R for Data science

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

```
## -- Attaching packages --------------------------------- tidyverse 1.3.0.9000 --

## v ggplot2 3.3.0     v purrr   0.3.3
## v tibble  3.0.1     v dplyr   0.8.5
## v tidyr   1.0.2     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.5.0

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

```r
str(flights)
```

```
## tibble [336,776 x 19] (S3: tbl_df/tbl/data.frame)
##  $ year          : int [1:336776] 2013 2013 2013 2013 2013 2013 2013 2013 2013 2013 ...
##  $ month         : int [1:336776] 1 1 1 1 1 1 1 1 1 1 ...
##  $ day           : int [1:336776] 1 1 1 1 1 1 1 1 1 1 ...
##  $ dep_time      : int [1:336776] 517 533 542 544 554 554 555 557 557 558 ...
##  $ sched_dep_time: int [1:336776] 515 529 540 545 600 558 600 600 600 600 ...
##  $ dep_delay     : num [1:336776] 2 4 2 -1 -6 -4 -5 -3 -3 -2 ...
##  $ arr_time      : int [1:336776] 830 850 923 1004 812 740 913 709 838 753 ...
##  $ sched_arr_time: int [1:336776] 819 830 850 1022 837 728 854 723 846 745 ...
##  $ arr_delay     : num [1:336776] 11 20 33 -18 -25 12 19 -14 -8 8 ...
##  $ carrier       : chr [1:336776] "UA" "UA" "AA" "B6" ...
##  $ flight        : int [1:336776] 1545 1714 1141 725 461 1696 507 5708 79 301 ...
##  $ tailnum       : chr [1:336776] "N14228" "N24211" "N619AA" "N804JB" ...
##  $ origin        : chr [1:336776] "EWR" "LGA" "JFK" "JFK" ...
##  $ dest          : chr [1:336776] "IAH" "IAH" "MIA" "BQN" ...
##  $ air_time      : num [1:336776] 227 227 160 183 116 150 158 53 140 138 ...
##  $ distance      : num [1:336776] 1400 1416 1089 1576 762 ...
##  $ hour          : num [1:336776] 5 5 5 5 6 5 6 6 6 6 ...
##  $ minute        : num [1:336776] 15 29 40 45 0 58 0 0 0 0 ...
##  $ time_hour     : POSIXct[1:336776], format: "2013-01-01 05:00:00" "2013-01-01 05:00:00" ...
```

```r
# filter() are combined with "and": every expression must be true in order
# for a row to be included in the output

(jan1 <- filter(flights, month==1 , day == 1))
```

```
## # 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>
```

```
# use near for double value equalities
near (sqrt(2) ^ 2 , 2)
```

```
## [1] TRUE
```

```
(jan1 <- filter(flights, month==11 | month == 12))
```

```
## # A tibble: 55,403 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    11     1        5           2359         6      352            345
##  2  2013    11     1       35           2250       105      123           2356
##  3  2013    11     1      455            500        -5      641            651
##  4  2013    11     1      539            545        -6      856            827
##  5  2013    11     1      542            545        -3      831            855
##  6  2013    11     1      549            600       -11      912            923
##  7  2013    11     1      550            600       -10      705            659
##  8  2013    11     1      554            600        -6      659            701
##  9  2013    11     1      554            600        -6      826            827
## 10  2013    11     1      554            600        -6      749            751
## # ... with 55,393 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>
```

```
# equivalently
jan1 <- filter(flights, month %in% c(11,12))

# filter excludes both FALSE and NA values
df <- tibble(x = c(1, NA, 3))
filter(df, x>1)
```

```
## # A tibble: 1 x 1
##       x
##   <dbl>
## 1     3
```

```
filter(df, is.na(x) | x > 1)
```

```
## # A tibble: 2 x 1
##       x
##   <dbl>
## 1    NA
## 2     3
```

```
# Exercise 5.2.4
#Find all flights that:
  # Had an arrival delay of two or more hours
filter(flights, arr_delay >= 2)
```

```
## # A tibble: 127,929 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      554            558        -4      740            728
##  5  2013     1     1      555            600        -5      913            854
##  6  2013     1     1      558            600        -2      753            745
##  7  2013     1     1      558            600        -2      924            917
##  8  2013     1     1      559            600        -1      941            910
##  9  2013     1     1      600            600         0      837            825
## 10  2013     1     1      602            605        -3      821            805
## # ... with 127,919 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>
```

```r
  #Flew to Houston (IAH or HOU)
filter(flights, dest %in% c("IAH", "HOU"))
```

```
## # A tibble: 9,313 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      623            627        -4      933            932
##  4  2013     1     1      728            732        -4     1041           1038
##  5  2013     1     1      739            739         0     1104           1038
##  6  2013     1     1      908            908         0     1228           1219
##  7  2013     1     1     1028           1026         2     1350           1339
##  8  2013     1     1     1044           1045        -1     1352           1351
##  9  2013     1     1     1114            900       134     1447           1222
## 10  2013     1     1     1205           1200         5     1503           1505
## # ... with 9,303 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>
```

```r
  #Were operated by United, American, or Delta
filter(flights, carrier %in% c("UA", "AM", "DEL"))
```

```
## # A tibble: 58,665 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      554            558        -4      740            728
##  4  2013     1     1      558            600        -2      924            917
##  5  2013     1     1      558            600        -2      923            937
##  6  2013     1     1      559            600        -1      854            902
##  7  2013     1     1      607            607         0      858            915
##  8  2013     1     1      611            600        11      945            931
##  9  2013     1     1      623            627        -4      933            932
## 10  2013     1     1      628            630        -2     1016            947
## # ... with 58,655 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>
```

