Data Manipulation with R

February 24, 2018

Introduction to R Data Processing Package “dplyr”

Package “**dplyr**” in R focussed on tools for working with data frames

- dplyr provides abstractions for basic data manipulation operations (called verbs)
- **Verbs** can be combined to achieve complicated data manipulation results using a series of simple data processing steps (by building a data manipulation pipeline)
- The approach is familiar to those who use UNIX/Linux and the "**dotadiw**" philosophy: Do One Thing and Do It Well

dplyr's Verbs for Data Processing

The verbs are:

- filter
- arrange
- select
- distinct
- mutate
- summarise

Data sets and R Code is available
`https://github.com/kiat/R-Examples`

```
#install.pacakges('dplyr')
library(dplyr)
delay.dat.houston <- read.csv("./Datasets/HoustonAirline.csv",
                        header=TRUE,
                        stringsAsFactors = FALSE)
# tbl_df allows for nice printing
delay.dat.houston <- tbl_df(delay.dat.houston)
```

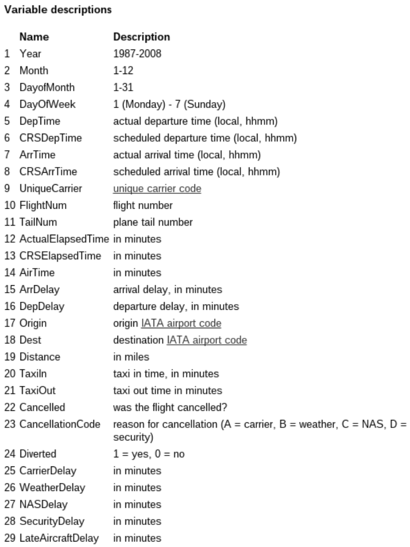
Take a look

```
> delay.dat.houston
# A tibble: 241,105 x 29
    Year Month DayofMonth DayOfWeek DepTime CRSDepTime ArrTime CRSArrTime
   <int> <int>      <int>     <int>   <int>      <int>   <int>      <int>
 1  2008     1          4         5    1910       1910    2025       2025
 2  2008     1          4         5    1345       1345    1453       1500
 3  2008     1          4         5     736        735     839        850
 4  2008     1          4         5    1603       1550    1647       1635
 5  2008     1          4         5    2105       2105    2151       2150
 6  2008     1          4         5     635        635     716        720
 7  2008     1          4         5    1331       1330    1411       1415
 8  2008     1          4         5    1850       1850    1936       1935
 9  2008     1          4         5     956       1000    1038       1045
10  2008     1          4         5     823        805     906        850
# ... with 241,095 more rows, and 21 more variables: UniqueCarrier <chr>,
#   FlightNum <int>, TailNum <chr>, ActualElapsedTime <int>,
#   CRSElapsedTime <int>, AirTime <int>, ArrDelay <int>, DepDelay <int>,
#   Origin <chr>, Dest <chr>, Distance <int>, TaxiIn <int>, TaxiOut <int>,
#   Cancelled <int>, CancellationCode <chr>, Diverted <int>, CarrierDelay <int>,
#   WeatherDelay <int>, NASDelay <int>, SecurityDelay <int>,
#   LateAircraftDelay <int>
```

Take a look

```
> airport.dat
# A tibble: 3,376 x 7
    iata              airport             city state country      lat       long
   <chr>                <chr>            <chr> <chr>   <chr>    <dbl>      <dbl>
 1   00M             Thigpen       Bay Springs    MS     USA 31.95376  -89.23450
 2   00R Livingston Municipal       Livingston    TX     USA 30.68586  -95.01793
 3   00V          Meadow Lake Colorado Springs    CO     USA 38.94575 -104.56989
 4   01G         Perry-Warsaw            Perry    NY     USA 42.74135  -78.05208
 5   01J     Hilliard Airpark         Hilliard    FL     USA 30.68801  -81.90594
 6   01M    Tishomingo County          Belmont    MS     USA 34.49167  -88.20111
 7   02A          Gragg-Wade           Clanton    AL     USA 32.85049  -86.61145
 8   02C              Capitol       Brookfield    WI     USA 43.08751  -88.17787
 9   02G    Columbiana County   East Liverpool    OH     USA 40.67331  -80.64141
10   03D     Memphis Memorial          Memphis    MO     USA 40.44726  -92.22696
# ... with 3,366 more rows
```

