R package dplyr, Data Manipulation

Jason Wang

Tuesday, April 28, 2015

This is an introduction to th R package `dplyr`. If you type the "dplyr" in google or youtube, you will get tremendous references, tutorials and examples. It is written in C, which mean that it will translate your R code into C and the manipulating functions are really fast. Its function and structure are like those we use in database language. Mainly, it contains six verb functions, `filter`, `select`, `arrange`, `mutate`, `summarise`, `group_by` and also some more advanced function.

```r
library(dplyr)
#Package which I use its data to demonstrate.
library(hflights)
```

```r
data(hflights)
dim(hflights)
```

```
## [1] 227496     21
```

```r
names(hflights)
```

```
##  [1] "Year"              "Month"             "DayofMonth"       
##  [4] "DayOfWeek"         "DepTime"           "ArrTime"          
##  [7] "UniqueCarrier"     "FlightNum"         "TailNum"          
## [10] "ActualElapsedTime" "AirTime"           "ArrDelay"         
## [13] "DepDelay"          "Origin"            "Dest"             
## [16] "Distance"          "TaxiIn"            "TaxiOut"          
## [19] "Cancelled"         "CancellationCode"  "Diverted"         
```

```r
head(hflights)
```

If you run the head code, the output is not really good for people to understand. I am not going to show the output here.

1. Data Type: tbl

First of all, we have to transfrom the our data into the table, local data frame. (Note that originally it is a data frame.)

```
#Transform data.frame to table
hflights <- tbl_df(hflights)
```

Normally, if you print out the data which have 227496 rows and 21columns, you will get a messy output. However, when the data is table, then we can print it without any worry. It will automatically adapt the data output to us.

```
#Nicer output compared with the originally one
hflights
```

```
## Source: local data frame [227,496 x 21]
##
##    Year Month DayofMonth DayOfWeek DepTime ArrTime UniqueCarrier FlightNum
## 1  2011     1          1         6    1400    1500            AA       428
## 2  2011     1          2         7    1401    1501            AA       428
## 3  2011     1          3         1    1352    1502            AA       428
## 4  2011     1          4         2    1403    1513            AA       428
## 5  2011     1          5         3    1405    1507            AA       428
## 6  2011     1          6         4    1359    1503            AA       428
## 7  2011     1          7         5    1359    1509            AA       428
## 8  2011     1          8         6    1355    1454            AA       428
## 9  2011     1          9         7    1443    1554            AA       428
## 10 2011     1         10         1    1443    1553            AA       428
## .. ...   ...        ...       ...     ...     ...           ...       ...
## Variables not shown: TailNum (chr), ActualElapsedTime (int), AirTime
##   (int), ArrDelay (int), DepDelay (int), Origin (chr), Dest (chr),
##   Distance (int), TaxiIn (int), TaxiOut (int), Cancelled (int),
##   CancellationCode (chr), Diverted (int)
```

In the beginning, it shows you the dimension of the data. In the middle, it gives you an appropriate output which fit the windows. In the end, it tells you what variables do not show.

2. Verb 1 filter (Row)

Usually, when we first get the data, we will explore it in a different way. We may thinks that maybe certain group in the data will have the same patterns. We want to find out those subjects (row) which have the same features in certain variable. Then, we can utilize the `filter` function to help us.

For instance, you may want to get those airflight in January. In `dplyr`, it is really easy.

```
#In dplyr
filter(hflights, Month == 1)
```

```
## Source: local data frame [18,910 x 21]
##
##    Year Month DayofMonth DayOfWeek DepTime ArrTime UniqueCarrier FlightNum
## 1  2011     1          1         6    1400    1500            AA       428
## 2  2011     1          2         7    1401    1501            AA       428
## 3  2011     1          3         1    1352    1502            AA       428
## 4  2011     1          4         2    1403    1513            AA       428
## 5  2011     1          5         3    1405    1507            AA       428
## 6  2011     1          6         4    1359    1503            AA       428
## 7  2011     1          7         5    1359    1509            AA       428
## 8  2011     1          8         6    1355    1454            AA       428
## 9  2011     1          9         7    1443    1554            AA       428
## 10 2011     1         10         1    1443    1553            AA       428
## ..  ...   ...        ...       ...     ...     ...           ...       ...
## Variables not shown: TailNum (chr), ActualElapsedTime (int), AirTime
##   (int), ArrDelay (int), DepDelay (int), Origin (chr), Dest (chr),
##   Distance (int), TaxiIn (int), TaxiOut (int), Cancelled (int),
##   CancellationCode (chr), Diverted (int)
```

