Using Databases

Will Doyle

Introduction: What's a Database?

Databases are the primary way large organizations and governments organize and store data. What makes a database distinct from a dataset is that it stores data in multiple tables, which are called "flat files" in the parlance of databases. The relationships between each of the tables is recorded in a schema. Some terms from database land and their translation in R:

- A table (sometimes called a file) in a database is like a data frame in R: a 2 dimensional array. The 2 dimensions are referred to as "records" and "fields". These are typically organized by their primary id, or unit of analysis.
- A record is what we could call a case or a unit of observation. A record is composed of fields.
- A field is what we would call a variable name. A field is composed of multiple records
- A relational database is a set of tables that are linked by a set of common identifiers. Not every table can be linked to every other table, and tables do not need to have the same primary ids.
- A database management system is software that provides a way to interact with a database.

The DBMS that is used most frequently is SQL and its variants, MYSQL and sqlite. Mariadb is a newer "sort of" variant of sql. Google BigQuery is one used by many companies, while Microsoft Access is a plague that has been set upon us for our transgressions. We hope someday the curse will be lifted.

An R analyst typically doesn't want to be a database person– they just want to get the data out in a way that they can use for data analysis. It's not worth it to store data on your own computer in a database– it will just slow you down. Instead, this lesson is intended for when you have a chance to interact with a truly large database.

NYC Flights

Today we'll use the `nyclfights13` database, which contains information on every single flight departing New York City in 2013, including airline information, airport information, flight information, plane information and weather information.

```r
# Mostly taken from : http://cran.r-project.org/web/packages/dplyr/vignettes/databases.html circa 2014

# Will need: nycflights13 RSQLite,

#Get libraries
library(tidyverse)
```

```
## Warning: package 'tidyverse' was built under R version 3.5.3

## -- Attaching packages --------------------------------------------------------------------------- tidyve

## v ggplot2 3.2.1     v purrr   0.3.2
## v tibble  2.1.3     v dplyr   0.8.1
## v tidyr   0.8.3     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.4.0

## Warning: package 'ggplot2' was built under R version 3.5.3
## Warning: package 'tibble' was built under R version 3.5.3

## Warning: package 'tidyr' was built under R version 3.5.3

## Warning: package 'purrr' was built under R version 3.5.3

## Warning: package 'dplyr' was built under R version 3.5.3

## Warning: package 'stringr' was built under R version 3.5.3

## Warning: package 'forcats' was built under R version 3.5.3

## -- Conflicts --------------------------------------------------------------------------- tidyverse_c
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```r
library(nycflights13)
```

```
## Warning: package 'nycflights13' was built under R version 3.5.3
```

```r
library(RSQLite)
```

```
## Warning: package 'RSQLite' was built under R version 3.5.3
```

The first thing we'll do is get all of the various tables open. They're stored in `data()` from the `nycflights` library.

```r
# Data sources from NYC flights
data(flights)
data(airlines)
data(airports)
data(weather)
data(planes)
```

Connecting to a database

I'm going to do something kind of unusual for this class: I'm going to create a database from existing flat files. You should NEVER do this– it's just for teaching. I'm using the RSQLite package to open a connection to a database. In your work, you would just open the connection, many times through a secure network. You'll need instructions from a database admin on how to do this.

`con` below refers to our connection to a database. It remains open until we close it. It will be based on the properties of the database we're trying to access.

```r
con <- dbConnect(RSQLite::SQLite(), ":memory:")

#Write flights tables to database (you won't usually do this)
dbWriteTable(con,
             "flights",
             as.data.frame(flights))

dbWriteTable(con,
             "planes",
             as.data.frame(planes))

dbWriteTable(con,
             "airlines",
             as.data.frame(airlines))
```

```
dbWriteTable(con,
             "weather",
             as.data.frame(weather))

dbWriteTable(con,
             "airports",
             as.data.frame(airports))
```

Tables and Fields

Now I have an open connection to a database that contains multiple tables. Let's ask which tables we have access to:

```
#List tables in database
dbListTables(con)
```

