Data Preparation

Dan Lizotte

2018-09-18

Data Wrangling and Exploration in R

Reasons to use R

- Easy-ish input from relational database, `.csv`, `.xlsx`
- Nice syntax for data table manipulation
- Elegant plotting
- There's a package for everything. (The python of the statistics world.)
- On the other hand, python is also great. **You are *not* required to use R in this course.**
- Resource for tidying/wrangling: [https://www.rstudio.com/wp-content/uploads/2015/02/data-wrangling-cheatsheet.pdf]

New York City Flights 2013

```
# IF you see library(blah) in my code, you will need to install.packages("blah") before running it.
library(nycflights13)
print(flights)
```

```
## # A tibble: 336,776 x 19
##     year month   day dep_time sched_dep_time dep_delay arr_time
##    <int> <int> <int>    <int>          <int>     <dbl>    <int>
##  1  2013     1     1      517            515         2      830
##  2  2013     1     1      533            529         4      850
##  3  2013     1     1      542            540         2      923
##  4  2013     1     1      544            545        -1     1004
##  5  2013     1     1      554            600        -6      812
##  6  2013     1     1      554            558        -4      740
##  7  2013     1     1      555            600        -5      913
##  8  2013     1     1      557            600        -3      709
##  9  2013     1     1      557            600        -3      838
## 10  2013     1     1      558            600        -2      753
## # ... with 336,766 more rows, and 12 more variables: sched_arr_time <int>,
## #   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>
```

```
print(planes)
```

```
## # A tibble: 3,322 x 9
##    tailnum  year type      manufacturer  model  engines seats speed engine
##    <chr>   <int> <chr>     <chr>         <chr>    <int> <int> <int> <chr>
##  1 N10156   2004 Fixed wi~ EMBRAER       EMB-1~       2    55    NA Turbo~
##  2 N102UW   1998 Fixed wi~ AIRBUS INDUS~ A320-~       2   182    NA Turbo~
##  3 N103US   1999 Fixed wi~ AIRBUS INDUS~ A320-~       2   182    NA Turbo~
##  4 N104UW   1999 Fixed wi~ AIRBUS INDUS~ A320-~       2   182    NA Turbo~
##  5 N10575   2002 Fixed wi~ EMBRAER       EMB-1~       2    55    NA Turbo~
##  6 N105UW   1999 Fixed wi~ AIRBUS INDUS~ A320-~       2   182    NA Turbo~
##  7 N107US   1999 Fixed wi~ AIRBUS INDUS~ A320-~       2   182    NA Turbo~
##  8 N108UW   1999 Fixed wi~ AIRBUS INDUS~ A320-~       2   182    NA Turbo~
##  9 N109UW   1999 Fixed wi~ AIRBUS INDUS~ A320-~       2   182    NA Turbo~
## 10 N110UW   1999 Fixed wi~ AIRBUS INDUS~ A320-~       2   182    NA Turbo~
## # ... with 3,312 more rows
```

```
print(weather)
```

```
## # A tibble: 26,115 x 15
##    origin  year month   day  hour  temp  dewp humid wind_dir wind_speed
##    <chr>  <dbl> <dbl> <int> <int> <dbl> <dbl> <dbl>    <dbl>      <dbl>
##  1 EWR     2013     1     1     1  39.0  26.1  59.4      270      10.4
##  2 EWR     2013     1     1     2  39.0  27.0  61.6      250       8.06
##  3 EWR     2013     1     1     3  39.0  28.0  64.4      240      11.5
##  4 EWR     2013     1     1     4  39.9  28.0  62.2      250      12.7
##  5 EWR     2013     1     1     5  39.0  28.0  64.4      260      12.7
##  6 EWR     2013     1     1     6  37.9  28.0  67.2      240      11.5
##  7 EWR     2013     1     1     7  39.0  28.0  64.4      240      15.0
##  8 EWR     2013     1     1     8  39.9  28.0  62.2      250      10.4
##  9 EWR     2013     1     1     9  39.9  28.0  62.2      260      15.0
## 10 EWR     2013     1     1    10  41    28.0  59.6      260      13.8
## # ... with 26,105 more rows, and 5 more variables: wind_gust <dbl>,
## #   precip <dbl>, pressure <dbl>, visib <dbl>, time_hour <dttm>
```

```
print(airlines)
```

```
## # A tibble: 16 x 2
##    carrier name
##    <chr>   <chr>
##  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.
```

```r
print(airports)
```

```
## # A tibble: 1,458 x 8
##    faa   name                   lat    lon   alt    tz dst   tzone
##    <chr> <chr>                <dbl>  <dbl> <int> <dbl> <chr> <chr>
##  1 04G   Lansdowne Airport     41.1  -80.6  1044    -5 A     America/New_~
##  2 06A   Moton Field Municip~  32.5  -85.7   264    -6 A     America/Chic~
##  3 06C   Schaumburg Regional   42.0  -88.1   801    -6 A     America/Chic~
##  4 06N   Randall Airport       41.4  -74.4   523    -5 A     America/New_~
##  5 09J   Jekyll Island Airpo~  31.1  -81.4    11    -5 A     America/New_~
##  6 0A9   Elizabethton Munici~  36.4  -82.2  1593    -5 A     America/New_~
##  7 0G6   Williams County Air~  41.5  -84.5   730    -5 A     America/New_~
##  8 0G7   Finger Lakes Region~  42.9  -76.8   492    -5 A     America/New_~
##  9 0P2   Shoestring Aviation~  39.8  -76.6  1000    -5 U     America/New_~
## 10 0S9   Jefferson County In~  48.1 -123.    108    -8 A     America/Los_~
## # ... with 1,448 more rows
```