As you can see, it will show you a nicely format. You can know how many flights are in Janauary. In normal R codes, you may have to write some codes like `hflights[hflights$Month == 1, ]`.

You can save it into another table such as Jan. It will keep the same data type. Also, you can transform it back to data.frame without any warning message.

```
Jan <- filter(hflights, Month == 1)
class(Jan)
```

```
## [1] "tbl_df"     "tbl"        "data.frame"
```

```
#You can also get the data as the way for data frame
Jan[1, 1]
```

```
## Source: local data frame [1 x 1]
##
##   Year
## 1 2011
```

```
Jan <- data.frame(Jan)
class(Jan)
```

```
## [1] "data.frame"
```

If you have multiple condition, you can use AND(&) as well as OR(|) in the function.

```
filter(hflights, Month == 1 & DayofMonth == 1)
```

```
## Source: local data frame [552 x 21]
##
##    Year Month DayofMonth DayOfWeek DepTime ArrTime UniqueCarrier FlightNum
## 1  2011     1          1         6    1400    1500            AA       428
## 2  2011     1          1         6     728     840            AA       460
## 3  2011     1          1         6    1631    1736            AA      1121
## 4  2011     1          1         6    1756    2112            AA      1294
## 5  2011     1          1         6    1012    1347            AA      1700
## 6  2011     1          1         6    1211    1325            AA      1820
## 7  2011     1          1         6     557     906            AA      1994
## 8  2011     1          1         6    1824    2106            AS       731
## 9  2011     1          1         6     654    1124            B6       620
## 10 2011     1          1         6    1639    2110            B6       622
## ..  ...   ...        ...       ...     ...     ...           ...       ...
## Variables not shown: TailNum (chr), ActualElapsedTime (int), AirTime
##   (int), ArrDelay (int), DepDelay (int), Origin (chr), Dest (chr),
##   Distance (int), TaxiIn (int), TaxiOut (int), Cancelled (int),
##   CancellationCode (chr), Diverted (int)
```

```
#You can also replace AND by ,
filter(hflights, Month == 1, DayofMonth == 1)
```

```
## Source: local data frame [552 x 21]
##
##    Year Month DayofMonth DayOfWeek DepTime ArrTime UniqueCarrier FlightNum
## 1  2011     1          1         6    1400    1500            AA       428
## 2  2011     1          1         6     728     840            AA       460
## 3  2011     1          1         6    1631    1736            AA      1121
## 4  2011     1          1         6    1756    2112            AA      1294
## 5  2011     1          1         6    1012    1347            AA      1700
## 6  2011     1          1         6    1211    1325            AA      1820
## 7  2011     1          1         6     557     906            AA      1994
## 8  2011     1          1         6    1824    2106            AS       731
## 9  2011     1          1         6     654    1124            B6       620
## 10 2011     1          1         6    1639    2110            B6       622
## ..  ...   ...        ...       ...     ...     ...           ...       ...
## Variables not shown: TailNum (chr), ActualElapsedTime (int), AirTime
##   (int), ArrDelay (int), DepDelay (int), Origin (chr), Dest (chr),
##   Distance (int), TaxiIn (int), TaxiOut (int), Cancelled (int),
##   CancellationCode (chr), Diverted (int)
```

