null

null

null

Data Manipulation with `dplyr`

Load `dplyr` package, supposing it is already installed.

```r
require(dplyr)
```

Data

All the following exercises are based on the `nycflights13` data, taken from the `nycflights13` package. So first of all, install and load this package

```r
install.packages("nycflights13")
require(nycflights13)
```

The `nycflights13` package contains information about all flights that departed from NYC (e.g. EWR, JFK and LGA) in 2013: 336,776 flights in total.

```r
ls(pos = "package:nycflights13")
```

```
## [1] "airlines" "airports" "flights"  "planes"   "weather"
```

To help understand what causes delays, it includes a number of useful datasets:

- `flights`: information about all flights that departed from NYC
- `weather`: hourly meterological data for each airport;
- `planes`: construction information about each plane;
- `airports`: airport names and locations;
- `airlines`: translation between two letter carrier codes and names.

Let us explore the features of `flights` datasets, which will be used in the following exercises.

```r
data("flights")
```

flights

This dataset contains on-time data for all flights that departed from NYC (i.e. JFK, LGA or EWR) in 2013. The data frame has 16 variables and 336776 observations. The variables are organised as follow:

- Date of departure: `year`, `month`, `day`;
- Departure and arrival times (local tz): `dep_time`, `arr_time`;
- Departure and arrival delays, in minutes: `dep_delay`, `arr_delay` (negative times represent early departures/arrivals);

- Time of departure broken in to hour and minutes: `hour`, `minute`;
- Two letter carrier abbreviation: `carrier`;
- Plane tail number: `tailnum`;
- Flight number: `flight`;
- Origin and destination: `origin`, `dest`;
- Amount of time spent in the air: `air_time`;
- Distance flown: `distance`.

```
dim(flights)
```

```
## [1] 336776     16
```

```
head(flights)
```

```
##   year month day dep_time dep_delay arr_time arr_delay carrier tailnum flight
## 1 2013     1   1      517         2      830        11      UA  N14228   1545
## 2 2013     1   1      533         4      850        20      UA  N24211   1714
## 3 2013     1   1      542         2      923        33      AA  N619AA   1141
## 4 2013     1   1      544        -1     1004       -18      B6  N804JB    725
## 5 2013     1   1      554        -6      812       -25      DL  N668DN    461
## 6 2013     1   1      554        -4      740        12      UA  N39463   1696
##   origin dest air_time distance hour minute
## 1    EWR  IAH      227     1400    5     17
## 2    LGA  IAH      227     1416    5     33
## 3    JFK  MIA      160     1089    5     42
## 4    JFK  BQN      183     1576    5     44
## 5    LGA  ATL      116      762    5     54
## 6    EWR  ORD      150      719    5     54
```

```
str(flights)
```

```
## Classes 'tbl_df', 'tbl' and 'data.frame':    336776 obs. of  16 variables:
##  $ year     : int  2013 2013 2013 2013 2013 2013 2013 2013 2013 2013 ...
##  $ month    : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ day      : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ dep_time : int  517 533 542 544 554 554 555 557 557 558 ...
##  $ dep_delay: num  2 4 2 -1 -6 -4 -5 -3 -3 -2 ...
##  $ arr_time : int  830 850 923 1004 812 740 913 709 838 753 ...
##  $ arr_delay: num  11 20 33 -18 -25 12 19 -14 -8 8 ...
##  $ carrier  : chr  "UA" "UA" "AA" "B6" ...
##  $ tailnum  : chr  "N14228" "N24211" "N619AA" "N804JB" ...
##  $ flight   : int  1545 1714 1141 725 461 1696 507 5708 79 301 ...
##  $ origin   : chr  "EWR" "LGA" "JFK" "JFK" ...
##  $ dest     : chr  "IAH" "IAH" "MIA" "BQN" ...
##  $ air_time : num  227 227 160 183 116 150 158 53 140 138 ...
##  $ distance : num  1400 1416 1089 1576 762 ...
##  $ hour     : num  5 5 5 5 5 5 5 5 5 5 ...
##  $ minute   : num  17 33 42 44 54 54 55 57 57 58 ...
```

Select

Exercise 1

Extract the following information:

- month;
- day;
- air_time;
- distance.

Exercise 2

Extract all information about `flights` except hour and minute.

Exercise 3

Extract `tailnum` variable and rename it into `tail_num`

Filter

Exercise 1

Select all flights which delayed more than 1000 minutes at departure.

Exercise 2

Select all flights which delayed more than 1000 minutes at departure or at arrival.

Exercise 3

Select all flights which took off from “EWR” and landed in “IAH”.

Arrange

Exercise 1

Sort the flights in chronological order.

Exercise 2

Sort the flights by decreasing arrival delay.

Exercise 3

Sort the flights by origin (in alphabetical order) and decreasing arrival delay.

Mutate

Exercise 1

Add the following new variable to the `flights` dataset:

- the speed in miles per hour, named `speed` (`distance` / `air_time` * 60).

Consider that times are in minutes and distances are in miles.

Exercise 2

Add the following new variables to the `flights` dataset:

- the gained time in minutes (named `gain`), defined as the difference between delay at departure and delay at arrival;
- the gain time per hours, defined as `gain` / (`air_time` / 60)

Summarise

Exercise 1

Calculate minimum, mean and maximum delay at arrival. Remember to add `na.rm=TRUE` option to all calculations.

Group_by

Exercise 1

Calculate number of flights, minimum, mean and maximum delay at departure for flights by month. Remember to add `na.rm=TRUE` option to all calculations.

Exercise 2

Calculate number of flights (using `n()` operator), mean delay at departure and arrival for flights by origin. Remember to add `na.rm=TRUE` option to mean calculations.

Chain multiple operations (%>%)

Exercise 1

Calculate number of flights, minimum, mean and maximum delay at departure for flights by month. Remember to add `na.rm=TRUE` option to all calculations.

Exercise 2

Calculate the monthly mean gained time in minutes, where the gained time is defined as the difference between delay at departure and delay at arrival. Remember to add `na.rm=TRUE` option to mean calculations.

Exercise 3

For each destination, select all days where the mean delay at arrival is greater than 30 minutes. Remember to add `na.rm=TRUE` option to mean calculations.