```
  # Departed in summer (July, August, and September)
filter(flights, month %in% c(7, 8, 9))
```

```
## # A tibble: 86,326 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     7     1        1           2029       212      236           2359
##  2  2013     7     1        2           2359         3      344            344
##  3  2013     7     1       29           2245       104      151              1
##  4  2013     7     1       43           2130       193      322             14
##  5  2013     7     1       44           2150       174      300            100
##  6  2013     7     1       46           2051       235      304           2358
##  7  2013     7     1       48           2001       287      308           2305
##  8  2013     7     1       58           2155       183      335             43
##  9  2013     7     1      100           2146       194      327             30
## 10  2013     7     1      100           2245       135      337            135
## # ... with 86,316 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>
```

```
  # Arrived more than two hours late, but didn’t leave late
filter(flights, dep_delay <= 0 & arr_delay >= 120)
```

```
## # A tibble: 29 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    27     1419           1420        -1     1754           1550
##  2  2013    10     7     1350           1350         0     1736           1526
##  3  2013    10     7     1357           1359        -2     1858           1654
##  4  2013    10    16      657            700        -3     1258           1056
##  5  2013    11     1      658            700        -2     1329           1015
##  6  2013     3    18     1844           1847        -3       39           2219
##  7  2013     4    17     1635           1640        -5     2049           1845
##  8  2013     4    18      558            600        -2     1149            850
##  9  2013     4    18      655            700        -5     1213            950
## 10  2013     5    22     1827           1830        -3     2217           2010
## # ... with 19 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>
```

```
flights[between(flights$dep_delay,0, 120), ]
```

```
## # A tibble: 143,478 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      559            559         0      702            706
##  5  2013     1     1      600            600         0      851            858
##  6  2013     1     1      600            600         0      837            825
##  7  2013     1     1      601            600         1      844            850
##  8  2013     1     1      607            607         0      858            915
##  9  2013     1     1      608            600         8      807            735
## 10  2013     1     1      611            600        11      945            931
## # ... with 143,468 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>
```

```
  # Were delayed by at least an hour, but made up over 30 minutes in flight
filter(flights, dep_delay > 60 & arr_delay <= 30)
```

```
## # A tibble: 211 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     3     1850           1745        65     2148           2120
##  2  2013     1     3     1950           1845        65     2228           2227
##  3  2013     1     6     1019            900        79     1558           1530
##  4  2013     1     7     1543           1430        73     1758           1735
##  5  2013     1    12     1706           1600        66     1949           1927
##  6  2013     1    12     1953           1845        68     2154           2137
##  7  2013     1    19     1456           1355        61     1636           1615
##  8  2013     1    21     1531           1430        61     1843           1815
##  9  2013     1    21     1648           1545        63     1939           1910
## 10  2013    10     5     1605           1500        65     1857           1827
## # ... with 201 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>
```

```
  # Departed between midnight and 6am (inclusive)
filter(flights, dep_time <= 600)
```

```
## # A tibble: 9,344 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 9,334 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>
```

```
#How many flights have a missing dep_time
paste0(nrow(filter(flights, is.na(dep_time))), " flights have missing dep_time value")
```

```
## [1] "8255 flights have missing dep_time value"
```

```
# complete.cases gives TRUE when all values in a row are not NA
flights[!complete.cases(flights), ]
```

```
## # A tibble: 9,430 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     1525           1530        -5     1934           1805
## 2  2013     1     1     1528           1459        29     2002           1647
## 3  2013     1     1     1740           1745        -5     2158           2020
## 4  2013     1     1     1807           1738        29     2251           2103
## 5  2013     1     1     1939           1840        59       29           2151
## 6  2013     1     1     1952           1930        22     2358           2207
## 7  2013     1     1     2016           1930        46       NA           2220
## 8  2013     1     1       NA           1630        NA       NA           1815
## 9  2013     1     1       NA           1935        NA       NA           2240
## 10 2013     1     1       NA           1500        NA       NA           1825
## # ... with 9,420 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>
```

```
# columns that have NA
colnames(flights[!complete.cases(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"
```

```
?complete.cases

# Arrange rows by getting a set of column names (or more complicated expressions) to order by
arrange(flights, year, month, desc(day))
```

```
## # 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    31        1           2100       181      124           2225
##  2  2013     1    31        4           2359         5      455            444
##  3  2013     1    31        7           2359         8      453            437
##  4  2013     1    31       12           2250        82      132              7
##  5  2013     1    31       26           2154       152      328             50
##  6  2013     1    31       34           2159       155      135           2315
##  7  2013     1    31       37           2249       108      132           2357
##  8  2013     1    31       54           2250       124      152           2359
##  9  2013     1    31      453            500        -7      651            648
## 10  2013     1    31      522            525        -3      820            820
## # ... 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>
```

```
# Missing values are always sorted at the end:
df <- tibble(x = c(5,2,NA))
arrange(df, x)
```

```
## # A tibble: 3 x 1
##       x
##   <dbl>
## 1     2
## 2     5
## 3    NA
```

```r
# 5.3.1 Exercises

# Sort missing values in the start
arrange(df, desc(is.na(x)), x)
```

```
## # A tibble: 3 x 1
##       x
##   <dbl>
## 1    NA
## 2     2
## 3     5
```

```r
# Sort flights to find the most delayed flights.
arrange(flights, desc(dep_delay))[1,]
```

```
## # A tibble: 1 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
## # ... 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>
```

```r
# Find the flights that left earliest.
arrange(flights, dep_time)[1,]
```

```
## # A tibble: 1 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    13        1           2249        72      108           2357
## # ... 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>
```