You can also filter by character.

```
filter(hflights, UniqueCarrier == "AA")
```

```
## Source: local data frame [3,244 x 21]
## 
##    Year Month DayofMonth DayOfWeek DepTime ArrTime UniqueCarrier FlightNum
## 1  2011     1          1         6    1400    1500            AA       428
## 2  2011     1          2         7    1401    1501            AA       428
## 3  2011     1          3         1    1352    1502            AA       428
## 4  2011     1          4         2    1403    1513            AA       428
## 5  2011     1          5         3    1405    1507            AA       428
## 6  2011     1          6         4    1359    1503            AA       428
## 7  2011     1          7         5    1359    1509            AA       428
## 8  2011     1          8         6    1355    1454            AA       428
## 9  2011     1          9         7    1443    1554            AA       428
## 10 2011     1         10         1    1443    1553            AA       428
## ..  ...   ...        ...       ...     ...     ...           ...       ...
## Variables not shown: TailNum (chr), ActualElapsedTime (int), AirTime
##   (int), ArrDelay (int), DepDelay (int), Origin (chr), Dest (chr),
##   Distance (int), TaxiIn (int), TaxiOut (int), Cancelled (int),
##   CancellationCode (chr), Diverted (int)
```

3. Verb 2 select (Column)

You might also want to examine some column in the data. In R, you may use the code like `hflights[, c("Month", "DayofMonth", "FlightNum")]`.

```
#In dplyr
select(hflights, Month, DayofMonth, FlightNum)
```

```
## Source: local data frame [227,496 x 3]
##
##    Month DayofMonth FlightNum
## 1      1          1       428
## 2      1          2       428
## 3      1          3       428
## 4      1          4       428
## 5      1          5       428
## 6      1          6       428
## 7      1          7       428
## 8      1          8       428
## 9      1          9       428
## 10     1         10       428
## ..   ...        ...       ...
```

```
#You can also use contain function to select the column which have the same keyword
select(hflights, contains("delay"))
```

```
## Source: local data frame [227,496 x 2]
##
##    ArrDelay DepDelay
## 1       -10        0
## 2        -9        1
## 3        -8       -8
## 4         3        3
## 5        -3        5
## 6        -7       -1
## 7        -1       -1
## 8       -16       -5
## 9        44       43
## 10       43       43
## ..      ...      ...
```

```r
#Or use starts_with
select(hflights, starts_with("M"))
```

```
## Source: local data frame [227,496 x 1]
##
##    Month
## 1      1
## 2      1
## 3      1
## 4      1
## 5      1
## 6      1
## 7      1
## 8      1
## 9      1
## 10     1
## ..   ...
```

```r
#Or use ends_with
select(hflights, ends_with("th"))
```

```
## Source: local data frame [227,496 x 2]
##
##    Month DayofMonth
## 1      1          1
## 2      1          2
## 3      1          3
## 4      1          4
## 5      1          5
## 6      1          6
## 7      1          7
## 8      1          8
## 9      1          9
## 10     1         10
## ..   ...        ...
```

Digression about a nicer way to write codes in dplyr

Sometimes we will combine the `filter` and `select` function together. For example, you may want to find out the FlightNum and Origin of those filghts which their UniqueCarrier are WN. You will write a code like the following:

```
filter(select(hflights, FlightNum, Origin, UniqueCarrier), UniqueCarrier == "WN")
```

```
## Source: local data frame [45,343 x 3]
##
##    FlightNum Origin UniqueCarrier
## 1       1266    HOU            WN
## 2       1689    HOU            WN
## 3       1024    HOU            WN
## 4       2430    HOU            WN
## 5       3013    HOU            WN
## 6       1038    HOU            WN
## 7       2345    HOU            WN
## 8       1454    HOU            WN
## 9       2360    HOU            WN
## 10      1593    HOU            WN
## ..       ...    ...           ...
```