Variable Description



Variable descriptions

	Name	Description
1	Year	1987-2008
2	Month	1-12
3	DayofMonth	1-31
4	DayOfWeek	1 (Monday) - 7 (Sunday)
5	DepTime	actual departure time (local, hhmm)
6	CRSDepTime	scheduled departure time (local, hhmm)
7	ArrTime	actual arrival time (local, hhmm)
8	CRSArrTime	scheduled arrival time (local, hhmm)
9	UniqueCarrier	unique carrier code
10	FlightNum	flight number
11	TailNum	plane tail number
12	ActualElapsedTime	in minutes
13	CRSElapsedTime	in minutes
14	AirTime	in minutes
15	ArrDelay	arrival delay, in minutes
16	DepDelay	departure delay, in minutes
17	Origin	origin IATA airport code
18	Dest	destination IATA airport code
19	Distance	in miles
20	TaxiIn	taxi in time, in minutes
21	TaxiOut	taxi out time in minutes
22	Cancelled	was the flight cancelled?
23	CancellationCode	reason for cancellation (A = carrier, B = weather, C = NAS, D = security)
24	Diverted	1 = yes, 0 = no
25	CarrierDelay	in minutes
26	WeatherDelay	in minutes
27	NASDelay	in minutes
28	SecurityDelay	in minutes
29	LateAircraftDelay	in minutes

Filter - Filtering the Data

- filter is probably the most familiar verb
- filter is dplyr's version of R's subset() function
- filter returns all rows (observations) for which a logical condition holds

Filter - Inputs and Outputs

- Inputs: data.frame and logical expressions
- Output: data.frame
- All dplyr verbs behave similarly
- A data.frame is inputted, and a data.frame is outputted

Example 1

```
# Find all flight which occurred in Janurary
> filter(delay.dat.houston, Month==1)
# A tibble: 20,349 x 29
    Year Month DayofMonth DayOfWeek DepTime CRSDepTime ArrTime CRSArrTime
   <int> <int>      <int>     <int>   <int>      <int>   <int>      <int>
 1  2008     1          4         5    1910       1910    2025       2025
 2  2008     1          4         5    1345       1345    1453       1500
 3  2008     1          4         5     736        735     839        850
 4  2008     1          4         5    1603       1550    1647       1635
 5  2008     1          4         5    2105       2105    2151       2150
 6  2008     1          4         5     635        635     716        720
 7  2008     1          4         5    1331       1330    1411       1415
 8  2008     1          4         5    1850       1850    1936       1935
 9  2008     1          4         5     956       1000    1038       1045
10  2008     1          4         5     823        805     906        850
# ... with 20,339 more rows, and 21 more variables: UniqueCarrier <chr>,
#   FlightNum <int>, TailNum <chr>, ActualElapsedTime <int>, CRSElapsedTime <int>,
#   AirTime <int>, ArrDelay <int>, DepDelay <int>, Origin <chr>, Dest <chr>,
#   Distance <int>, TaxiIn <int>, TaxiOut <int>, Cancelled <int>,
#   CancellationCode <chr>, Diverted <int>, CarrierDelay <int>, WeatherDelay <int>,
#   NASDelay <int>, SecurityDelay <int>, LateAircraftDelay <int>
```

Example 2

```
# Using airport data, find a list of iata abbreviations for houston texas airports
> filter(airport.dat, state=='TX', city=='Houston')
# A tibble: 8 x 7
   iata                      airport    city state country      lat      long
  <chr>                        <chr>   <chr> <chr>   <chr>    <dbl>     <dbl>
1   DWH   David Wayne Hooks Memorial Houston    TX     USA 30.06186 -95.55278
2   EFD                   Ellington  Houston    TX     USA 29.60733 -95.15875
3   HOU              William P Hobby Houston    TX     USA 29.64542 -95.27889
4   IAH George Bush Intercontinental Houston    TX     USA 29.98047 -95.33972
5   IWS                 West Houston Houston    TX     USA 29.81819 -95.67261
6   LVJ                      Clover  Houston    TX     USA 29.52131 -95.24217
7   SGR   Sugar Land Municipal/Hull  Houston    TX     USA 29.62225 -95.65653
8   SPX                 Houston-Gulf Houston    TX     USA 29.50836 -95.05133
```

Introduction to R Data Processing Package “dplyr”

Package “**dplyr**” in R focussed on tools for working with data frames

- Find the subset of flight departing from Hobby Airport **“HOU”** for which the Actual Elapsed Time was greater than the CRS Elapsed Time (ActualElapsedTime $>$ CRSElapsedTime)
- Find the subset of flights departing on the weekend.