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

For some of these tables, what fields are included?

```
#List fields in various tables
dbListFields(con,"airlines")
```

```
## [1] "carrier" "name"
```

```
dbListFields(con,"flights")
```

```
##  [1] "year"           "month"          "day"            "dep_time"
##  [5] "sched_dep_time" "dep_delay"      "arr_time"       "sched_arr_time"
##  [9] "arr_delay"      "carrier"        "flight"         "tailnum"
## [13] "origin"         "dest"           "air_time"       "distance"
## [17] "hour"           "minute"         "time_hour"
```

```
dbListFields(con,"planes")
```

```
## [1] "tailnum"      "year"         "type"         "manufacturer"
## [5] "model"        "engines"      "seats"        "speed"
## [9] "engine"
```

```
dbListFields(con,"weather")
```

```
##  [1] "origin"     "year"       "month"      "day"        "hour"
##  [6] "temp"       "dewp"       "humid"      "wind_dir"   "wind_speed"
## [11] "wind_gust"  "precip"     "pressure"   "visib"      "time_hour"
```

Verbs for Databases

SQL is its own language. One of the main things people do with SQL is to generate requests. Below, we create request text which asks for every field from the flights table, then sends that query to the database. Once we get the return back, we turn it into a data frame. As a matter of good practice, we also clear that request. Click here for a cheat sheet on SQL queries.

```
#Generate a SQL request to a database
req_text<-"Select * from airlines"

#Send query through connection
req<-dbSendQuery(con,req_text)
```

```r
#Generate dataframe from results
req_df<-dbFetch(req,n=-1)

#Good practice: clear request
dbClearResult(req)
```

Let's take a look at the generated data frame.

```r
dim(req_df)
```

```
## [1] 16  2
```

```r
head(req_df,20)
```

```
##    carrier                        name
## 1       9E           Endeavor Air Inc.
## 2       AA      American Airlines Inc.
## 3       AS        Alaska Airlines Inc.
## 4       B6             JetBlue Airways
## 5       DL        Delta Air Lines Inc.
## 6       EV    ExpressJet Airlines Inc.
## 7       F9      Frontier Airlines Inc.
## 8       FL AirTran Airways Corporation
## 9       HA      Hawaiian Airlines Inc.
## 10      MQ                   Envoy Air
## 11      OO       SkyWest Airlines Inc.
## 12      UA       United Air Lines Inc.
## 13      US             US Airways Inc.
## 14      VX              Virgin America
## 15      WN      Southwest Airlines Co.
## 16      YV          Mesa Airlines Inc.
```

Quick Exercise: Get the weather table and put it in a data frame

Select and From

Many times we want to generate a request that will only give us part of some tables. We'd also like to join tables together in many circumstances. Below we build a SQL request that includes the key verbs SELECT, FROM, JOIN and WHERE.

Let's say that we're interested in which airlines might have longer delays.

```r
req1_text<-"SELECT f.dep_delay from flights f
            "
req1<-dbSendQuery(con,req1_text)

req1_df<-dbFetch(req1,n=-1)

dbClearResult(req1)

head(req1_df)
```

```
##   dep_delay
## 1         2
## 2         4
## 3         2
## 4        -1
## 5        -6
## 6        -4
```

We then need to get the airline data from the airline table

```
req1_text <- "SELECT a.carrier, a.name
FROM airlines a"
```

Notice how the SELECT commands give a prefix to field names, and that the FROM command identifies that same prefix when specifying the table.

Join and Where

We have airline information and departure information, but now we need to put them together. The JOIN command does just this. We need to specify for the JOIN command how we're going to put these two tables together.

```
req1_text<-"SELECT a.carrier, a.name, f.dep_delay
            FROM flights f
            JOIN airlines a ON a.carrier=f.carrier
            "
req1<-dbSendQuery(con,req1_text)

req1_df<-dbFetch(req1,n=-1)

dbClearResult(req1)

head(req1_df)
```

```
##   carrier                   name dep_delay
## 1      UA  United Air Lines Inc.         2
## 2      UA  United Air Lines Inc.         4
## 3      AA American Airlines Inc.         2
## 4      B6        JetBlue Airways        -1
## 5      DL   Delta Air Lines Inc.        -6
## 6      UA  United Air Lines Inc.        -4
```

