Flying with dplyr

Kaitlyn Watson Group 9

Module 2 ICA4

Introduction

We will work with the data frame `flights`, which is included in the `nycflights13` package. To get started load `tidyverse` and `nycflights13`.

```
library(tidyverse)
library(nycflights13)
```

You may need to install `nycflights13`. Run `install.packages("nycflights13")`in your RStudio Console pane.

Package `nycflights13` contains a data frame `flights` that has on-time data for all flights that departed NYC (i.e. JFK, LGA or EWR) in 2013. Take a few minutes to examine the variables and their descriptions with regards to `flights`. Run `?flights` in your RStudio Console pane.

```
flights
```

```
# 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     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 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>
```

Object `flights` is a tibble. Another way to view the tibble in order to see all variables is with function `glimpse()`.

```
glimpse(flights)
```

```
Rows: 336,776
Columns: 19
$ year           <int> 2013, 2013, 2013, 2013, 2013, 2013, 2013, 2013, 2013, 2~
$ month          <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1~
$ day            <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1~
$ dep_time       <int> 517, 533, 542, 544, 554, 554, 555, 557, 557, 558, 558, ~
$ sched_dep_time <int> 515, 529, 540, 545, 600, 558, 600, 600, 600, 600, 600, ~
$ dep_delay      <dbl> 2, 4, 2, -1, -6, -4, -5, -3, -3, -2, -2, -2, -2, -2, -1~
$ arr_time       <int> 830, 850, 923, 1004, 812, 740, 913, 709, 838, 753, 849,~
$ sched_arr_time <int> 819, 830, 850, 1022, 837, 728, 854, 723, 846, 745, 851,~
$ arr_delay      <dbl> 11, 20, 33, -18, -25, 12, 19, -14, -8, 8, -2, -3, 7, -1~
$ carrier        <chr> "UA", "UA", "AA", "B6", "DL", "UA", "B6", "EV", "B6", "~
$ flight         <int> 1545, 1714, 1141, 725, 461, 1696, 507, 5708, 79, 301, 4~
$ tailnum        <chr> "N14228", "N24211", "N619AA", "N804JB", "N668DN", "N394~
$ origin         <chr> "EWR", "LGA", "JFK", "JFK", "LGA", "EWR", "EWR", "LGA",~
$ dest           <chr> "IAH", "IAH", "MIA", "BQN", "ATL", "ORD", "FLL", "IAD",~
$ air_time       <dbl> 227, 227, 160, 183, 116, 150, 158, 53, 140, 138, 149, 1~
$ distance       <dbl> 1400, 1416, 1089, 1576, 762, 719, 1065, 229, 944, 733, ~
$ hour           <dbl> 5, 5, 5, 5, 6, 5, 6, 6, 6, 6, 6, 6, 6, 6, 6, 5, 6, 6, 6~
$ minute         <dbl> 15, 29, 40, 45, 0, 58, 0, 0, 0, 0, 0, 0, 0, 0, 0, 59, 0~
$ time_hour      <dttm> 2013-01-01 05:00:00, 2013-01-01 05:00:00, 2013-01-01 0~
```

Comparison operators

Before you get started, take a few minutes to refresh on some of R's comparison operators detailed below.

Operator	Description
`>`	greater than
`<`	less than
`>=`	greater than or equal to
`<=`	less than or equal to
`==`	equal to
`!=`	not equal to
`&`	and (ex: `(5 > 7) & (6*7 == 42)` will return the value FALSE)
`|`	or (ex: `(5 > 7) | (6*7 == 42)` will return the value TRUE)
`%in%`	group membership

To evaluate group membership:

```r
# Generating the group:
set.seed(634789234)
die.out <- sample(x = 1:6, size = 10, replace = T)
die.out

#Checking for group membership:
die.out %in% c(3, 4)
c(3, 4) %in% die.out

die.out %in% c(1)
c(1) %in% die.out
```

dplyr

Package dplyr is based on the concept of functions as verbs that manipulate data frames.

Function	Action and purpose
`filter()`	choose rows matching a set of criteria
`slice()`	choose rows using indices
`select()`	choose columns by name
`pull()`	grab a column as a vector
`rename()`	rename specific columns
`arrange()`	reorder rows
`mutate()`	add new variables to the data frame
`transmute()`	create a new data frame with new variables
`distinct()`	filter for unique rows
`sample_n / sample_frac()`	randomly sample rows
`summarise()`	reduce variables to values

Exercise set 1

Make use of `%>%` operator and any of the functions in package `dplyr` to answer the following questions.

Question 1.

Filter `flights` for those in January with a destination of Detroit Metro (DTW) or Chicago O'Hare (ORD).

```
flights %>%
  filter(month==1) %>%
  filter(dest=="DTW"|dest=="ORD")
```

```
# A tibble: 2,056 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      554            558        -4      740            728
 2  2013     1     1      558            600        -2      753            745
 3  2013     1     1      602            605        -3      821            805
 4  2013     1     1      608            600         8      807            735
 5  2013     1     1      629            630        -1      824            810
 6  2013     1     1      656            700        -4      854            850
 7  2013     1     1      659            705        -6      907            913
 8  2013     1     1      709            700         9      852            832
 9  2013     1     1      715            713         2      911            850
10  2013     1     1      739            745        -6      918            930
# ... with 2,046 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>
```

Question 2.