```r
# Sort flights to find the fastest flights.
arrange(flights, distance/air_time)[1,]
```

```
## # A tibble: 1 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    28     1917           1825        52     2118           1935
## # ... 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>
```

```r
#  Which travelled the shortest?
arrange(flights, distance)[1,]
```

```
## # A tibble: 1 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     7    27       NA            106        NA       NA            245
## # ... 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>
```

```r
# Which flights travelled the farthest?
arrange(flights, desc(distance))[1,]
```

```
## # A tibble: 1 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      857            900        -3     1516           1530
## # ... 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>
```

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

select(flights, year, month, day)
```

```
## # 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
```

```r
# Select all columns between year and day (inclusive)
select(flights, year:day)
```

```
## # 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
```

```r
# Select all columns except those from year to day (inclusive)
select(flights, -(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>
```

```
# select all columns whose name contains string "ela"
select(flights, contains("ela"))
```

```
## # 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
```

```
# select all columns whose name ends with "_time"
select(flights, ends_with("_time"))
```

```
## # A tibble: 336,776 x 5
##    dep_time sched_dep_time arr_time sched_arr_time air_time
##       <int>          <int>    <int>          <int>    <dbl>
##  1      517            515      830            819      227
##  2      533            529      850            830      227
##  3      542            540      923            850      160
##  4      544            545     1004           1022      183
##  5      554            600      812            837      116
##  6      554            558      740            728      150
##  7      555            600      913            854      158
##  8      557            600      709            723       53
##  9      557            600      838            846      140
## 10      558            600      753            745      138
## # ... with 336,766 more rows
```

```
# rename a column
rename(flights, departure_time=dep_time, arrival_time = arr_time)
```

```
## # A tibble: 336,776 x 19
##     year month   day departure_time sched_dep_time dep_delay arrival_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>
```

```
#  handful of variables you'd like to move to the start of the data frame.
select(flights, dep_time, arr_time, sched_dep_time, sched_arr_time, everything())
```

```
## # A tibble: 336,776 x 19
##    dep_time arr_time sched_dep_time sched_arr_time  year month   day dep_delay
##       <int>    <int>          <int>          <int> <int> <int> <int>     <dbl>
##  1      517      830            515            819  2013     1     1         2
##  2      533      850            529            830  2013     1     1         4
##  3      542      923            540            850  2013     1     1         2
##  4      544     1004            545           1022  2013     1     1        -1
##  5      554      812            600            837  2013     1     1        -6
##  6      554      740            558            728  2013     1     1        -4
##  7      555      913            600            854  2013     1     1        -5
##  8      557      709            600            723  2013     1     1        -3
##  9      557      838            600            846  2013     1     1        -3
## 10      558      753            600            745  2013     1     1        -2
## # ... 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>
```

```
# 5.4.1 Exercises

# Brainstorm as many ways as possible to select dep_time, dep_delay, arr_time, and arr_delay from fligh

select(flights, dep_time, dep_delay, arr_time, arr_delay)
```

```
## # A tibble: 336,776 x 4
##    dep_time dep_delay arr_time arr_delay
##       <int>     <dbl>    <int>     <dbl>
##  1      517         2      830        11
##  2      533         4      850        20
##  3      542         2      923        33
##  4      544        -1     1004       -18
##  5      554        -6      812       -25
##  6      554        -4      740        12
##  7      555        -5      913        19
##  8      557        -3      709       -14
##  9      557        -3      838        -8
## 10      558        -2      753         8
## # ... with 336,766 more rows
```

```
select(flights, starts_with("dep_") | starts_with("arr_"))
```

```
## # A tibble: 336,776 x 4
##    dep_time dep_delay arr_time arr_delay
##       <int>     <dbl>    <int>     <dbl>
##  1      517         2      830        11
##  2      533         4      850        20
##  3      542         2      923        33
##  4      544        -1     1004       -18
##  5      554        -6      812       -25
##  6      554        -4      740        12
##  7      555        -5      913        19
##  8      557        -3      709       -14
##  9      557        -3      838        -8
## 10      558        -2      753         8
## # ... with 336,766 more rows
```

```
select(flights, starts_with("dep_")  & (ends_with("_delay") | ends_with("_time"))
       | starts_with("arr_")  & (ends_with("_delay") | ends_with("_time")))
```

```
## # A tibble: 336,776 x 4
##    dep_delay dep_time arr_delay arr_time
##        <dbl>    <int>     <dbl>    <int>
##  1         2      517        11      830
##  2         4      533        20      850
##  3         2      542        33      923
##  4        -1      544       -18     1004
##  5        -6      554       -25      812
##  6        -4      554        12      740
##  7        -5      555        19      913
##  8        -3      557       -14      709
##  9        -3      557        -8      838
## 10        -2      558         8      753
## # ... with 336,766 more rows
```

```
# What happens if you include the name of a variable multiple times in a select() call?
select(flights, dep_time, dep_time, dep_time, arr_delay)
```

```
## # A tibble: 336,776 x 2
##    dep_time arr_delay
##       <int>     <dbl>
##  1      517        11
##  2      533        20
##  3      542        33
##  4      544       -18
##  5      554       -25
##  6      554        12
##  7      555        19
##  8      557       -14
##  9      557        -8
## 10      558         8
## # ... with 336,766 more rows
```

```
# What does the one_of() function do? Why might it be helpful in conjunction with this vector?
cols <- c("dep_time","XXX","dep_delay","ZZZ","QQQ","arr_delay")
```

```r
select(flights, one_of(cols))
```

```
## Warning: Unknown columns: `XXX`, `ZZZ`, `QQQ`

## # A tibble: 336,776 x 3
##    dep_time dep_delay arr_delay
##       <int>     <dbl>     <dbl>
##  1      517         2        11
##  2      533         4        20
##  3      542         2        33
##  4      544        -1       -18
##  5      554        -6       -25
##  6      554        -4        12
##  7      555        -5        19
##  8      557        -3       -14
##  9      557        -3        -8
## 10      558        -2         8
## # ... with 336,766 more rows
```

```r
# case insensetivity is surprising
select(flights, contains("TIME", ignore.case = F))
```

```
## # A tibble: 336,776 x 0
```