`dplyr`

The `dplyr` package in R provides simple functions that correspond to the most common data manipulation operations (or *verbs*) and uses efficient storage approaches.

https://www.rstudio.com/wp-content/uploads/2015/02/data-wrangling-cheatsheet.pdf

```
verb           meaning
--------------------------------------------
select()       select variables (or columns)
filter()       subset observations (or rows)
mutate()       add new variables (or columns)
arrange()      re-order the observations
summarise()    reduce to a single row
group_by()     aggregate
left_join()    merge two data objects
distinct()     remove duplicate entries
collect()      force computation and bring data back into R
```

Designed for tidy data

- `dplyr` was designed with the idea of tidy data in mind, but can be applied to all data
- good for coercing data into a new format
- convenient syntax:

```
newdata <- olddata %>% verb1(options) %>% verb2(options) %>% verb3(options)
```

Filtering observations

- `filter()` extracts a subset of rows of interest
- Suppose we wanted to find out which airports certain codes belong to?

```
library(dplyr)
airports %>% filter(faa %in% c('ALB', 'BDL', 'BTV'))
```

```
## # A tibble: 3 x 8
##   faa   name               lat   lon   alt    tz dst   tzone
##   <chr> <chr>            <dbl> <dbl> <int> <dbl> <chr> <chr>
## 1 ALB   Albany Intl       42.7 -73.8   285    -5 A     America/New_York
## 2 BDL   Bradley Intl      41.9 -72.7   173    -5 A     America/New_York
## 3 BTV   Burlington Intl   44.5 -73.2   335    -5 A     America/New_York
```