When you see the above codes, it may take some times to really get what the codes is doing. However, in `dplyr`, it gives user a more generally way to organize their code called "**chaining**" which can give a more readable code.

```
#Another much more easier way to write code
hflights %>%
    select(FlightNum, Origin, UniqueCarrier) %>%
    filter(UniqueCarrier == "WN")
```

```
## Source: local data frame [45,343 x 3]
##
##    FlightNum Origin UniqueCarrier
## 1       1266    HOU            WN
## 2       1689    HOU            WN
## 3       1024    HOU            WN
## 4       2430    HOU            WN
## 5       3013    HOU            WN
## 6       1038    HOU            WN
## 7       2345    HOU            WN
## 8       1454    HOU            WN
## 9       2360    HOU            WN
## 10      1593    HOU            WN
## ..       ...    ...           ...
```

You can regard the %>% syntax as “then”. It will be like that get the hflights data, then select FlightNum, Origin, UniqueCarrier, then filter the outcome to find those UniqueCarrier is “WN”. It is much more interpretable way to present the code to others. Actually, the syntax exists in the R code and we can utilize it in other code too.

```
x <- rnorm(2)
y <- rnorm(2)
#Distance
sqrt(sum((x - y)^2))
```

```
## [1] 3.098904
```

```
#In %>% way
(x - y)^2 %>% sum %>% sqrt
```

```
## [1] 3.098904
```

4. Verb 3 arrange

Usually, we want to sort data and see whether overall data shows pattern after sorting. We can use `hflights[order(hflights$DepTime), c("Month", "DepTime", "ArrTime")]`

```
#In dplyr
arrange(select(hflights, Month, DepTime, ArrTime), DepTime)
```

```
## Source: local data frame [227,496 x 3]
##
##    Month DepTime ArrTime
## 1      1       1     621
## 2      3       1     557
## 3      4       1     510
## 4      6       1     515
## 5     11       1      55
## 6     12       1     642
## 7     12       1     633
## 8      7       2      53
## 9     12       2     611
## 10     7       3     521
## ..   ...     ...     ...
```

```r
#Chaining
hflights %>%
  select(Month, DepTime, ArrTime) %>%
  arrange(DepTime)
```

```
## Source: local data frame [227,496 x 3]
##
##    Month DepTime ArrTime
## 1      1       1     621
## 2      3       1     557
## 3      4       1     510
## 4      6       1     515
## 5     11       1      55
## 6     12       1     642
## 7     12       1     633
## 8      7       2      53
## 9     12       2     611
## 10     7       3     521
## ..   ...     ...     ...
```

```r
#In decreasing way
hflights %>%
  select(Month, DepTime, ArrTime) %>%
  arrange(desc(DepTime))
```

```
## Source: local data frame [227,496 x 3]
##
##    Month DepTime ArrTime
## 1      5    2400     144
## 2      4    2359     455
## 3      5    2359     130
## 4      5    2359      56
## 5      6    2359     113
## 6      6    2359      40
## 7      6    2359     111
## 8      7    2359     108
## 9      7    2359     105
## 10     9    2359     106
## ..   ...     ...     ...
```

5. Verb 4 mutate

Sometimes, you may want to add or create a new variable into data. For example, if we want to add speed, we use `hflights$speed <- hflights$Distance/hflights$AirTime`.

```
#In dplyr
hflights %>%
  select(Distance, AirTime) %>%
  mutate(Speed=Distance/AirTime)
```

```
## Source: local data frame [227,496 x 3]
##
##    Distance AirTime    Speed
## 1       224      40 5.600000
## 2       224      45 4.977778
## 3       224      48 4.666667
## 4       224      39 5.743590
## 5       224      44 5.090909
## 6       224      45 4.977778
## 7       224      43 5.209302
## 8       224      40 5.600000
## 9       224      41 5.463415
## 10      224      45 4.977778
## ..      ...     ...      ...
```

```
#To store
hflights <- mutate(hflights, Speed=Distance/AirTime)
select(hflights, Distance, AirTime, Speed)
```

```
## Source: local data frame [227,496 x 3]
##
##    Distance AirTime    Speed
## 1       224      40 5.600000
## 2       224      45 4.977778
## 3       224      48 4.666667
## 4       224      39 5.743590
## 5       224      44 5.090909
## 6       224      45 4.977778
## 7       224      43 5.209302
## 8       224      40 5.600000
## 9       224      41 5.463415
## 10      224      45 4.977778
## ..      ...     ...      ...
```