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

# mutate() always adds new columns at the end of your dataset so we'll start by creating a narrower dat

flights_small <- select(flights, year:day, ends_with("delay"), distance, air_time)
# view(flights_small)

flights_small
```

```
## # A tibble: 336,776 x 7
##     year month   day dep_delay arr_delay distance air_time
##    <int> <int> <int>     <dbl>     <dbl>    <dbl>    <dbl>
##  1  2013     1     1         2        11     1400      227
##  2  2013     1     1         4        20     1416      227
##  3  2013     1     1         2        33     1089      160
##  4  2013     1     1        -1       -18     1576      183
##  5  2013     1     1        -6       -25      762      116
##  6  2013     1     1        -4        12      719      150
##  7  2013     1     1        -5        19     1065      158
##  8  2013     1     1        -3       -14      229       53
##  9  2013     1     1        -3        -8      944      140
## 10  2013     1     1        -2         8      733      138
## # ... with 336,766 more rows
```

```r
mutate(flights, gain = dep_delay - arr_delay, gain_per_hour = gain/hour,
       speed = distance/air_time *60)
```

```
## # A tibble: 336,776 x 22
##     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 14 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>, gain_per_hour <dbl>, speed <dbl>
```

```
# If you only want to keep the new variables, use transmute()
transmute(flights, gain = dep_delay - arr_delay, gain_per_hour = gain/hour,
       speed = distance/air_time *60)
```

```
## # A tibble: 336,776 x 3
##     gain gain_per_hour speed
##    <dbl>         <dbl> <dbl>
##  1    -9         -1.8    370.
##  2   -16         -3.2    374.
##  3   -31         -6.2    408.
##  4    17          3.4    517.
##  5    19          3.17   394.
##  6   -16         -3.2    288.
##  7   -24         -4      404.
##  8    11          1.83   259.
##  9     5          0.833  405.
## 10   -10         -1.67   319.
## # ... with 336,766 more rows
```

```
# function must be vectorised to be able to use in mutate

# Modular arithmetic: %/% (integer division) and %% (remainder) are vectorized
transmute(flights, hour = dep_time %/% 100, min = dep_time %% 100)
```

```
## # A tibble: 336,776 x 2
##     hour   min
##    <dbl> <dbl>
##  1     5    17
##  2     5    33
##  3     5    42
##  4     5    44
##  5     5    54
##  6     5    54
##  7     5    55
##  8     5    57
##  9     5    57
## 10     5    58
## # ... with 336,766 more rows
```

```
# log() functions are also vectorize
transmute(flights, gain = dep_delay - arr_delay, gain_per_hour = gain/hour, logOfGain = log2(gain_per_h
## Warning: NaNs produced
```

```
## # A tibble: 336,776 x 3
##     gain gain_per_hour logOfGain
##    <dbl>         <dbl>     <dbl>
##  1    -9        -1.8       NaN
##  2   -16        -3.2       NaN
##  3   -31        -6.2       NaN
##  4    17         3.4       1.77
##  5    19         3.17      1.66
##  6   -16        -3.2       NaN
##  7   -24        -4         NaN
##  8    11         1.83      0.874
##  9     5         0.833    -0.263
## 10   -10        -1.67      NaN
## # ... with 336,766 more rows
```

```
# lead() and lag() allows you to compute running differences (e.g. x - lag(x)) or
# find when values change (x != lag(x)).

(x <- 1:10)
```

```
##  [1]  1  2  3  4  5  6  7  8  9 10
```

```
lag(x)
```

```
##  [1] NA  1  2  3  4  5  6  7  8  9
```

```
lead(x)
```

```
##  [1]  2  3  4  5  6  7  8  9 10 NA
```

```
transmute(flights, dest, lag(dest), lead(dest))
```

```
## # A tibble: 336,776 x 3
##    dest  `lag(dest)` `lead(dest)`
##    <chr> <chr>       <chr>
##  1 IAH   <NA>        IAH
##  2 IAH   IAH         MIA
##  3 MIA   IAH         BQN
##  4 BQN   MIA         ATL
##  5 ATL   BQN         ORD
##  6 ORD   ATL         FLL
##  7 FLL   ORD         IAD
##  8 IAD   FLL         MCO
##  9 MCO   IAD         ORD
## 10 ORD   MCO         PBI
## # ... with 336,766 more rows
```

```
# Cumulative and rolling aggregates (i.e. a sum computed over a rolling window):
# R => cumsum(), cumprod(), cummin(), cummax();
(x <- 1:10)
```

```
##  [1]  1  2  3  4  5  6  7  8  9 10
```

```
cumsum(x)
```

```
##  [1]  1  3  6 10 15 21 28 36 45 55
```

```r
cumprod(x)
```

```
##  [1]       1       2       6      24     120     720    5040   40320  362880
## [10] 3628800
```