The resulting dataset has lots of short delays, those less than an hour. Let's say we're interested in finding out which airlines have more delays of more than an hour. We need to use the WHERE command to figure this out.

```
#New request: which carriers have longer delays?
req1_text<-"SELECT a.carrier, a.name, f.dep_delay
            FROM flights f
            JOIN airlines a ON a.carrier=f.carrier
            WHERE f.dep_delay>60
            "
req1<-dbSendQuery(con,req1_text)

req1_df<-dbFetch(req1,n=-1)

dbClearResult(req1)

head(req1_df)
```

```
##   carrier                     name dep_delay
## 1      MQ                Envoy Air       101
## 2      AA   American Airlines Inc.        71
## 3      MQ                Envoy Air       853
## 4      UA    United Air Lines Inc.       144
## 5      UA    United Air Lines Inc.       134
## 6      EV ExpressJet Airlines Inc.        96
```

```
table(req1_df$name)
```

```
## 
## AirTran Airways Corporation        Alaska Airlines Inc. 
##                         314                          39 
##      American Airlines Inc.        Delta Air Lines Inc. 
##                        2003                        2651 
##           Endeavor Air Inc.                   Envoy Air 
##                        1966                        1996 
##    ExpressJet Airlines Inc.      Frontier Airlines Inc. 
##                        6861                          73 
##      Hawaiian Airlines Inc.             JetBlue Airways 
##                          10                        4571 
##          Mesa Airlines Inc.       SkyWest Airlines Inc. 
##                          79                           4 
##      Southwest Airlines Co.       United Air Lines Inc. 
##                        1061                        3824 
##             US Airways Inc.              Virgin America 
##                         766                         363
```

```
delay_summary<-req1_df%>%
  group_by(name)%>%
  summarize(avg_delay=mean(dep_delay,na.rm=TRUE))%>%
  arrange(-avg_delay)

delay_summary
```

```
## # A tibble: 16 x 2
##    name                        avg_delay
##    <chr>                           <dbl>
##  1 Hawaiian Airlines Inc.           243.
##  2 Virgin America                   146.
##  3 AirTran Airways Corporation      145.
##  4 Frontier Airlines Inc.           145.
##  5 Delta Air Lines Inc.             136.
##  6 Southwest Airlines Co.           132.
##  7 Endeavor Air Inc.                124.
##  8 American Airlines Inc.           123.
##  9 United Air Lines Inc.            121.
## 10 Alaska Airlines Inc.             118.
## 11 ExpressJet Airlines Inc.         118.
## 12 JetBlue Airways                  118.
## 13 Mesa Airlines Inc.               117.
## 14 US Airways Inc.                  116.
## 15 Envoy Air                        115.
## 16 SkyWest Airlines Inc.            109.
```