Grouping

```
flights
```

```
## # A tibble: 336,776 x 19
##     year month   day dep_time sched_dep_time dep_delay arr_time
##    <int> <int> <int>    <int>          <int>     <dbl>    <int>
##  1  2013     1     1      517            515         2      830
##  2  2013     1     1      533            529         4      850
##  3  2013     1     1      542            540         2      923
##  4  2013     1     1      544            545        -1     1004
##  5  2013     1     1      554            600        -6      812
##  6  2013     1     1      554            558        -4      740
##  7  2013     1     1      555            600        -5      913
##  8  2013     1     1      557            600        -3      709
##  9  2013     1     1      557            600        -3      838
## 10  2013     1     1      558            600        -2      753
## # ... with 336,766 more rows, and 12 more variables: sched_arr_time <int>,
## #   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>
```

```
bycarrier <- flights %>% group_by(carrier)
bycarrier
```

```
## # A tibble: 336,776 x 19
## # Groups:   carrier [16]
##     year month   day dep_time sched_dep_time dep_delay arr_time
##    <int> <int> <int>    <int>          <int>     <dbl>    <int>
##  1  2013     1     1      517            515         2      830
##  2  2013     1     1      533            529         4      850
##  3  2013     1     1      542            540         2      923
##  4  2013     1     1      544            545        -1     1004
##  5  2013     1     1      554            600        -6      812
##  6  2013     1     1      554            558        -4      740
##  7  2013     1     1      555            600        -5      913
##  8  2013     1     1      557            600        -3      709
##  9  2013     1     1      557            600        -3      838
## 10  2013     1     1      558            600        -2      753
```

```
## # ... with 336,766 more rows, and 12 more variables: sched_arr_time <int>,
## #   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>
```

Grouping and Summaries

```r
flights %>% summarise(numflights = n()) #n() counts rows
```

```
## # A tibble: 1 x 1
##   numflights
##        <int>
## 1     336776
```

```r
bycarrier %>% summarise(numflights = n()) #n() counts rows
```

```
## # A tibble: 16 x 2
##    carrier numflights
##    <chr>        <int>
##  1 9E           18460
##  2 AA           32729
##  3 AS             714
##  4 B6           54635
##  5 DL           48110
##  6 EV           54173
##  7 F9             685
##  8 FL            3260
##  9 HA             342
## 10 MQ           26397
## 11 OO              32
## 12 UA           58665
## 13 US           20536
## 14 VX            5162
## 15 WN           12275
## 16 YV             601
```

Aggregating observations

Aggregate the counts of flights at the monthly level.

```r
monthlycounts <- flights %>%
   filter(dest %in% c('ALB', 'BDL', 'BTV')) %>%
   group_by(year, month) %>%
   summarise(numflights = n())
monthlycounts
```

```
## # A tibble: 12 x 3
## # Groups:   year [?]
##     year month numflights
##    <int> <int>      <int>
##  1  2013     1        324
##  2  2013     2        293
##  3  2013     3        376
```

```
## 4  2013     4         297
## 5  2013     5         372
## 6  2013     6         346
## 7  2013     7         253
## 8  2013     8         284
## 9  2013     9         204
## 10  2013    10         248
## 11  2013    11         214
## 12  2013    12         260
```

Aggregating observations

Aggregate the counts of flights at three airports at the monthly level.

```
airportmonthlycounts <- flights %>%
   filter(dest %in% c('ALB', 'BDL', 'BTV')) %>%
   group_by(year, month, dest) %>%
   summarise(numflights = n())
airportmonthlycounts
```

```
## # A tibble: 36 x 4
## # Groups:   year, month [?]
##     year month dest  numflights
##    <int> <int> <chr>      <int>
##  1  2013     1 ALB           64
##  2  2013     1 BDL           37
##  3  2013     1 BTV          223
##  4  2013     2 ALB           58
##  5  2013     2 BDL           46
##  6  2013     2 BTV          189
##  7  2013     3 ALB           57
##  8  2013     3 BDL           62
##  9  2013     3 BTV          257
## 10  2013     4 ALB           13
## # ... with 26 more rows
```