6. Verb 5 summarise + Verb 6 group_by

When it comes to descriptive statistics, we will want to statistics based on several group if we have multiple group. For example, if we want to know the average arrival delay time for different destination, then we run `head(tapply(hflights$ArrDelay, hflights$Dest, mean, na.rm=T))` or `head(aggregate(ArrDelay ~ Dest, hflights, mean))`.

```r
#In dplyr
summarise(group_by(hflights, Dest), mean(ArrDelay, na.rm=T))
```

```
## Source: local data frame [116 x 2]
##
##    Dest mean(ArrDelay, na.rm = T)
## 1   ABQ                  7.226259
## 2   AEX                  5.839437
## 3   AGS                  4.000000
## 4   AMA                  6.840095
## 5   ANC                 26.080645
## 6   ASE                  6.794643
## 7   ATL                  8.233251
## 8   AUS                  7.448718
## 9   AVL                  9.973988
## 10  BFL                -13.198807
## ..  ...                       ...
```

```r
#Chaining
hflights %>%
  group_by(Dest) %>%
  summarise(delay_Time=mean(ArrDelay, na.rm=T))
```

```
## Source: local data frame [116 x 2]
##
##    Dest delay_Time
## 1   ABQ   7.226259
## 2   AEX   5.839437
## 3   AGS   4.000000
## 4   AMA   6.840095
## 5   ANC  26.080645
## 6   ASE   6.794643
## 7   ATL   8.233251
## 8   AUS   7.448718
## 9   AVL   9.973988
## 10  BFL -13.198807
## ..  ...        ...
```

Sometimes, we may want to apply the function to several columns or apply different functions to column.

```
#Apply function to multiple columns
hflights %>%
  group_by(UniqueCarrier) %>%
  summarise_each(funs(mean(., na.rm=T)), Cancelled, Diverted)
```

```
## Source: local data frame [15 x 3]
##
##    UniqueCarrier   Cancelled    Diverted
## 1             AA 0.018495684 0.001849568
## 2             AS 0.000000000 0.002739726
## 3             B6 0.025899281 0.005755396
## 4             CO 0.006782614 0.002627370
## 5             DL 0.015903067 0.003029156
## 6             EV 0.034482759 0.003176044
## 7             F9 0.007159905 0.000000000
## 8             FL 0.009817672 0.003272557
## 9             MQ 0.029044750 0.001936317
## 10            OO 0.013946828 0.003486707
## 11            UA 0.016409266 0.002413127
## 12            US 0.011268986 0.001469868
## 13            WN 0.015504047 0.002293629
## 14            XE 0.015495599 0.003449550
## 15            YV 0.012658228 0.000000000
```

```
#Apply multiple functions to one column
hflights %>%
  group_by(Dest) %>%
  summarise_each(funs(mean(., na.rm=T), min(., na.rm=T), max(., na.rm=T)), ArrDelay)
```

```
## Source: local data frame [116 x 4]
##
##    Dest       mean min max
## 1   ABQ   7.226259 -26 290
## 2   AEX   5.839437 -34 257
## 3   AGS   4.000000   4   4
## 4   AMA   6.840095 -28 301
## 5   ANC  26.080645 -21 281
## 6   ASE   6.794643 -31 252
## 7   ATL   8.233251 -41 701
## 8   AUS   7.448718 -24 244
## 9   AVL   9.973988 -23 331
## 10  BFL -13.198807 -56 206
## ..  ...        ... ... ...
```

```
#Apply multiple functions to several columns
hflights %>%
  group_by(UniqueCarrier) %>%
  summarise_each(funs(min(., na.rm=T), max(., na.rm=T)), contains("Delay"))
```

```
## Source: local data frame [15 x 5]
##
##    UniqueCarrier ArrDelay_min DepDelay_min ArrDelay_max DepDelay_max
## 1             AA          -39          -15          978          970
## 2             AS          -43          -15          183          172
## 3             B6          -44          -14          335          310
## 4             CO          -55          -18          957          981
## 5             DL          -32          -17          701          730
## 6             EV          -40          -18          469          479
## 7             F9          -24          -15          277          275
## 8             FL          -30          -14          500          507
## 9             MQ          -38          -23          918          931
## 10            OO          -57          -33          380          360
## 11            UA          -47          -11          861          869
## 12            US          -42          -17          433          425
## 13            WN          -44          -10          499          548
## 14            XE          -70          -19          634          628
## 15            YV          -32          -11           72           54
```