R Command

```
# Find the subset of flight departing from
# Hobby Airport "HOU" for which the Actual
# Elapsed Time was greater than the CRS Elapsed Time.
> filter(delay.dat.houston,
       Origin == 'HOU', # iata code for Hobby
       ActualElapsedTime > CRSElapsedTime)
```

R Command

```
# Find the subset of flights departing on the weekend.
> filter(delay.dat.houston, DayOfWeek == 6 | DayOfWeek == 7)

# another alternative
> filter(delay.dat.houston,  DayOfWeek %in% c(6,7))
```

arrange

- arrange, like filter, operates on data.frame rows
- arrange is used for sorting data.frame rows w.r.t. a given column(s)

arrange

```
> arrange(delay.dat.houston, DayofMonth)
# A tibble: 241,105 x 29
    Year Month DayofMonth DayOfWeek DepTime CRSDepTime ArrTime CRSArrTime
   <int> <int>      <int>     <int>   <int>      <int>   <int>      <int>
 1  2008     1          1         2    1531       1525    1626       1622
 2  2008     1          1         2    1848       1850    2022       2025
 3  2008     1          1         2    1024       1025    1353       1352
 4  2008     1          1         2     707        705     818        822
 5  2008     1          1         2    1047       1045    1423       1415
 6  2008     1          1         2    1110       1110    1237       1240
 7  2008     1          1         2    1653       1655    2038       2058
 8  2008     1          1         2    2013       1950    2335       2319
 9  2008     1          1         2    1212       1220    1454       1512
10  2008     1          1         2    1021       1020    1136       1132
# ... with 241,095 more rows, and 21 more variables: UniqueCarrier <chr>,
#   FlightNum <int>, TailNum <chr>, ActualElapsedTime <int>, CRSElapsedTime <int>,
#   AirTime <int>, ArrDelay <int>, DepDelay <int>, Origin <chr>, Dest <chr>,
#   Distance <int>, TaxiIn <int>, TaxiOut <int>, Cancelled <int>,
#   CancellationCode <chr>, Diverted <int>, CarrierDelay <int>, WeatherDelay <int>,
#   NASDelay <int>, SecurityDelay <int>, LateAircraftDelay <int>
```

arrange

```
> arrange(delay.dat.houston, desc(Month), desc(DayofMonth))
# A tibble: 241,105 x 29
    Year Month DayofMonth DayOfWeek DepTime CRSDepTime ArrTime CRSArrTime
   <int> <int>      <int>     <int>   <int>      <int>   <int>      <int>
 1  2008    12         31         3     707        705     810        815
 2  2008    12         31         3    1256       1245    1355       1400
 3  2008    12         31         3    1553       1550    1632       1635
 4  2008    12         31         3    1801       1750    1841       1835
 5  2008    12         31         3    1101       1055    1141       1140
 6  2008    12         31         3    1325       1315    1408       1400
 7  2008    12         31         3     948        950    1113       1125
 8  2008    12         31         3    1555       1555    1719       1730
 9  2008    12         31         3    1952       1955    2124       2135
10  2008    12         31         3    1755       1720    1936       1910
# ... with 241,095 more rows, and 21 more variables: UniqueCarrier <chr>,
#   FlightNum <int>, TailNum <chr>, ActualElapsedTime <int>, CRSElapsedTime <int>,
#   AirTime <int>, ArrDelay <int>, DepDelay <int>, Origin <chr>, Dest <chr>,
#   Distance <int>, TaxiIn <int>, TaxiOut <int>, Cancelled <int>,
#   CancellationCode <chr>, Diverted <int>, CarrierDelay <int>, WeatherDelay <int>,
#   NASDelay <int>, SecurityDelay <int>, LateAircraftDelay <int>
```

