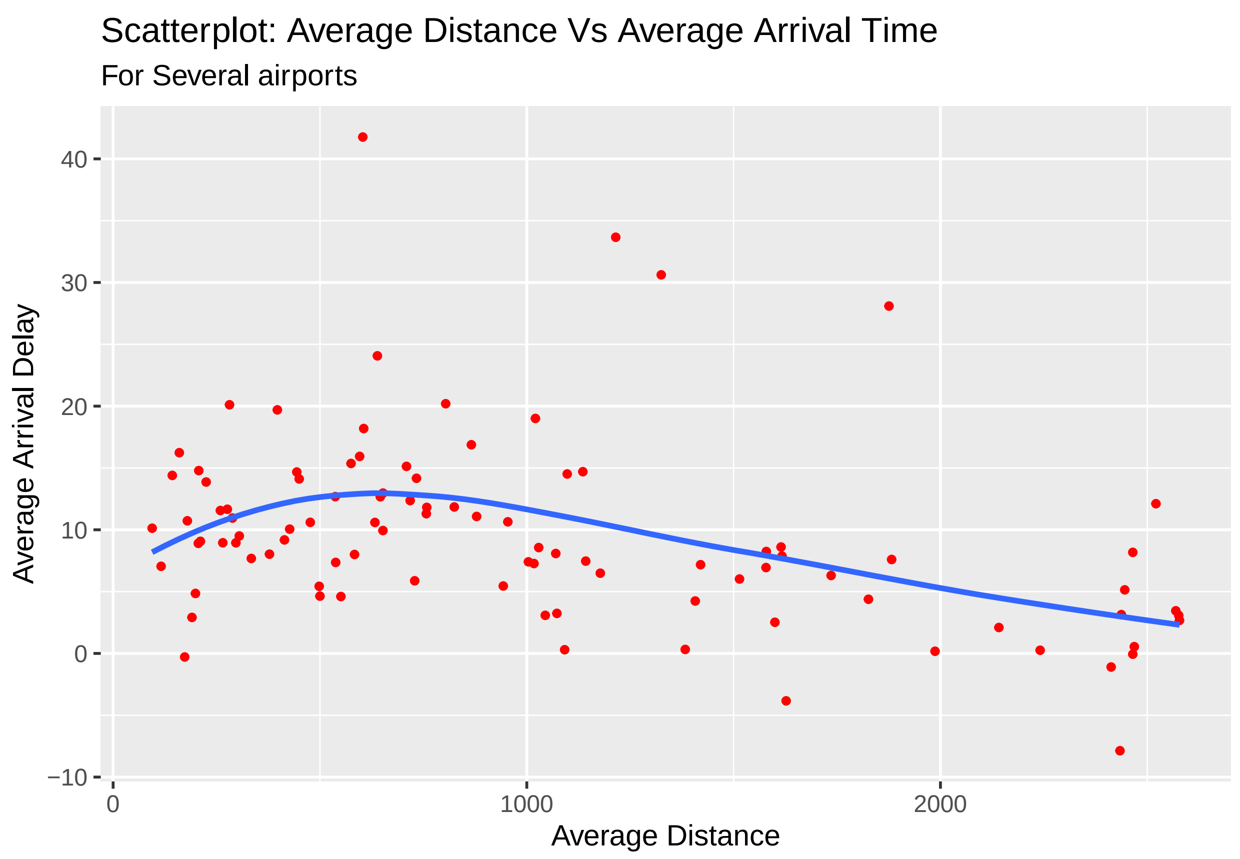
Here noisy point means those destinations that has less or equal to 20 flights arrived.

```
dest_wise = group_by(flights, dest)
destwise_summary = summarise(dest_wise, avg_dist = mean(distance, na.rm=TRUE), avg_arr_delay = mean(arr
filtered = filter(destwise_summary, dest != 'HNL', flight_count > 20)

library(ggplot2)

ggplot(filtered, aes(x = avg_dist, y = avg_arr_delay))+
  geom_point(colour='Red', size =1.02)+
  geom_smooth(se=FALSE)+
  labs(x = 'Average Distance', y = 'Average Arrival Delay',
       title = 'Scatterplot: Average Distance Vs Average Arrival Time',
       subtitle = 'For Several airports')
```

```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```



As the distance increases the arrival delay decreases, that means it is easier for the flights to make up the delay when it is getting a long airtime.

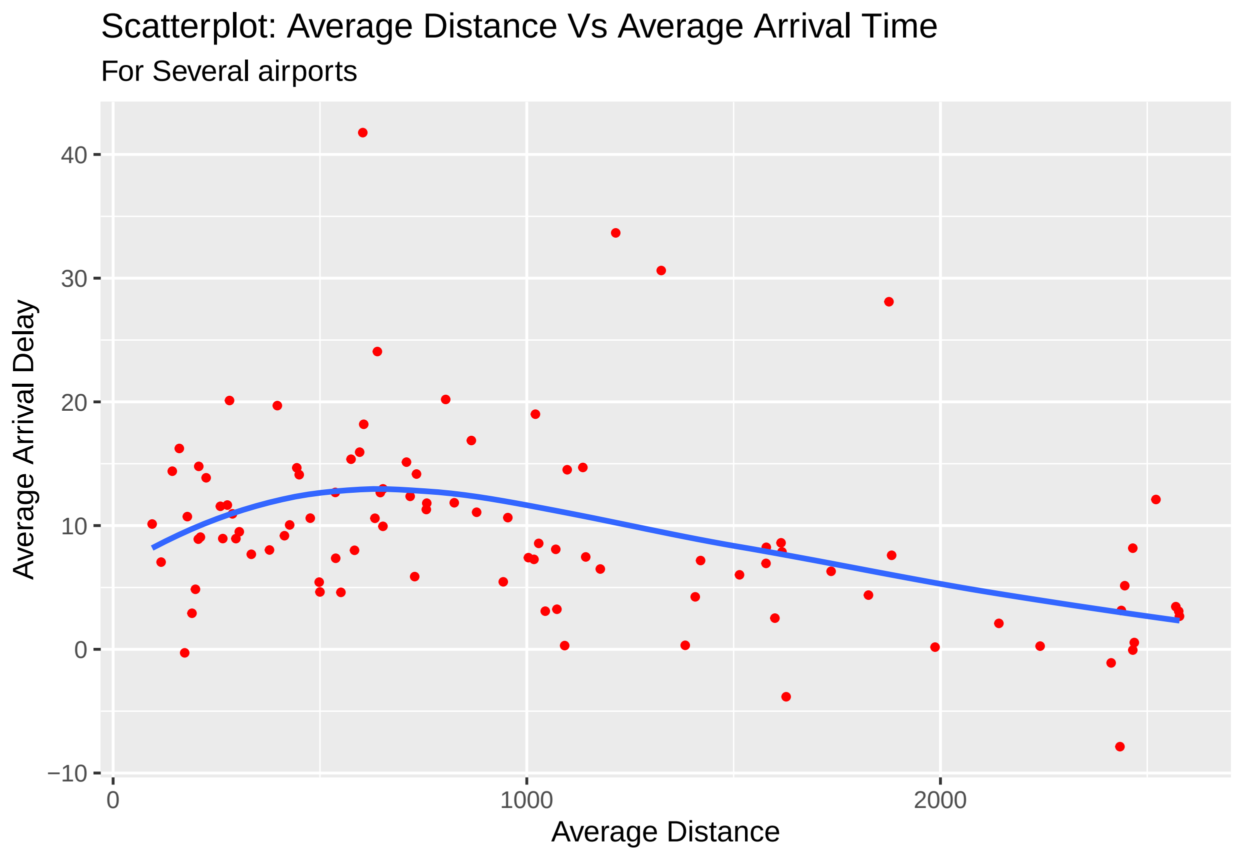
Pipe operator %>%

pipe operator (%>%) is used when a number of functions are applied sequentially. It helps us to avoid renaming for multiple time. The result of the previous query using pipe operator is given below.

```
### previous query using pipe operator
new_data = flights %>%
      group_by(dest) %>%
      summarise(avg_dist = mean(distance, na.rm=TRUE),
                avg_arr_delay = mean(arr_delay, na.rm=TRUE),
                flight_count=n()) %>%
      filter(dest != 'HNL', flight_count > 20)

library(ggplot2)
ggplot(new_data, aes(x = avg_dist, y = avg_arr_delay))+
  geom_point(colour='Red', size =1.02)+
  geom_smooth(se=FALSE)+
  labs(x = 'Average Distance', y = 'Average Arrival Delay',
       title = 'Scatterplot: Average Distance Vs Average Arrival Time',
       subtitle = 'For Several airports')
```

```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```



### note: Tibble functions used so far filter(): filter a data according to some condition
names(): find the names of the variable in the dataset
between(): condition on variable having values between two other values
select(): used to select particular columns from a dataset
mutate(): used to append a column in the dataset
transmute(): used to show a tranformation on column in the dataset
summarise(): used to get a aggregate measure on any column
group_by(): used to group the dataset accorning to the values of some column. n(): used to count the number of observations