Filter `flights` for those before April with a destination that is not Detroit Metro (DTW) and had an origin of JFK.

```
flights %>%
  filter(month<4) %>%
  filter(dest!="DTW" & origin=="JFK")
```

```
# A tibble: 27,009 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      542            540         2      923            850
 2  2013     1     1      544            545        -1     1004           1022
 3  2013     1     1      557            600        -3      838            846
 4  2013     1     1      558            600        -2      849            851
 5  2013     1     1      558            600        -2      853            856
 6  2013     1     1      558            600        -2      924            917
 7  2013     1     1      559            559         0      702            706
 8  2013     1     1      606            610        -4      837            845
 9  2013     1     1      611            600        11      945            931
10  2013     1     1      613            610         3      925            921
# ... with 26,999 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>
```

Question 3.

Choose rows 1, 3, 7, 20 from `flights`.

```
flights %>%
  slice(1,3,7,20)
```

```
# A tibble: 4 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      542            540         2      923            850
3  2013     1     1      555            600        -5      913            854
4  2013     1     1      601            600         1      844            850
# ... 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>
```

Question 4.

Arrange flights by distance and then by departure delay, with the sorting being in descending order in both cases. *Hint:* `desc()`

```
flights %>%
  arrange(desc(dep_delay)) %>%
  arrange(desc(distance))
```

```
# 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     2    23     1226            900       206     1746           1540
 3  2013     2     9     1206            900       186     1814           1540
 4  2013     8    25     1214           1000       134     1645           1440
 5  2013     1    18     1103            900       123     1635           1530
 6  2013     6    13     1153           1000       113     1649           1435
 7  2013     1     7     1042            900       102     1620           1530
 8  2013     1    23     1041            900       101     1652           1530
 9  2013     7     7     1128           1000        88     1553           1430
10  2013     1     6     1019            900        79     1558           1530
# ... 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>
```

Question 5.

Select only columns month, origin, and destination from `flights`.

```
flights %>%
  select(month, origin, dest)
```

```
# A tibble: 336,776 x 3
   month origin dest
   <int> <chr>  <chr>
 1     1 EWR    IAH
 2     1 LGA    IAH
 3     1 JFK    MIA
 4     1 JFK    BQN
 5     1 LGA    ATL
 6     1 EWR    ORD
 7     1 EWR    FLL
 8     1 LGA    IAD
 9     1 JFK    MCO
10     1 LGA    ORD
# ... with 336,766 more rows
```

Question 6.

Add a new variable to `flights` called `gain`, where `gain` is the arrival delay minus the departure delay.

```
flights %>%
  mutate(gain=arr_delay- dep_delay)
```

```
# A tibble: 336,776 x 20
    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 336,766 more rows, and 12 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>,
#   gain <dbl>
```

Question 7.

Use summarise to obtain the mean departure delay and mean arrival delay for all flights with an origin of EWR.

```
flights %>%
  filter(origin=="EWR") %>%
  summarise(mean_dep_delay = mean(dep_delay, na.rm=TRUE), mean_arr_delay = mean(arr_delay, na.rm=TRUE))
```

```
# A tibble: 1 x 2
  mean_dep_delay mean_arr_delay
           <dbl>          <dbl>
1           15.1           9.11
```

Exercise set 2

Grouping adds substantially to the power of the `dplyr` functions. We will focus on using `summarise()` with `group_by()`, but grouping also can be used with other `dplyr` functions.

Question 1.

Create a data frame which contains the number of flights and the mean arrival delay for flights on carrier UA (United Airlines) whose destination is O'Hare Airport (ORD). The number of flights and mean arrival delay is calculated separately for flights out of each of the origin airports.

```
flights %>%
  filter(carrier=="UA" & dest=="ORD") %>%
  group_by(origin) %>%
  summarise(mean_arr_delay = mean(arr_delay, na.rm=TRUE), n())
```

```
# A tibble: 2 x 3
  origin mean_arr_delay `n()`
  <chr>           <dbl> <int>
1 EWR              4.88  3822
2 LGA              7.53  3162
```

Question 2.

```
flights %>%
  filter(carrier=="UA" & origin=="EWR") %>%
  group_by(dest) %>%
  summarise(mean_hours=(mean(hour, na.rm=TRUE))) %>%
  arrange(desc(mean_hours))
```

```
# A tibble: 47 x 2
   dest  mean_hours
   <chr>      <dbl>
 1 BDL         22
 2 DTW         21
 3 BQN         20.1
 4 RDU         20
 5 SAT         17.3
 6 MSP         16.5
 7 ANC         16
 8 AUS         15.7
 9 CLE         15.3
10 PIT         15
# ... with 37 more rows
```

Create a data frame which contains the mean number of flight hours for carrier UA (United Airlines) originating from Liberty International Airport (EWR) to each unique destination.Arrange the data in descending order.

References

1. https://cran.r-project.org/web/packages/dplyr/vignettes/introduction.html