7. n() and n_distinct

Here we introduce a useful functions that can facilitate our analysis. The first one is `n()`. It can help us count the number in the group(category) and perhaps we want to sort them by the its numbers. For example, we may want to know what's the rush month or day for airline. We can count the number by these two category and sort them to find out the results.

```
#In dplyr
hflights %>%
  group_by(Month, DayofMonth) %>%
  summarise(flight_count=n()) %>%
  ungroup() %>%
  arrange(desc(flight_count))
```

```
## Source: local data frame [365 x 3]
##
##    Month DayofMonth flight_count
## 1      8          4          706
## 2      8        11         706
## 3      8        12         706
## 4      8         5         705
## 5      8         3         704
## 6      8        10         704
## 7      1         3         702
## 8      7         7         702
## 9      7        14         702
## 10     7        28         701
## ..   ...        ...        ...
```

```r
#Another easier way to write it
hflights %>%
  group_by(Month, DayofMonth) %>%
  tally(sort=T)
```

```
## Source: local data frame [365 x 3]
## Groups: Month
##
##    Month DayofMonth   n
## 1      1          3 702
## 2      1          2 678
## 3      1         20 663
## 4      1         27 663
## 5      1         13 662
## 6      1          7 661
## 7      1         14 661
## 8      1         21 661
## 9      1         28 661
## 10     1          6 660
## ..   ...        ... ...
```

`n_distinct` funtion provide the number of unique subject in certain column you assign.

```r
hflights %>%
  group_by(Dest) %>%
  summarise(flight_count=n(), plane_count=n_distinct(TailNum))
```

```
## Source: local data frame [116 x 3]
##
##    Dest flight_count plane_count
## 1   ABQ         2812         716
## 2   AEX          724         215
## 3    AGS            1            1
## 4    AMA         1297          158
## 5    ANC          125           38
## 6    ASE          125           60
## 7    ATL         7886          983
## 8    AUS         5022         1015
## 9    AVL          350          142
## 10   BFL          504           70
## ..   ...          ...          ...
```

Sometimes, `group_by` can be very useful even without `summarise`.

```
hflights %>%
  group_by(Dest) %>%
  select(Cancelled) %>%
  table() %>%
  head()
```

```
##      Cancelled
## Dest     0  1
##   ABQ 2787 25
##   AEX  712 12
##   AGS    1  0
##   AMA 1265 32
##   ANC  125  0
##   ASE  120  5
```

8. Windows function

- Aggregate function: Input n values, output 1 value

*Windows function: Input n values, output n values

```
#For each destination, calculate which two days of the year they had their longest dep
#We use desc(DepDelay) because I want to find the largest value of DepDelay.
hflights %>%
  group_by(Dest) %>%
  select(Month, DayofMonth, DepDelay) %>%
  filter(min_rank(desc(DepDelay)) <= 2) %>%
  arrange(Dest, desc(DepDelay)) %>%
  print(n=15)
```

```
## Source: local data frame [233 x 4]
## Groups: Dest
##
##    Dest Month DayofMonth DepDelay
## 1   ABQ     3         31      300
## 2   ABQ    10         24      275
## 3   AEX    12         31      266
## 4   AEX     2          5      173
## 5   AGS     4          3       10
## 6   AMA    10          9      304
## 7   AMA    10          9      246
## 8   ANC     8         11      292
## 9   ANC     6         25      215
## 10  ASE    12         29      269
## 11  ASE     2          7      208
## 12  ATL    10         25      730
## 13  ATL     2         19      507
## 14  AUS     6         22      240
## 15  AUS    10          9      239
## ..  ...   ...        ...      ...
```