```r
cummin(x)
```

```
##  [1] 1 1 1 1 1 1 1 1 1 1
```

```r
cummax(x)
```

```
##  [1]  1  2  3  4  5  6  7  8  9 10
```

```r
# dplyr => cummean() for cumulative means
cummean(x)
```

```
##  [1] 1.0 1.5 2.0 2.5 3.0 3.5 4.0 4.5 5.0 5.5
```

```r
# For Rolling aggregates use RcppRoll package
x <- matrix(rnorm(100),nrow=50,ncol=2)
x
```

```
##              [,1]        [,2]
##  [1,] -0.29959524 -0.63147089
##  [2,] -0.32246643  1.38927160
##  [3,]  0.39175228 -0.70178125
##  [4,] -0.65047246 -0.29528673
##  [5,] -0.31543296  0.01803912
##  [6,] -1.74899113  1.20615087
##  [7,] -0.90550442 -0.18016485
##  [8,] -1.63085485 -0.79654724
##  [9,] -0.01186019  0.82201281
## [10,]  0.51400268 -0.43872827
## [11,]  0.32506658 -0.77086793
## [12,] -0.52904793  0.89440511
## [13,] -2.08637920 -0.18307730
## [14,] -0.49910263  0.57321819
## [15,] -0.71880337  0.58602363
## [16,]  0.33800372  0.89548646
## [17,] -1.87720006 -1.82684575
## [18,]  1.01635860  1.58421898
## [19,]  0.25625602  0.98569845
## [20,] -0.42996769  1.04024702
## [21,]  2.37215561  0.61679673
## [22,] -0.77185834  0.21951974
## [23,]  1.95081160  1.29609260
## [24,] -1.41486967 -0.06892722
## [25,]  0.20850842  0.77680706
## [26,]  0.70111753 -1.29330171
## [27,] -0.59051599  1.17481428
## [28,] -1.17618094 -0.25125114
## [29,] -0.83247354 -0.76162277
## [30,] -1.20257030  0.79582482
## [31,] -1.97827664 -0.33878024
## [32,] -0.56421367 -1.06955657
## [33,]  1.17510767 -0.19830257
## [34,]  0.75657449 -0.62932429
## [35,]  0.88684945 -1.06833687
## [36,] -0.18783132  0.84877298
## [37,]  0.61024189 -0.56139558
## [38,] -0.36923513  0.85426007
## [39,] -0.68913266  1.22024214
## [40,] -0.15747136  0.72666180
## [41,] -0.97592204 -0.20249386
## [42,]  1.46461134  0.55606345
## [43,] -0.77340892  0.98374893
## [44,]  0.64020765  0.19751422
## [45,]  0.81140438 -1.08392045
## [46,]  0.07058968  1.48854290
## [47,]  0.67038346  0.37198268
## [48,] -1.14539236 -1.81877756
## [49,] -2.85036021 -1.20865887
## [50,]  0.03711969  0.77637460
```

```
RcppRoll::roll_sum(x,12)
```

```
##              [,1]       [,2]
##  [1,] -5.18340408  0.5150323
##  [2,] -6.97018803  0.9634259
##  [3,] -7.14682423  0.1473725
##  [4,] -8.25737988  1.4351774
##  [5,] -7.26890370  2.6259506
##  [6,] -8.83067080  0.7810657
##  [7,] -6.06532107  1.1591338
##  [8,] -4.90356063  2.3249971
##  [9,] -3.70267347  4.1617914
## [10,] -1.31865766  3.9565753
## [11,] -2.60451869  4.6148233
## [12,] -0.97877366  6.6817839
## [13,] -1.86459540  5.7184515
## [14,]  0.43029221  6.6783359
## [15,]  1.63051237  4.8118160
## [16,]  1.75879975  5.4006066
## [17,]  0.24461508  4.2538690
## [18,]  1.28934161  5.3190920
## [19,] -0.92958729  4.5306978
## [20,] -3.16411995  3.2062192
## [21,] -3.29836593  1.0964156
## [22,] -4.49541387  0.2813163
## [23,] -2.96698104 -0.5675278
## [24,] -4.03094319 -2.9319572
## [25,] -2.80390483 -2.0142570
## [26,] -2.40217136 -3.3524597
## [27,] -3.47252402 -1.2048979
## [28,] -3.57114069 -1.1594700
## [29,] -2.55243111 -0.1815571
## [30,] -2.69587962  0.3775718
## [31,] -0.02869798  0.1378104
## [32,]  1.17616974  1.4603396
## [33,]  2.38059106  2.7274104
## [34,]  2.01688777  1.8417925
## [35,]  1.33090296  3.9596597
## [36,]  1.11443697  5.3999793
## [37,]  0.15687593  2.7324287
## [38,] -3.30372617  2.0851655
## [39,] -2.89737135  2.0072800
```

```
# ranking (basically report the indices of elements if they were sorted, NA and INF goes to the end)
x <- c(5, 1, 3,Inf, 2, 2, NA) # => (1,2,2,3,5,Inf,NA)
row_number(x)
```

```
## [1]  5  1  4  6  2  3 NA
```

```
min_rank(x)
```

```
## [1]  5  1  4  6  2  2 NA
```

```
dense_rank(x)
```

```
## [1]  4  1  3  5  2  2 NA
```

```
percent_rank(x) # a number between 0 and 1 computed by rescaling min_rank to [0, 1]
```

```
## [1] 0.8 0.0 0.6 1.0 0.2 0.2  NA
```

```
cume_dist(x)  #a cumulative distribution function. Proportion of all values less than or equal to the c
```

```
## [1] 0.8333333 0.1666667 0.6666667 1.0000000 0.5000000 0.5000000        NA
```

```
# ntile creates a rough rank, which breaks the input vector into n buckets
ntile(x, 2)
```