In this year, if flights were to be delayed by more than an hour, Hawaiian Airlines had the longest subsequent delays.

```
#New request: which carriers have the most long-distance delays?
req1_text<-"SELECT a.carrier, a.name, f.distance, f.dep_delay
            FROM flights f
            JOIN airlines a ON a.carrier=f.carrier
            WHERE f.distance>1000
            "
req1<-dbSendQuery(con,req1_text)

req1_df<-dbFetch(req1,n=-1)

dbClearResult(req1)

head(req1_df)
```

```
##   carrier                   name distance dep_delay
## 1      UA  United Air Lines Inc.     1400         2
## 2      UA  United Air Lines Inc.     1416         4
## 3      AA American Airlines Inc.     1089         2
## 4      B6        JetBlue Airways     1576        -1
## 5      B6        JetBlue Airways     1065        -5
## 6      B6        JetBlue Airways     1028        -2
```

```
table(req1_df$name)
```

```
##
##     Alaska Airlines Inc.   American Airlines Inc.     Delta Air Lines Inc.
##                      714                    23583                    28096
##        Endeavor Air Inc.                Envoy Air ExpressJet Airlines Inc.
##                     2720                     2291                     6248
##   Frontier Airlines Inc.   Hawaiian Airlines Inc.          JetBlue Airways
##                      685                      342                    30022
##    SkyWest Airlines Inc.   Southwest Airlines Co.    United Air Lines Inc.
##                        4                     3832                    41135
##          US Airways Inc.           Virgin America
##                     2271                     5162
```

```
dist_flights<-req1_df%>%
  group_by(name)%>%
  tally()%>%
  arrange(-n)

dist_flights
```

```
## # A tibble: 14 x 2
##    name                         n
##    <chr>                    <int>
##  1 United Air Lines Inc.    41135
##  2 JetBlue Airways          30022
##  3 Delta Air Lines Inc.     28096
##  4 American Airlines Inc.   23583
##  5 ExpressJet Airlines Inc.  6248
##  6 Virgin America            5162
##  7 Southwest Airlines Co.    3832
##  8 Endeavor Air Inc.         2720
##  9 Envoy Air                 2291
## 10 US Airways Inc.           2271
## 11 Alaska Airlines Inc.       714
## 12 Frontier Airlines Inc.     685
## 13 Hawaiian Airlines Inc.     342
## 14 SkyWest Airlines Inc.        4
```

```
delay_summary<-req1_df%>%
  group_by(name)%>%
  summarize(avg_delay=mean(dep_delay,na.rm=TRUE))%>%
  arrange(-avg_delay)
```

We can combine multiple tables to answer questions about how multiple factors, like weather and carrier, might be related.

```
#SQL request, flight info combined with weather info
weather_text<-"SELECT f.year, f.month, f.day, f.distance, f.dep_delay, w.visib, w.wind_speed, w.wind_gus
              FROM weather w
              JOIN flights f
              JOIN airlines a
              ON f.year=w.year
              AND f.month=w.month
              AND f.day=w.day
              AND f.carrier=a.carrier
              WHERE f.dep_delay>20 AND w.wind_gust<100"

weather_req<-dbSendQuery(con,weather_text)

weather_df<-dbFetch(weather_req,n=-1)

dbClearResult(weather_req)

head(weather_df)
```

```
##   year month day distance dep_delay visib wind_speed wind_gust
## 1 2013     1   1     1372        21    10   13.80936  20.71404
## 2 2013     1   1      264        21    10   13.80936  20.71404
## 3 2013     1   1      301        21    10   13.80936  20.71404
## 4 2013     1   1     2475        21    10   13.80936  20.71404
## 5 2013     1   1      266        21    10   13.80936  20.71404
## 6 2013     1   1      282        21    10   13.80936  20.71404
##                      name
## 1   American Airlines Inc.
## 2          JetBlue Airways
## 3          JetBlue Airways
## 4     Delta Air Lines Inc.
## 5 ExpressJet Airlines Inc.
## 6 ExpressJet Airlines Inc.
```