- top_n

```
#top_n
hflights %>%
    group_by(Dest) %>%
    select(Month, DayofMonth, DepDelay) %>%
    top_n(2) %>%
    arrange(Dest, desc(DepDelay))
```

```
## Selecting by DepDelay
```

```
## Source: local data frame [233 x 4]
## Groups: Dest
##
##    Dest Month DayofMonth DepDelay
## 1   ABQ     3         31      300
## 2   ABQ    10         24      275
## 3   AEX    12         31      266
## 4   AEX     2          5      173
## 5   AGS     4          3       10
## 6   AMA    10          9      304
## 7   AMA    10          9      246
## 8   ANC     8         11      292
## 9   ANC     6         25      215
## 10  ASE    12         29      269
## ..  ...   ...        ...      ...
```

```
#Compute the change from month to month
hflights %>%
  group_by(Month) %>%
  summarise(flight_count=n()) %>%
  mutate(change=flight_count-lag(flight_count))
```

```
## Source: local data frame [12 x 3]
##
##    Month flight_count change
## 1      1        18910     NA
## 2      2        17128  -1782
## 3      3        19470   2342
## 4      4        18593   -877
## 5      5        19172    579
## 6      6        19600    428
## 7      7        20548    948
## 8      8        20176   -372
## 9      9        18065  -2111
## 10    10        18696    631
## 11    11        18021   -675
## 12    12        19117   1096
```

```
#By tally function
hflights %>%
    group_by(Month) %>%
    tally() %>%
    mutate(change=n - lag(n))
```

```
## Source: local data frame [12 x 3]
##
##    Month     n change
## 1      1 18910     NA
## 2      2 17128  -1782
## 3      3 19470   2342
## 4      4 18593   -877
## 5      5 19172    579
## 6      6 19600    428
## 7      7 20548    948
## 8      8 20176   -372
## 9       9 18065  -2111
## 10     10 18696    631
## 11     11 18021   -675
## 12     12 19117   1096
```