```
## [1]  2  1  2  2  1  1 NA
```

```
ntile(runif(100), 10)
```

```
##   [1]  5  9  2  1  3 10 10 10  4  2  9  7  5  6  5  9  4  1  6 10  8  9  2  2  1
##  [26]  4 10  8  3 10  4  5 10  5  4  2  5  8  3  2 10  8  6  3  3  2  1  2 10  5
##  [51]  6  6  6 10  9  8  2  9  8  3  9  5  9  9  7  9  7  3  2  7  8  5  7  1  3
##  [76]  6  4  7  5  1  4  8  7  6  7  4  4  8  3  1  7  3  7  8  4  6  1  1  6  1
```

```
# 5.5.2 Exercises

# Convert dep_time and sched_dep_time to a more convenient representation of number of minutes since mi

#flights[order(flights$dep_time, na.last = T,decreasing = T), ]
flights1 <- mutate(flights, dep_time_minute = (dep_time%/%100) * 60 + dep_time%%100, sched_dep_time_minu
select (flights1,year,month, day,dep_time, dep_time_minute, sched_dep_time, sched_dep_time_minute, every
```

```
## # A tibble: 336,776 x 21
##     year month   day dep_time dep_time_minute sched_dep_time sched_dep_time_~
##    <int> <int> <int>    <int>           <dbl>          <int>            <dbl>
##  1  2013     1     1      517             317            515              315
##  2  2013     1     1      533             333            529              329
##  3  2013     1     1      542             342            540              340
##  4  2013     1     1      544             344            545              345
##  5  2013     1     1      554             354            600              360
##  6  2013     1     1      554             354            558              358
##  7  2013     1     1      555             355            600              360
##  8  2013     1     1      557             357            600              360
##  9  2013     1     1      557             357            600              360
## 10  2013     1     1      558             358            600              360
## # ... with 336,766 more rows, and 14 more variables: dep_delay <dbl>,
## #   arr_time <int>, 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>
```

```
# Compare air_time with arr_time - dep_time. What do you expect to see? What do you ne
fixed_air_time_flights <- transmute (flights, arr_time, dep_time, air_time, fixed_air_time_minute = abs
fixed_air_time_flights[order(fixed_air_time_flights$fixed_air_time), ]
```

```
## # A tibble: 336,776 x 5
##    arr_time dep_time air_time fixed_air_time_minute fixed_air_time
##       <int>    <int>    <dbl>                 <dbl>          <dbl>
##  1     1206     1133       23                    33             33
##  2     1358     1323       23                    35             35
##  3     1347     1312       23                    35             35
##  4     1238     1203       21                    35             35
##  5     1531     1455       22                    36             36
##  6      758      722       22                    36             36
##  7      758      722       22                    36             36
##  8      754      718       24                    36             36
##  9     1403     1326       22                    37             37
## 10     1533     1456       21                    37             37
## # ... with 336,766 more rows
```

```
filter(fixed_air_time_flights, arr_time <= dep_time)
```

```
## # A tibble: 10,633 x 5
##    arr_time dep_time air_time fixed_air_time_minute fixed_air_time
##       <int>    <int>    <dbl>                 <dbl>          <dbl>
##  1        3     1929      192                  1166           1926
##  2       29     1939       NA                  1150           1910
##  3        8     2058      159                  1250           2050
##  4      146     2102      199                  1156           1916
##  5       25     2108      354                  1243           2043
##  6       16     2120      160                  1264           2104
##  7        6     2121      143                  1275           2115
##  8       26     2128      338                  1262           2102
##  9       20     2134      152                  1274           2114
## 10       25     2136      154                  1271           2111
## # ... with 10,623 more rows
```

```
# Compare dep_time, sched_dep_time, and dep_delay. How would you expect those three numbers to be relat
transmute(flights, dep_time, sched_dep_time, dep_delay, dep_delay_fixed =  dep_time - sched_dep_time )
```

```
## # A tibble: 336,776 x 4
##    dep_time sched_dep_time dep_delay dep_delay_fixed
##       <int>          <int>     <dbl>           <int>
##  1      517            515         2               2
##  2      533            529         4               4
##  3      542            540         2               2
##  4      544            545        -1              -1
##  5      554            600        -6             -46
##  6      554            558        -4              -4
##  7      555            600        -5             -45
##  8      557            600        -3             -43
##  9      557            600        -3             -43
## 10      558            600        -2             -42
## # ... with 336,766 more rows
```

```
# Find the 10 most delayed flights using a ranking function. How do you want to handle ties? Carefully
filter (flights , min_rank(desc(dep_delay)) <= 10)
```

```
## # A tibble: 10 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     1    10     1121           1635      1126     1239           1810
##  3  2013    12     5      756           1700       896     1058           2020
##  4  2013     3    17     2321            810       911      135           1020
##  5  2013     4    10     1100           1900       960     1342           2211
##  6  2013     6    15     1432           1935      1137     1607           2120
##  7  2013     6    27      959           1900       899     1236           2226
##  8  2013     7    22      845           1600      1005     1044           1815
##  9  2013     7    22     2257            759       898      121           1026
## 10  2013     9    20     1139           1845      1014     1457           2210
## # ... 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>
```

```
flights[order(flights$dep_delay, decreasing = T), ]
```

```
## # 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     6    27      959           1900       899     1236           2226
##  9  2013     7    22     2257            759       898      121           1026
## 10  2013    12     5      756           1700       896     1058           2020
## # ... 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>
```

```
library(nycflights13)
library(tidyverse)

