Code Workshop 1

Yue Tian

4/3/2021

## Introduction

For the series of workshops we will cover data manipulation, data visualization and model implementation. This workshop will help you have more confidence with processing data in R.

Today, we will start the first part, data manipulation. Data manipulation includes

* Get the data you wanted
* Arrange the data
* Clean the data

We will use the following packages to help us.

* dplyr
* tidyverse
* nycflights13(this is the data we will use)

## Installing packages

#install.packages("dplyr")  
#install.packages("tidyverse")  
#install.packages("nycflights13")

## Call packages

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

library("tidyverse")

## ── Attaching packages ─────────────────────────────────────── tidyverse 1.3.0 ──

## ✓ ggplot2 3.3.2 ✓ purrr 0.3.4  
## ✓ tibble 3.0.4 ✓ stringr 1.4.0  
## ✓ tidyr 1.1.2 ✓ forcats 0.5.0  
## ✓ readr 1.4.0

## ── Conflicts ────────────────────────────────────────── tidyverse\_conflicts() ──  
## x dplyr::filter() masks stats::filter()  
## x dplyr::lag() masks stats::lag()

library("nycflights13")

## Preview of data

Whenever we have the data, the first thing we need to do is to exam it. - This is the data of flights of new york airports in 2013 - The attributes includes the flight month, day and time, the destination, the airliner, the delay time and their flight hours. - Using head() function only shows the first couple rows of the dataframe. This will help us save time when we are dealing with very large data

dat = flights  
head(dat)

## # A tibble: 6 x 19  
## year month day dep\_time sched\_dep\_time dep\_delay arr\_time sched\_arr\_time  
## <int> <int> <int> <int> <int> <dbl> <int> <int>  
## 1 2013 1 1 517 515 2 830 819  
## 2 2013 1 1 533 529 4 850 830  
## 3 2013 1 1 542 540 2 923 850  
## 4 2013 1 1 544 545 -1 1004 1022  
## 5 2013 1 1 554 600 -6 812 837  
## 6 2013 1 1 554 558 -4 740 728  
## # … with 11 more variables: arr\_delay <dbl>, carrier <chr>, flight <int>,  
## # tailnum <chr>, origin <chr>, dest <chr>, air\_time <dbl>, distance <dbl>,  
## # hour <dbl>, minute <dbl>, time\_hour <dttm>

## dplry basics

1. Filter the rows with filter() function

* filter() functions allows you to subset the data based on their values
* for example, I want to find all the flights in January 1st
* For the filter() functions the first argument is the name of the data and other arguments are the conditions of the filter

jan1 = filter(dat,month == 1,day ==1)  
print(jan1)

## # A tibble: 842 x 19  
## year month day dep\_time sched\_dep\_time dep\_delay arr\_time sched\_arr\_time  
## <int> <int> <int> <int> <int> <dbl> <int> <int>  
## 1 2013 1 1 517 515 2 830 819  
## 2 2013 1 1 533 529 4 850 830  
## 3 2013 1 1 542 540 2 923 850  
## 4 2013 1 1 544 545 -1 1004 1022  
## 5 2013 1 1 554 600 -6 812 837  
## 6 2013 1 1 554 558 -4 740 728  
## 7 2013 1 1 555 600 -5 913 854  
## 8 2013 1 1 557 600 -3 709 723  
## 9 2013 1 1 557 600 -3 838 846  
## 10 2013 1 1 558 600 -2 753 745  
## # … with 832 more rows, and 11 more variables: arr\_delay <dbl>, carrier <chr>,  
## # flight <int>, tailnum <chr>, origin <chr>, dest <chr>, air\_time <dbl>,  
## # distance <dbl>, hour <dbl>, minute <dbl>, time\_hour <dttm>

## Logic operators

At the Filter functions there is something like this “==” - This is a logic operator meaning equal. - In the filter function above, “month == 1” means that we want data that month is 1 - There are other logic operators - “>” : larger than - “<” : smaller than - “!=” : not equal

2 == 1

## [1] FALSE

3!= 2

## [1] TRUE

## Other logic operators

Other logic operator includes: - “&”: and (we don’t need this in filter() as a comma in filter in filter() means and) - “|”: or - For example we want to get the flight in January and February

filter(dat, month == 1 | month ==2)