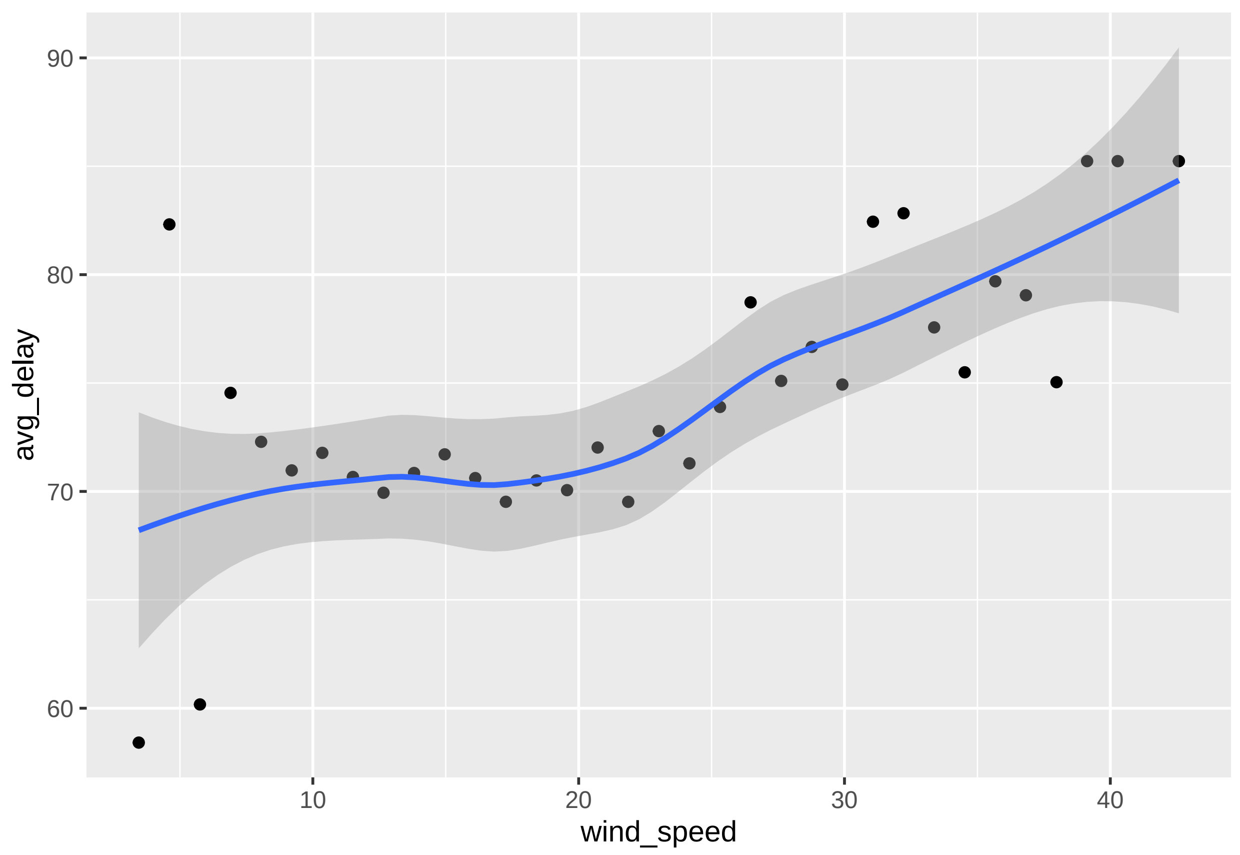
```
weather_summary<-
  weather_df%>%
  group_by(name,wind_gust)%>%
  summarize(avg_delay=mean(dep_delay,na.rm=TRUE))



weather_summary_2<-
  weather_df%>%
  group_by(wind_speed)%>%
  summarize(avg_delay=mean(dep_delay,na.rm=TRUE))


weather_summary
```

```
## # A tibble: 554 x 3
## # Groups:   name [16]
##    name                        wind_gust avg_delay
##    <chr>                           <dbl>     <dbl>
##  1 AirTran Airways Corporation      16.1      90.1
##  2 AirTran Airways Corporation      17.3      72.6
##  3 AirTran Airways Corporation      18.4      87.9
##  4 AirTran Airways Corporation      19.6      69.9
##  5 AirTran Airways Corporation      20.7      87.9
##  6 AirTran Airways Corporation      21.9      77.2
##  7 AirTran Airways Corporation      23.0      78.8
##  8 AirTran Airways Corporation      24.2      74.3
##  9 AirTran Airways Corporation      25.3      72.6
## 10 AirTran Airways Corporation      26.5      79.5
## # ... with 544 more rows
```

As always, this data can then be plotted to view trends.

```
#Plot average delay by wind gust
g1<-ggplot(data=weather_summary_2,aes(x=wind_speed,y=avg_delay))
g1<-g1+geom_point()
g1<-g1+geom_smooth()
g1
```

```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```



Quick Exercise Plot average delay by wind speed.

It's good practice to disconnect from a database when done.

```
## Disconnect when done
dbDisconnect(con)
```

MY ADVICE

Unless you're going to be a database analyst, do a bare minimum in the DBMS. Instead, use the DBMS to generate tables of manageable size for your computer, then get to work with R. Also, buy low and sell high.