Creating new derived variables

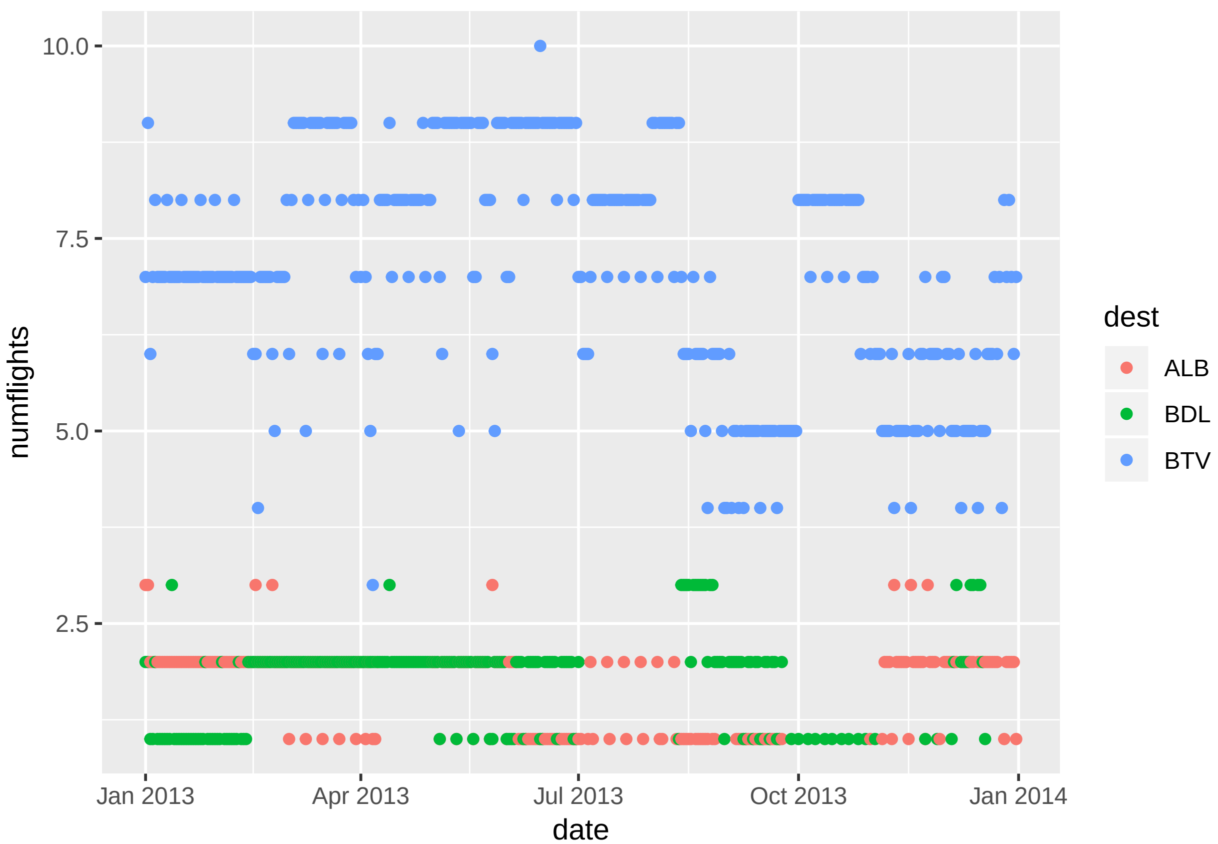
Add a new column by constructing a date variable using `mutate()`. R has a special "date" data type that is useful; dates can be constructed different ways, including using the `ymd()` function.

```
library(lubridate) #To get the ymd() function
airportdailycounts <- flights %>%
   filter(dest %in% c('ALB', 'BDL', 'BTV')) %>% group_by(year, month, day, dest) %>%
   summarise(numflights = n()) %>% mutate(date = ymd(paste(year, month, day, sep = "-")))
airportdailycounts
```

```
## # A tibble: 876 x 6
## # Groups:   year, month, day [365]
##     year month   day dest  numflights date
##    <int> <int> <int> <chr>      <int> <date>
##  1  2013     1     1 ALB            3 2013-01-01
##  2  2013     1     1 BDL            2 2013-01-01
##  3  2013     1     1 BTV            7 2013-01-01
```