9. Others

We can also do sampling easily.

```
#In dplyr
hflights %>% sample_n(10)
```

```
## Source: local data frame [10 x 22]
## 
##    Year Month DayofMonth DayOfWeek DepTime ArrTime UniqueCarrier FlightNum
## 1  2011     6          5         7    1909    2004            XE      3027
## 2  2011     8          5         5    1122    1240            XE      2277
## 3  2011    11         14         1    1445    1819            CO      1623
## 4  2011     7         21         4     758    1104            XE      2470
## 5  2011    12         17         6     722     920            XE      4676
## 6  2011     4         24         7    1910    2304            XE      2171
## 7  2011     8         18         4    1220    1316            CO      1555
## 8  2011     3         10         4    1551    1734            OO      1173
## 9  2011    12          9         5    1302    1643            CO      1653
## 10 2011     7          4         1    1152    1255            CO      1629
## Variables not shown: TailNum (chr), ActualElapsedTime (int), AirTime
##   (int), ArrDelay (int), DepDelay (int), Origin (chr), Dest (chr),
##   Distance (int), TaxiIn (int), TaxiOut (int), Cancelled (int),
##   CancellationCode (chr), Diverted (int), Speed (dbl)
```

```
#Or by fraction
hflights %>% sample_frac(0.25, replace=T)
```

```
## Source: local data frame [56,874 x 22]
## 
##    Year Month DayofMonth DayOfWeek DepTime ArrTime UniqueCarrier FlightNum
## 1  2011    11         18         5    1323    1520            OO      5249
## 2  2011    11         30         3     807    1126            CO      1160
## 3  2011     1         26         3    1141    1412            CO       546
## 4  2011     2         27         7    1114    1353            WN        11
## 5  2011     1         11         2     729    1026            XE      2586
## 6  2011     1          6         4    1907    2029            XE      2280
## 7  2011     3          28          1     1036    1610            CO        212
## 8  2011     8          25          4     1436    1525            XE       2451
## 9  2011     8          15          1     2056    2119            OO       1108
## 10 2011     4          22          5     2128    2235            WN        776
## ..  ...    ...         ...         ...    ...     ...            ...       ...
## Variables not shown: TailNum (chr), ActualElapsedTime (int), AirTime
##   (int), ArrDelay (int), DepDelay (int), Origin (chr), Dest (chr),
##   Distance (int), TaxiIn (int), TaxiOut (int), Cancelled (int),
##   CancellationCode (chr), Diverted (int), Speed (dbl)
```

```
#Like the str function in base R
glimpse(hflights)
```

```
## Observations: 227496
## Variables:
## $ Year              (int) 2011, 2011, 2011, 2011, 2011, 2011, 2011, 20...
## $ Month             (int) 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,...
## $ DayofMonth        (int) 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 1...
## $ DayOfWeek         (int) 6, 7, 1, 2, 3, 4, 5, 6, 7, 1, 2, 3, 4, 5, 6,...
## $ DepTime           (int) 1400, 1401, 1352, 1403, 1405, 1359, 1359, 13...
## $ ArrTime           (int) 1500, 1501, 1502, 1513, 1507, 1503, 1509, 14...
## $ UniqueCarrier     (chr) "AA", "AA", "AA", "AA", "AA", "AA", "AA", "A...
## $ FlightNum         (int) 428, 428, 428, 428, 428, 428, 428, 428, 428,...
## $ TailNum           (chr) "N576AA", "N557AA", "N541AA", "N403AA", "N49...
## $ ActualElapsedTime (int) 60, 60, 70, 70, 62, 64, 70, 59, 71, 70, 70, ...
## $ AirTime           (int) 40, 45, 48, 39, 44, 45, 43, 40, 41, 45, 42, ...
## $ ArrDelay          (int) -10, -9, -8, 3, -3, -7, -1, -16, 44, 43, 29,...
## $ DepDelay          (int) 0, 1, -8, 3, 5, -1, -1, -5, 43, 43, 29, 19, ...
## $ Origin            (chr) "IAH", "IAH", "IAH", "IAH", "IAH", "IAH", "I...
## $ Dest              (chr) "DFW", "DFW", "DFW", "DFW", "DFW", "DFW", "D...
## $ Distance          (int) 224, 224, 224, 224, 224, 224, 224, 224, 224,...
## $ TaxiIn            (int) 7, 6, 5, 9, 9, 6, 12, 7, 8, 6, 8, 4, 6, 5, 6...
## $ TaxiOut           (int) 13, 9, 17, 22, 9, 13, 15, 12, 22, 19, 20, 11...
## $ Cancelled         (int) 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,...
## $ CancellationCode  (chr) "", "", "", "", "", "", "", "", "", "", "", ...
## $ Diverted          (int) 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,...
## $ Speed             (dbl) 5.600000, 4.977778, 4.666667, 5.743590, 5.09...
```

10. Connecting with database

- dplyr can connect to a database as if the data was loaded into a data frame
- Instruction for create a database

```r
# connect to an SQLite database
my_db <- src_sqlite("my_db.sqlite3")

# connect to the "hflights" table in that database
flights_tbl <- tbl(my_db, "hflights")

# identical query using the database
flights_tbl %>%
    select(UniqueCarrier, DepDelay) %>%
    arrange(desc(DepDelay))

# ask dplyr for the SQL commands
flights_tbl %>%
    select(UniqueCarrier, DepDelay) %>%
    arrange(desc(DepDelay)) %>%
    explain()
```

References

- Hands-on dplyr tutorial for faster data manipulation in R
- A more comprehensive and advanced tutorial