## # A tibble: 51,955 x 19  
## year month day dep\_time sched\_dep\_time dep\_delay arr\_time sched\_arr\_time  
## <int> <int> <int> <int> <int> <dbl> <int> <int>  
## 1 2013 1 1 517 515 2 830 819  
## 2 2013 1 1 533 529 4 850 830  
## 3 2013 1 1 542 540 2 923 850  
## 4 2013 1 1 544 545 -1 1004 1022  
## 5 2013 1 1 554 600 -6 812 837  
## 6 2013 1 1 554 558 -4 740 728  
## 7 2013 1 1 555 600 -5 913 854  
## 8 2013 1 1 557 600 -3 709 723  
## 9 2013 1 1 557 600 -3 838 846  
## 10 2013 1 1 558 600 -2 753 745  
## # … with 51,945 more rows, and 11 more variables: arr\_delay <dbl>,  
## # carrier <chr>, flight <int>, tailnum <chr>, origin <chr>, dest <chr>,  
## # air\_time <dbl>, distance <dbl>, hour <dbl>, minute <dbl>, time\_hour <dttm>

# ## Exercise

* Find the flights that has an arrival delay is more than 1 hour (the attribute name is: arr\_delay)

flight\_delay = filter(dat,arr\_delay >= 60)

* Find the all United flights flew to Houston (attribute name: dest, Houston airports : IAH or HOU)

United\_huston = filter(dat,carrier == "UA", dest == "IAH" | dest == "HOU")

## Arrange rows using arrange()

-arrange() works like filter() and except it changes the order of rows rather than selecting them

* if you want the data in descending order use desc()
* arrange the flights based on their delay (attribute arr\_delay)

arrange(dat,arr\_delay)

## # A tibble: 336,776 x 19  
## year month day dep\_time sched\_dep\_time dep\_delay arr\_time sched\_arr\_time  
## <int> <int> <int> <int> <int> <dbl> <int> <int>  
## 1 2013 5 7 1715 1729 -14 1944 2110  
## 2 2013 5 20 719 735 -16 951 1110  
## 3 2013 5 2 1947 1949 -2 2209 2324  
## 4 2013 5 6 1826 1830 -4 2045 2200  
## 5 2013 5 4 1816 1820 -4 2017 2131  
## 6 2013 5 2 1926 1929 -3 2157 2310  
## 7 2013 5 6 1753 1755 -2 2004 2115  
## 8 2013 5 7 2054 2055 -1 2317 28  
## 9 2013 5 13 657 700 -3 908 1019  
## 10 2013 1 4 1026 1030 -4 1305 1415  
## # … with 336,766 more rows, and 11 more variables: arr\_delay <dbl>,  
## # carrier <chr>, flight <int>, tailnum <chr>, origin <chr>, dest <chr>,  
## # air\_time <dbl>, distance <dbl>, hour <dbl>, minute <dbl>, time\_hour <dttm>

* put them in a descending order

arrange(dat,desc(arr\_delay))

## # A tibble: 336,776 x 19  
## year month day dep\_time sched\_dep\_time dep\_delay arr\_time sched\_arr\_time  
## <int> <int> <int> <int> <int> <dbl> <int> <int>  
## 1 2013 1 9 641 900 1301 1242 1530  
## 2 2013 6 15 1432 1935 1137 1607 2120  
## 3 2013 1 10 1121 1635 1126 1239 1810  
## 4 2013 9 20 1139 1845 1014 1457 2210  
## 5 2013 7 22 845 1600 1005 1044 1815  
## 6 2013 4 10 1100 1900 960 1342 2211  
## 7 2013 3 17 2321 810 911 135 1020  
## 8 2013 7 22 2257 759 898 121 1026  
## 9 2013 12 5 756 1700 896 1058 2020  
## 10 2013 5 3 1133 2055 878 1250 2215  
## # … with 336,766 more rows, and 11 more variables: arr\_delay <dbl>,  
## # carrier <chr>, flight <int>, tailnum <chr>, origin <chr>, dest <chr>,  
## # air\_time <dbl>, distance <dbl>, hour <dbl>, minute <dbl>, time\_hour <dttm>

## Practice : find the flight with least time in air (attribute is air\_time)

arrange(dat,air\_time)

## # A tibble: 336,776 x 19  
## year month day dep\_time sched\_dep\_time dep\_delay arr\_time sched\_arr\_time  
## <int> <int> <int> <int> <int> <dbl> <int> <int>  
## 1 2013 1 16 1355 1315 40 1442 1411  
## 2 2013 4 13 537 527 10 622 628  
## 3 2013 12 6 922 851 31 1021 954  
## 4 2013 2 3 2153 2129 24 2247 2224  
## 5 2013 2 5 1303 1315 -12 1342 1411  
## 6 2013 2 12 2123 2130 -7 2211 2225  
## 7 2013 3 2 1450 1500 -10 1547 1608  
## 8 2013 3 8 2026 1935 51 2131 2056  
## 9 2013 3 18 1456 1329 87 1533 1426  
## 10 2013 3 19 2226 2145 41 2305 2246  
## # … with 336,766 more rows, and 11 more variables: arr\_delay <dbl>,  
## # carrier <chr>, flight <int>, tailnum <chr>, origin <chr>, dest <chr>,  
## # air\_time <dbl>, distance <dbl>, hour <dbl>, minute <dbl>, time\_hour <dttm>