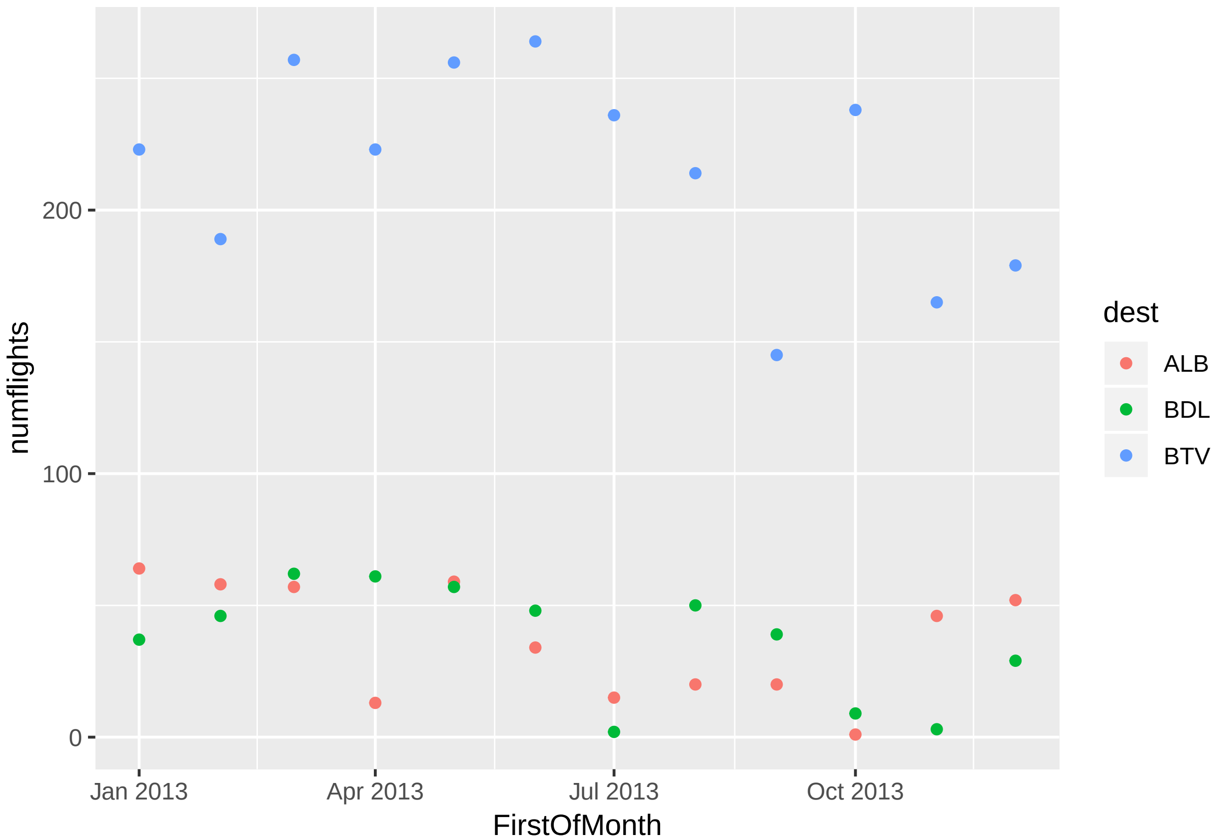
```
## 4 2013 1 2 ALB 3 2013-01-02
## 5 2013 1 2 BDL 2 2013-01-02
## 6 2013 1 2 BTV 9 2013-01-02
## 7 2013 1 3 ALB 2 2013-01-03
## 8 2013 1 3 BDL 1 2013-01-03
## 9 2013 1 3 BTV 6 2013-01-03
## 10 2013 1 4 ALB 2 2013-01-04
## # ... with 866 more rows
```

```
library(ggplot2)
ggplot(data = airportdailycounts, aes(x = date, y = numflights, colour = dest)) +
  geom_point()
```



Plot by month instead

```
airportmonthlycounts <- airportmonthlycounts %>%
    mutate(FirstOfMonth = ymd(paste(year, "-", month, "-01", sep="")))
ggplot(data = airportmonthlycounts, aes(x = FirstOfMonth, y = numflights, colour = dest)) +
  geom_point()
```



Sorting and selecting

`arrange()` lets us display the months with the largest number of flights.