# summarize() with group_by() changes the unit of analysis from the complete dataset to individual grou
# Then, when you use the dplyr verbs on a grouped data frame they'll be automatically applied "by group

(by_day <- group_by(flights, year, month, day))
```

```
## # A tibble: 336,776 x 19
## # Groups:   year, month, day [365]
##     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>
```

```r
str(attributes(by_day))
```

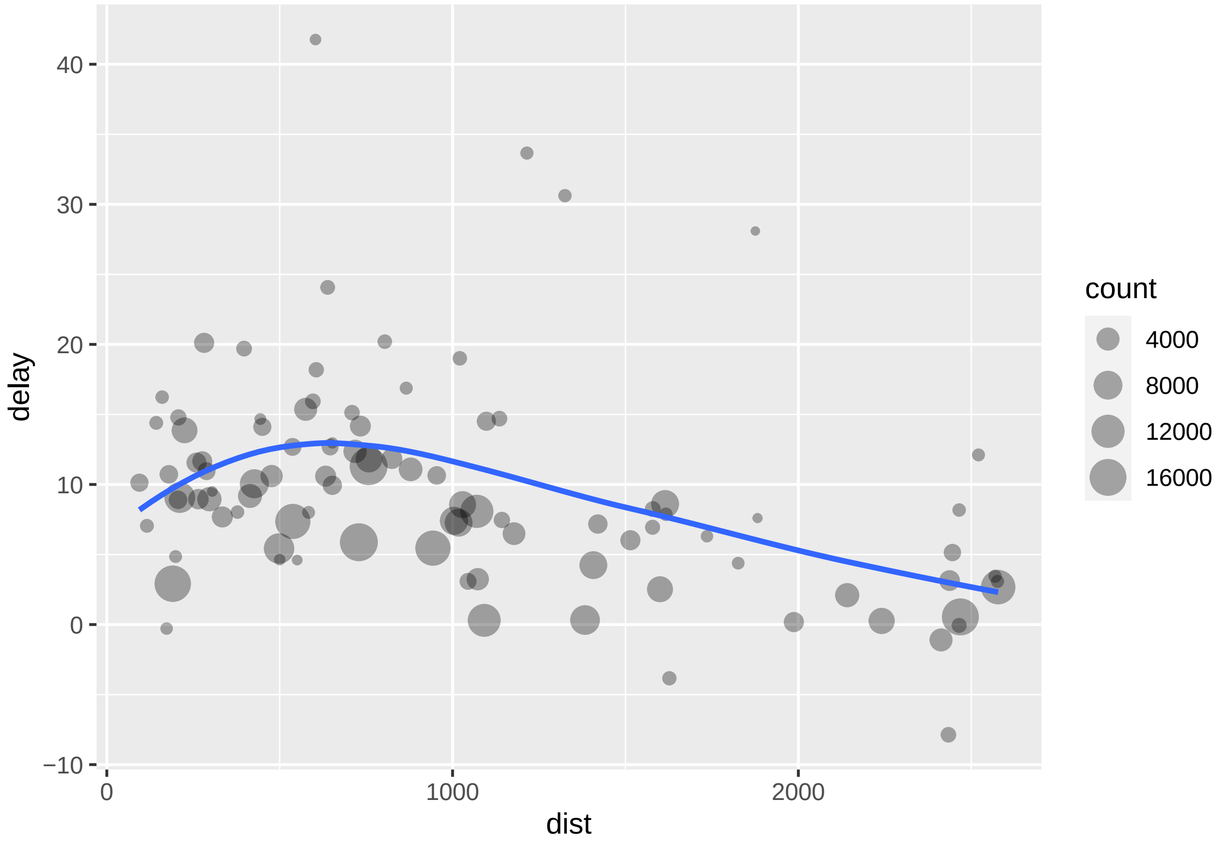
```
## List of 4
##  $ names    : chr [1:19] "year" "month" "day" "dep_time" ...
##  $ row.names: int [1:336776] 1 2 3 4 5 6 7 8 9 10 ...
##  $ class    : chr [1:4] "grouped_df" "tbl_df" "tbl" "data.frame"
##  $ groups   : tibble [365 x 4] (S3: tbl_df/tbl/data.frame)
##   ..$ year : int [1:365] 2013 2013 2013 2013 2013 2013 2013 2013 2013 2013 ...
##   ..$ month: int [1:365] 1 1 1 1 1 1 1 1 1 1 ...
##   ..$ day  : int [1:365] 1 2 3 4 5 6 7 8 9 10 ...
##   ..$ .rows:List of 365
##   .. ..$ : int [1:842] 1 2 3 4 5 6 7 8 9 10 ...
##   .. ..$ : int [1:943] 843 844 845 846 847 848 849 850 851 852 ...
##   .. ..$ : int [1:914] 1786 1787 1788 1789 1790 1791 1792 1793 1794 1795 ...
##   .. ..$ : int [1:915] 2700 2701 2702 2703 2704 2705 2706 2707 2708 2709 ...
##   .. ..$ : int [1:720] 3615 3616 3617 3618 3619 3620 3621 3622 3623 3624 ...
##   .. ..$ : int [1:832] 4335 4336 4337 4338 4339 4340 4341 4342 4343 4344 ...
##   .. ..$ : int [1:933] 5167 5168 5169 5170 5171 5172 5173 5174 5175 5176 ...
##   .. ..$ : int [1:899] 6100 6101 6102 6103 6104 6105 6106 6107 6108 6109 ...
##   .. ..$ : int [1:902] 6999 7000 7001 7002 7003 7004 7005 7006 7007 7008 ...
##   .. ..$ : int [1:932] 7901 7902 7903 7904 7905 7906 7907 7908 7909 7910 ...
##   .. ..$ : int [1:930] 8833 8834 8835 8836 8837 8838 8839 8840 8841 8842 ...
##   .. ..$ : int [1:690] 9763 9764 9765 9766 9767 9768 9769 9770 9771 9772 ...
##   .. ..$ : int [1:828] 10453 10454 10455 10456 10457 10458 10459 10460 10461 10462 ...
##   .. ..$ : int [1:928] 11281 11282 11283 11284 11285 11286 11287 11288 11289 11290 ...
##   .. ..$ : int [1:894] 12209 12210 12211 12212 12213 12214 12215 12216 12217 12218 ...
##   .. ..$ : int [1:901] 13103 13104 13105 13106 13107 13108 13109 13110 13111 13112 ...
##   .. ..$ : int [1:927] 14004 14005 14006 14007 14008 14009 14010 14011 14012 14013 ...
##   .. ..$ : int [1:924] 14931 14932 14933 14934 14935 14936 14937 14938 14939 14940 ...
##   .. ..$ : int [1:674] 15855 15856 15857 15858 15859 15860 15861 15862 15863 15864 ...
##   .. ..$ : int [1:786] 16529 16530 16531 16532 16533 16534 16535 16536 16537 16538 ...
##   .. ..$ : int [1:912] 17315 17316 17317 17318 17319 17320 17321 17322 17323 17324 ...
##   .. ..$ : int [1:890] 18227 18228 18229 18230 18231 18232 18233 18234 18235 18236 ...
##   .. ..$ : int [1:897] 19117 19118 19119 19120 19121 19122 19123 19124 19125 19126 ...
##   .. ..$ : int [1:925] 20014 20015 20016 20017 20018 20019 20020 20021 20022 20023 ...
##   .. ..