## Select Columns using select()

* If you just want a few attributes from your dataframe simply use select() to get them

select(dat,year: day)#(inclusive)

## # A tibble: 336,776 x 3  
## year month day  
## <int> <int> <int>  
## 1 2013 1 1  
## 2 2013 1 1  
## 3 2013 1 1  
## 4 2013 1 1  
## 5 2013 1 1  
## 6 2013 1 1  
## 7 2013 1 1  
## 8 2013 1 1  
## 9 2013 1 1  
## 10 2013 1 1  
## # … with 336,766 more rows

* you can also exclude the columns
* Select all the columns except year day and month

select(dat,-(year:day))

## # A tibble: 336,776 x 16  
## dep\_time sched\_dep\_time dep\_delay arr\_time sched\_arr\_time arr\_delay carrier  
## <int> <int> <dbl> <int> <int> <dbl> <chr>   
## 1 517 515 2 830 819 11 UA   
## 2 533 529 4 850 830 20 UA   
## 3 542 540 2 923 850 33 AA   
## 4 544 545 -1 1004 1022 -18 B6   
## 5 554 600 -6 812 837 -25 DL   
## 6 554 558 -4 740 728 12 UA   
## 7 555 600 -5 913 854 19 B6   
## 8 557 600 -3 709 723 -14 EV   
## 9 557 600 -3 838 846 -8 B6   
## 10 558 600 -2 753 745 8 AA   
## # … with 336,766 more rows, and 9 more variables: flight <int>, tailnum <chr>,  
## # origin <chr>, dest <chr>, air\_time <dbl>, distance <dbl>, hour <dbl>,  
## # minute <dbl>, time\_hour <dttm>

* Other functions that helps select
* starts\_with(“abc”)
* For example: we want attributes about arrival

select(dat,starts\_with("arr"))

## # A tibble: 336,776 x 2  
## arr\_time arr\_delay  
## <int> <dbl>  
## 1 830 11  
## 2 850 20  
## 3 923 33  
## 4 1004 -18  
## 5 812 -25  
## 6 740 12  
## 7 913 19  
## 8 709 -14  
## 9 838 -8  
## 10 753 8  
## # … with 336,766 more rows

* ends\_with(“abc”) # this is similiar to starts\_with()

-contains(“abc”) -For example, we want to get information about delays

select(dat,contains("delay"))

## # A tibble: 336,776 x 2  
## dep\_delay arr\_delay  
## <dbl> <dbl>  
## 1 2 11  
## 2 4 20  
## 3 2 33  
## 4 -1 -18  
## 5 -6 -25  
## 6 -4 12  
## 7 -5 19  
## 8 -3 -14  
## 9 -3 -8  
## 10 -2 8  
## # … with 336,766 more rows

## Add new attributes with mutate()

* Sometimes we also want to add variables to our dataframe, for instance add an average. And we can use mutate() to add a new column
* notice: mutate adds new columns at the end of your dataframe
* Before we start, let’s make a narrow dataframe with less attributes

delay\_flights = select(dat, year:day, ends\_with("delay"), distance, air\_time)

* if I want to calculate the speed of the delaied flights

delay\_flights = select(dat, year:day, ends\_with("delay"), distance, air\_time)  
delay\_flights\_with\_speed = mutate(delay\_flights,speed = distance / air\_time\*60)

## Summaries using summarize()

* get the average delay for entire 2013
* na.rm = TRUE : we remove all NA’s in our dataframe

summarize(dat,delay = mean(dep\_delay,na.rm = TRUE))

## # A tibble: 1 x 1  
## delay  
## <dbl>  
## 1 12.6

* Use group\_by() with summarize
* Let’s calculate average delay per day

by\_day = group\_by(dat,year,month,day)  
summarize(by\_day,delay = mean(dep\_delay,na.rm = TRUE))

## `summarise()` regrouping output by 'year', 'month' (override with `.groups` argument)

## # A tibble: 365 x 4  
## # Groups: year, month [12]  
## year month day 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