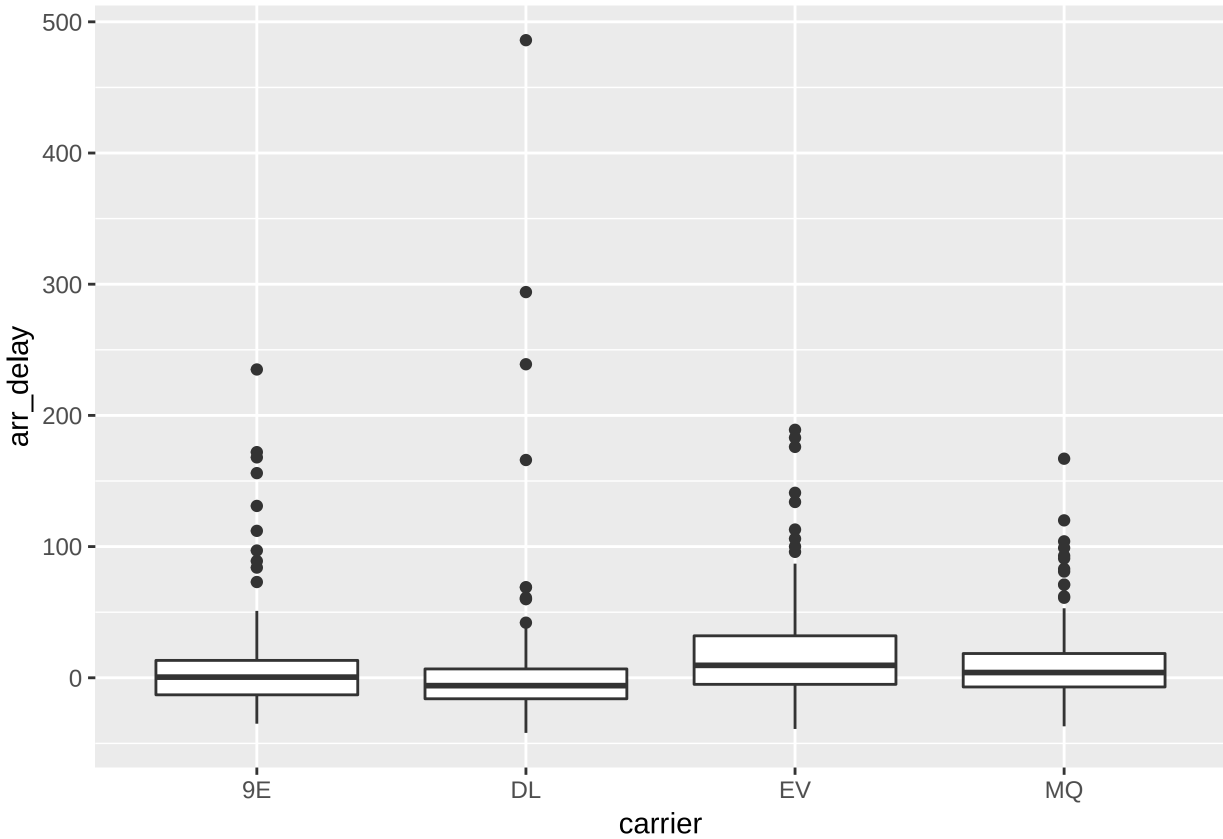
```
airportmonthlycounts %>% arrange(desc(numflights))
```

```
## # A tibble: 36 x 5
## # Groups:   year, month [12]
##     year month dest  numflights FirstOfMonth
##    <int> <int> <chr>      <int> <date>
##  1  2013     6 BTV          264 2013-06-01
##  2  2013     3 BTV          257 2013-03-01
##  3  2013     5 BTV          256 2013-05-01
##  4  2013    10 BTV          238 2013-10-01
##  5  2013     7 BTV          236 2013-07-01
##  6  2013     1 BTV          223 2013-01-01
##  7  2013     4 BTV          223 2013-04-01
##  8  2013     8 BTV          214 2013-08-01
##  9  2013     2 BTV          189 2013-02-01
## 10  2013    12 BTV          179 2013-12-01
## # ... with 26 more rows
```