arrange

```
> arrange(delay.dat.houston, desc(Month), desc(DayofMonth))
# A tibble: 241,105 x 29
    Year Month DayofMonth DayOfWeek DepTime CRSDepTime ArrTime CRSArrTime
   <int> <int>      <int>     <int>   <int>      <int>   <int>      <int>
 1  2008    12         31         3     707        705     810        815
 2  2008    12         31         3    1256       1245    1355       1400
 3  2008    12         31         3    1553       1550    1632       1635
 4  2008    12         31         3    1801       1750    1841       1835
 5  2008    12         31         3    1101       1055    1141       1140
 6  2008    12         31         3    1325       1315    1408       1400
 7  2008    12         31         3     948        950    1113       1125
 8  2008    12         31         3    1555       1555    1719       1730
 9  2008    12         31         3    1952       1955    2124       2135
10  2008    12         31         3    1755       1720    1936       1910
# ... with 241,095 more rows, and 21 more variables: UniqueCarrier <chr>,
#   FlightNum <int>, TailNum <chr>, ActualElapsedTime <int>, CRSElapsedTime <int>,
#   AirTime <int>, ArrDelay <int>, DepDelay <int>, Origin <chr>, Dest <chr>,
#   Distance <int>, TaxiIn <int>, TaxiOut <int>, Cancelled <int>,
#   CancellationCode <chr>, Diverted <int>, CarrierDelay <int>, WeatherDelay <int>,
#   NASDelay <int>, SecurityDelay <int>, LateAircraftDelay <int>
```

select

- select is like filter but for columns
- select is used for keeping/dropping a subset of variables/columns

R Command

Try out the following examples using select

```
select(delay.dat.houston, Year, Month, DayofMonth)
select(delay.dat.houston,Year:DayofMonth)
select(delay.dat.houston,-(Year:DayofMonth))
```

select

Here we use the contains helper:

```
> select(delay.dat.houston, contains('Dep'))
# A tibble: 241,105 x 3
   DepTime CRSDepTime DepDelay
     <int>      <int>    <int>
 1    1910       1910        0
 2    1345       1345        0
 3     736        735        1
 4    1603       1550       13
 5    2105       2105        0
 6     635        635        0
 7    1331       1330        1
 8    1850       1850        0
 9     956       1000       -4
10     823        805       18
# ... with 241,095 more rows
```

select helper

Create a select statement using

- one_of helper
- ends_with helper

select helper

```
select(delay.dat.houston,
           one_of('UniqueCarrier',
           'FlightNum'))

select(delay.dat.houston,
           ends_with('Time'))
```

distinct

- distinct finds unique values of a variable
- distinct returns the first observation/row containing each value

distinct

```
> distinct(delay.dat.houston, Month)
# A tibble: 12 x 1
   Month
   <int>
 1     1
 2     2
 3     3
 4     4
 5     5
 6     6
 7     7
 8     8
 9     9
10    10
11    11
12    12
```

distinct

```
> distinct(delay.dat.houston, Month,.keep_all=TRUE)
# A tibble: 12 x 29
    Year Month DayofMonth DayOfWeek DepTime CRSDepTime ArrTime CRSArrTime
   <int> <int>      <int>     <int>   <int>      <int>   <int>      <int>
 1  2008     1          4         5    1910       1910    2025       2025
 2  2008     2          3         7     758        800     903        915
 3  2008     3          3         1     800        800     920        915
 4  2008     4          4         5     900        900    1027       1010
 5  2008     5          4         7     857        900    1008       1010
 6  2008     6          3         2    1951       1935    2050       2040
 7  2008     7          3         4    1935       1935    2032       2040
 8  2008     8          3         7    1940       1935    2049       2040
 9  2008     9          3         3     804        805     857        910
10  2008    10          3         5     715        720     828        845
11  2008    11          4         2    1834       1825    1933       1935
12  2008    12          3         3    1845       1825    1958       1935
# ... with 21 more variables: UniqueCarrier <chr>, FlightNum <int>, TailNum <chr>,
#   ActualElapsedTime <int>, CRSElapsedTime <int>, AirTime <int>, ArrDelay <int>,
#   DepDelay <int>, Origin <chr>, Dest <chr>, Distance <int>, TaxiIn <int>,
#   TaxiOut <int>, Cancelled <int>, CancellationCode <chr>, Diverted <int>,
#   CarrierDelay <int>, WeatherDelay <int>, NASDelay <int>, SecurityDelay <int>,
#   LateAircraftDelay <int>
```

distinct

```
> distinct(delay.dat.houston, Month,DayOfWeek)
# A tibble: 84 x 2
   Month DayOfWeek
   <int>     <int>
 1     1         5
 2     1         6
 3     1         7
 4     1         1
 5     1         2
 6     1         3
 7     1         4
 8     2         7
 9     2         1
10     2         2
# ... with 74 more rows
```

Combination of verbs

You can combine distinct with the select verb from previous. What do you think the following will do?

```
select(
        distinct(
                arrange(
                        filter(delay.dat.houston,DayOfWeek==6),
                        desc(ActualElapsedTime)),
                        UniqueCarrier,.keep_all = TRUE),
                        UniqueCarrier,ActualElapsedTime)
```

Reading from the inside out we can see it:

- Only considers flights departing on Saturday
- Arranges these by ActucalElapsedTime in decrease order
- Selects the first row for each carrier
- In total this gives the largest ActualElapsedTime for Saturday departing flights for each carrier.
- distinct returns the first observation/row containing each value

Combination of verbs

We can do the previous example with the chaining

```
delay.dat.houston %>%
  filter(DayOfWeek == 6) %>%
  arrange(desc(ActualElapsedTime)) %>%
  distinct(UniqueCarrier,.keep_all=TRUE) %>%
  select(UniqueCarrier,ActualElapsedTime)
```

Chain together the verbs we've seen so far to:

- Find a list of Origin Airports
- Find a list of (Origin,Destination) pairs
- Find the Origin airport which had the largest departure delay in the month of January
- Find the largest departure delay for each carrier for each month

Combination of verbs

```r
# Find a list of the distinct Origin airports
delay.dat.houston %>%
  distinct(Origin)

# Find a list of distinct (Origin, Dest) pairs
delay.dat.houston %>%
  distinct(Origin, Dest)

# Origin airport with largest Janurary departure delay
delay.dat.houston %>%
  filter(Month==1) %>%
  arrange(desc(DepDelay)) %>%
  select(Month,Origin, DepDelay) %>%
  distinct(Origin,.keep_all = TRUE)
```

Combination of verbs

```
# largest departure delay for each carrier for each month
delay.dat.houston %>%
  arrange(Month,desc(DepDelay)) %>%
  select(Month,UniqueCarrier,DepDelay) %>%
  distinct(Month,UniqueCarrier,.keep_all=TRUE)
```

Two verbs: mutate and summarise

- mutate allows us to create new variables

summarise:

- summarise let's us compute summary statistics on groups of data
- summarise is used in conjunction with the group by verb

summarise

```
# Basic example with no grouping
> delay.dat.houston %>%
    summarise(MeanDistance = mean(Distance,na.rm=TRUE))
# A tibble: 1 x 1
  MeanDistance
         <dbl>
1     778.5913
```

summarise

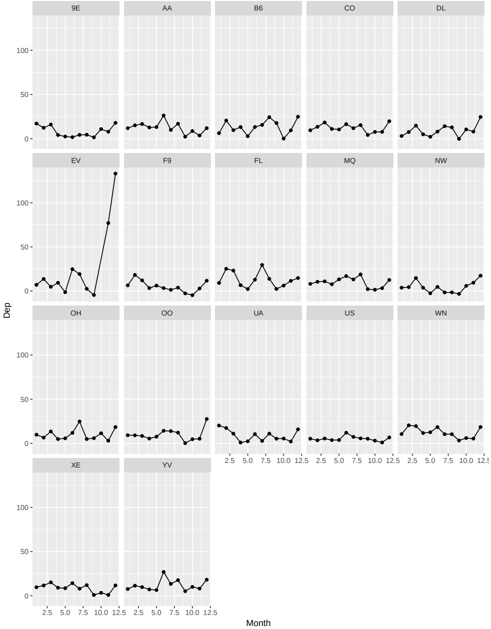
```
# With grouping
# n() is dplyr function counts # obs in each group
> delay.dat.houston %>%
+   group_by(UniqueCarrier) %>%
+   summarise(
+     MeanDistance=mean(Distance,na.rm=TRUE),
+     NFlights = n())
# A tibble: 17 x 3
   UniqueCarrier MeanDistance NFlights
           <chr>        <dbl>    <int>
 1            9E     630.9294     2721
 2            AA     586.7512     4325
 3            B6    1428.0000      944
 4            CO    1055.0753    85642
 5            DL     690.3982     1517
 6            EV     704.0464      194
 7            F9     861.0000      846
 8            FL     696.0000     1792
 9            MQ     247.0000     2425
10            NW    1013.0745     1598
11            OH     912.1431     1013
12            OO    1007.0786     2595
13            UA    1019.8512     2325
14            US     965.7900     1924
15            WN     562.0526    48968
16            XE     611.3961    80194
17            YV     991.3463     2082
```