Comparing airlines

Which airline was most reliable flying from New York to Minneapolis/St. Paul (MSP) in January, 2013?

```r
jandelays <- flights %>% select(origin, dest, year, month, day, carrier, arr_delay) %>%
  filter(dest == 'MSP' & month == 1)
ggplot(data = jandelays, aes(x = carrier, y = arr_delay)) + geom_boxplot()
```



Merging or "Joining"

Here, the full carrier names are merged (or joined, in database speak) using the `left_join()`

```r
merged <- left_join(jandelays, airlines, by = c("carrier" = "carrier"))
merged
```

```
## # A tibble: 546 x 8
##    origin dest   year month   day carrier arr_delay name
##    <chr>  <chr> <int> <int> <int> <chr>       <dbl> <chr>
##  1 LGA    MSP    2013     1     1 DL             -8 Delta Air Lines Inc.
##  2 EWR    MSP    2013     1     1 EV             29 ExpressJet Airlines I~
##  3 LGA    MSP    2013     1     1 MQ             10 Envoy Air
##  4 LGA    MSP    2013     1     1 DL              8 Delta Air Lines Inc.
##  5 JFK    MSP    2013     1     1 9E             11 Endeavor Air Inc.
##  6 LGA    MSP    2013     1     1 DL            -10 Delta Air Lines Inc.
##  7 LGA    MSP    2013     1     1 DL              3 Delta Air Lines Inc.
##  8 LGA    MSP    2013     1     1 MQ             93 Envoy Air
##  9 LGA    MSP    2013     1     1 DL             -8 Delta Air Lines Inc.
## 10 LGA    MSP    2013     1     1 MQ             91 Envoy Air
## # ... with 536 more rows
```

Truth in Advertising

`?flights` gives description: "On-time data for **all** flights that departed NYC (i.e. JFK, LGA or EWR) in 2013."

```r
flights %>% filter(dest == 'ORD') %>% summarize(count = n())
```

```
## # A tibble: 1 x 1
##   count
##   <int>
## 1 17283
```

```r
flights %>% filter(dest == 'YYZ') %>% summarize(count = n())
```

```
## # A tibble: 1 x 1
##   count
##   <int>
## 1     0
```

Big Databases

`nycflights13` is just a fraction of the available flight information. See http://www.amherst.edu/~nhorton/precursors for example code using SQLite.

Relational databases, first popularized in the 1970's, provide fast and efficient access to terabyte-sized files. These systems use a structured query language (SQL) to specify data operations.

Database systems have been highly optimized and tuned since they were first invented. Connections between general purpose statistics packages such as R and database systems can be facilitated through use of SQL.

Key operators in SQL

```
verb          meaning
--------------------------------------------
SELECT      create a new result set from a table
FROM        specify table
WHERE       subset observations
GROUP BY    aggregate
ORDER       re-order the observations
DISTINCT    remove duplicate values
JOIN        merge two data objects
```

Data Cleaning

Data cleansing, data cleaning, or data scrubbing is the process of **detecting and correcting (or removing) corrupt or inaccurate records** from a record set, table, or database and refers to **identifying incomplete, incorrect, inaccurate or irrelevant parts of the data** and then **replacing, modifying, or deleting the dirty or coarse data.**

(Wikipedia)

Tools

- Your scripting/programming language of choice. (Will look at R today.)
  - Ensures reproducibility.
  - R cheat sheets under Other Resources on the course wiki
- OpenRefine http://openrefine.org
  - Formerly Google Refine. Open source.
- Tableau http://www.tableau.com
  - Industrial-strength. Free student license available.