We could also redo our previous example, finding the largest departure delay for each carrier for each month

```
> delay.dat.houston %>%
+   group_by(Month, UniqueCarrier) %>%
+   summarise(MaxDepDelay = max(DepDelay,na.rm=TRUE)) %>%
+   head(5)
# A tibble: 5 x 3
# Groups:   Month [1]
  Month UniqueCarrier MaxDepDelay
  <int>         <chr>       <dbl>
1     1            9E         356
2     1            AA         234
3     1            B6         183
4     1            CO         475
5     1            DL         131
```

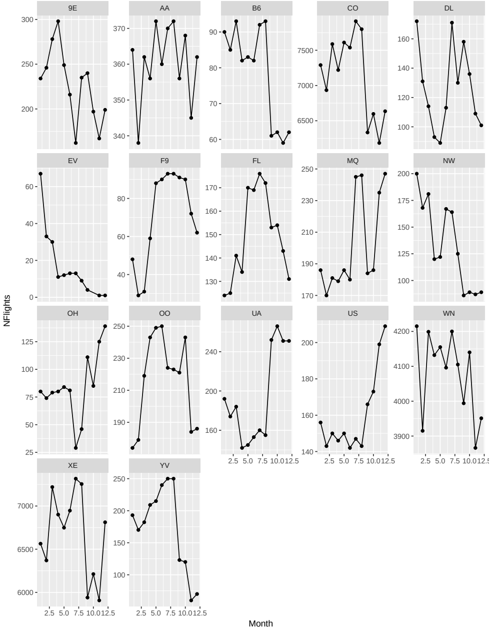
- For each carrier plot the average Departure delay for each month.
- Do you notice anything strange? What might be the cause?
- Hint: Use summarise and faceting
- Hint: For each carrier also plot the number of flights per month.

```r
library(ggplot2)
delay.dat.houston %>%
  group_by(Month,UniqueCarrier) %>%
  summarise(
    Dep = mean(DepDelay,na.rm=TRUE)
  ) -> tmp


qplot(Month,Dep,data=tmp) +
  geom_line() +
  facet_wrap(~UniqueCarrier)
```


9E
AA
B6
CO
DL
EV
F9
FL
MQ
NW
OH
OO
UA
US
WN
XE
YV
Dep
0
50
100
Month
2.5
5.0
7.5
10.0
12.5

What could cause this? Try this:

```
delay.dat.houston %>%
  group_by(Month,UniqueCarrier) %>%
  summarise(
    NFlights = n()
  ) -> tmp


qplot(Month,NFlights,data=tmp) +
  geom_line() +
  facet_wrap(~UniqueCarrier,scale='free_y')
```


9E
300
250
200
AA
370
360
350
340
B6
90
80
70
60
CO
7500
7000
6500
DL
160
140
120
100
EV
60
40
20
0
F9
80
60
40
FL
170
160
150
140
130
MQ
250
230
210
190
170
NW
200
175
150
125
100
OH
125
100
75
50
25
OO
250
230
210
190
UA
240
200
160
US
200
180
160
140
WN
4200
4100
4000
3900
XE
7000
6500
6000
YV
250
200
150
100
2.5
5.0
7.5
10.0
12.5
NFlights
Month

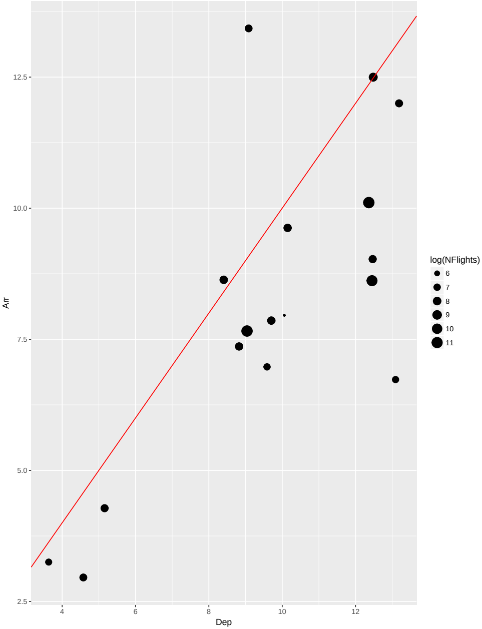
- Find the percent of flights cancelled for each carrier.
- Use summarise to get total number of flights for each carrier (UniqueCarrier) and the total number of cancelled flights
- Create a new variable PercentCancelled based on the results above
- Return a data.frame with only UniqueCarrier and PercentCancelled

```
delay.dat.houston %>%
  group_by(UniqueCarrier) %>%
  summarise(
    NFlights = n(),
    NCancelled = sum(Cancelled)) %>%
  mutate(
    PercentCancelled = (NCancelled/NFlights)*100) %>%
  select(UniqueCarrier,
    PercentCancelled)
```

```
# A tibble: 17 x 2
   UniqueCarrier PercentCancelled
           <chr>            <dbl>
 1            9E         3.601617
 2            AA         4.138728
 3            B6         3.283898
 4            CO         1.122113
 5            DL         2.834542
 6            EV         3.092784
 7            F9         1.418440
 8            FL         1.450893
 9            MQ         3.835052
10            NW         1.251564
11            OH         3.849951
12            OO         2.581888
13            UA         2.408602
14            US         1.663202
15            WN         2.783450
16            XE         2.188443
17            YV         3.073967
```

- For each Destination find the average Arrival and Departure delay; create associated variables AvgArrDel, AvgDepDel
- Plot AvgArrDel vs AvgDepDel for the three largest carriers (largest in terms of number of flights)
- Plot AvgArrDel vs AvgDepDel for all carriers. Use point size to indicate carrier size

```
delay.dat.houston %>%
  group_by(UniqueCarrier) %>%
  summarise(
    Dep = mean(DepDelay,na.rm=TRUE),
    Arr = mean(ArrDelay,na.rm=TRUE),
    NFlights = n()
  ) %>%
  select(Dep,Arr,NFlights) -> tmp


qplot(Dep,
      Arr,
      data=tmp,
      size=log(NFlights))+
  geom_abline(intercept=0,slope=1,color='red')
```


Arr
12.5
10.0
7.5
5.0
2.5
Dep
4
6
8
10
12
log(NFlights)
● 6
● 7
● 8
● 9
● 10
● 11

For our final dplyr stop we'll look at it's merging capabilities. Let's start by reading in some more toy datasets

```
people.info <- read.table('./Datasets/mergedata/PeopleInfo.csv',
                          sep=',',
                          header=TRUE)
occup.info <-
↪  read.table('./Datasets/mergedata/OccupationInfo.csv',
                          sep=',',
                          header=TRUE)
```

People Dataset

```
> people.info
    ID    Last DOB
1 1718   Jones  85
2 1817   Smith  72
3 1558 Wallace  50
4 1742   Marks  90
> occup.info
    ID      Title Office
1 1558 Supervisor    101
2 1718      Clerk    110
3 2234 Accountant    502
4  943     Doctor    409
5 1119    Manager    404
```

Basic Join

dplyr's basic merging functions are:

- **inner_join** : return all rows from x where there are matching values in y, and all columns from x and y. If there are multiple matches between x and y, all combination of the matches are returned.
- **left_join** : return all rows from x, and all columns from x and y. Rows in x with no match in y will have NA values in the new columns. If there are multiple matches between x and y, all combinations of the matches are returned.
- **right_join** :

INNER Join

Venn Diagram for Join Operation.

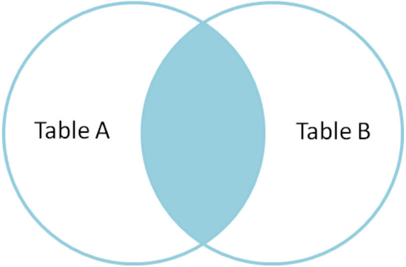


Figure: TableA INNER JOIN TableB

Full LEFT OUTER Join

Full Left Outer Join.

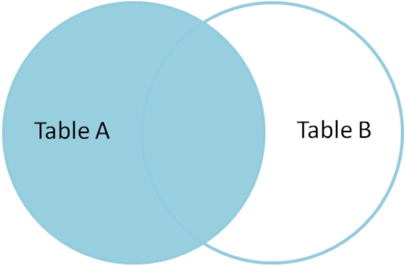


Figure: TableA LEFT OUTER JOIN TableB

Join

```
# What do you think the following snippets will do
# Try to guess before running, then run to confirm
left_join(people.info, occup.info)
right_join(people.info, occup.info)
inner_join(people.info, occup.info)

# Do the following return the same data set?
left_join(people.info, occup.info)
right_join(occup.info, people.info)

# Do you think this will work?
people.info %>% left_join(occup.info)
```

Other Join

- **semi_join** returns only lhs columns, and only for ids common to both
- **anti_join** returns only lhs columns, and only for ids *not* common to both
- **full_join** returns all columns, for all ids, merging with inner/left/right when applicable

LEFT OUTER JOIN Join

TableA LEFT OUTER JOIN TableB

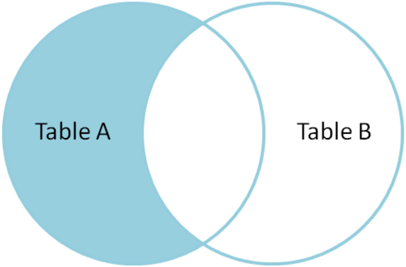


Figure: TableA LEFT OUTER JOIN TableB

FULL OUTER Join

TableA FULL OUTER JOIN TableB

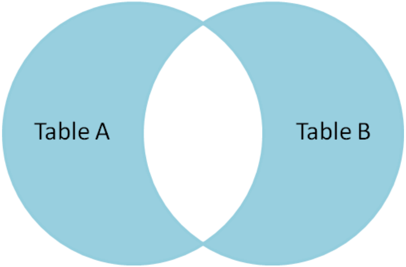


Figure: TableA FULL OUTER JOIN TableB

full join

TableA FULL OUTER JOIN TableB

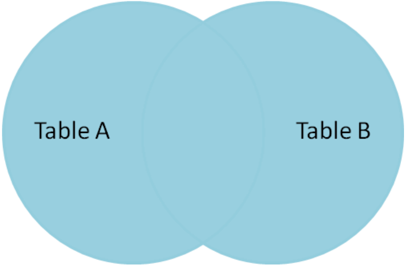


Figure: TableA FULL OUTER JOIN TableB

Join

```
semi_join(people.info, occup.info)
anti_join(people.info, occup.info)
full_join(people.info, occup.info)
```

Join

Merge the airport and delay data so that we have state/city information regarding the destination
Hint: use left_join with by=c(”Dest” = ”iata”)

Join

```
delay.dat.houston %>%
  left_join(airport.dat,
            by=c("Dest" = 'iata'))
```

Join

Calculate the number of flights to each destination state

For each carrier, for which state do they have the largest average delay?

Join

```
# one option
delay.dat.houston %>%
  left_join(airport.dat,
            by=c("Dest" = 'iata')) %>%
  group_by(state) %>%
  summarise(
    NFlights = n()
  ) %>%
  select(state,NFlights)
```

```
# A tibble: 41 x 2
   state NFlights
   <chr>    <int>
 1    AK      206
 2    AL     5778
 3    AR     2911
 4    AZ     7568
 5    CA    17448
 6    CO     7869
 7    CT      120
 8    FL    18951
 9    GA     9533
10    HI      702
# ... with 31 more rows
```

```
delay.dat.houston %>%
  left_join(airport.dat,
            by=c("Dest" = 'iata')) %>%
  group_by(UniqueCarrier, state) %>%
  summarise(
    AvgDelay = mean(DepDelay,na.rm=TRUE)
  ) %>%
  select(state,UniqueCarrier, AvgDelay) %>%
  arrange(UniqueCarrier, desc(AvgDelay)) %>%
  distinct(UniqueCarrier,.keep_all = TRUE)
```