$ : int [1:922] 20939 20940 20941 20942 20943 20944 20945 20946 20947 20948 ...
##   .. ..$ : int [1:680] 21861 21862 21863 21864 21865 21866 21867 21868 21869 21870 ...
##   .. ..$ : int [1:823] 22541 22542 22543 22544 22545 22546 22547 22548 22549 22550 ...
##   .. ..$ : int [1:923] 23364 23365 23366 23367 23368 23369 23370 23371 23372 23373 ...
##   .. ..$ : int [1:890] 24287 24288 24289 24290 24291 24292 24293 24294 24295 24296 ...
##   .. ..$ : int [1:900] 25177 25178 25179 25180 25181 25182 25183 25184 25185 25186 ...
##   .. ..$ : int [1:928] 26077 26078 26079 26080 26081 26082 26083 26084 26085 26086 ...
##   .. ..$ : int [1:926] 111297 111298 111299 111300 111301 111302 111303 111304 111305 111306 ...
##   .. ..$ : int [1:682] 112223 112224 112225 112226 112227 112228 112229 112230 112231 112232 ...
##   .. ..$ : int [1:814] 112905 112906 112907 112908 112909 112910 112911 112912 112913 112914 ...
##   .. ..$ : int [1:932] 113719 113720 113721 113722 113723 113724 113725 113726 113727 113728 ...
##   .. ..$ : int [1:896] 114651 114652 114653 114654 114655 114656 114657 114658 114659 114660 ...
##   .. ..$ : int [1:901] 115547 115548 115549 115550 115551 115552 115553 115554 115555 115556 ...
##   .. ..$ : int [1:932] 116448 116449 116450 116451 116452 116453 116454 116455 116456 116457 ...
##   .. ..$ : int [1:930] 117380 117381 117382 117383 117384 117385 117386 117387 117388 117389 ...
##   .. ..$ : int [1:684] 118310 118311 118312 118313 118314 118315 118316 118317 118318 118319 ...
##   .. ..$ : int [1:829] 118994 118995 118996 118997 118998 118999 119000 119001 119002 119003 ...
##   .. ..$ : int [1:929] 119823 119824 119825 119826 119827 119828 119829 119830 119831 119832 ...
##   .. ..$ : int [1:893] 120752 120753 120754 120755 120756 120757 120758 120759 120760 120761 ...
##   .. ..$ : int [1:918] 121645 121646 121647 121648 121649 121650 121651 121652 121653 121654 ...
##   .. ..$ : int [1:956] 122563 122564 122565 122566 122567 122568 122569 122570 122571 122572 ...
##   .. ..$ : int [1:954] 123519 123520 123521 123522 123523 123524 123525 123526 123527 123528 ...
##   .. ..$ : int [1:738] 124473 124474 124475 124476 124477 124478 124479 124480 124481 124482 ...
##   .. ..$ : int [1:848] 125211 125212 125213 125214 125215 125216 125217 125218 125219 125220 ...
##   .. ..$ : int [1:948] 126059 126060 126061 126062 126063 126064 126065 126066 126067 126068 ...
##   .. ..$ : int [1:943] 127007 127008 127009 127010 127011 127012 127013 127014 127015 127016 ...
##   .. ..$ : int [1:949] 127950 127951 127952 127953 127954 127955 127956 127957 127958 127959 ...
##   .. ..$ : int [1:961] 128899 128900 128901 128902 128903 128904 128905 128906 128907 128908 ...
##   .. ..$ : int [1:957] 129860 129861 129862 129863 129864 129865 129866 129867 129868 129869 ...
##   .. ..$ : int [1:743] 130817 130818 130819 130820 130821 130822 130823 130824 130825 130826 ...
##   .. ..$ : int [1:880] 131560 131561 131562 131563 131564 131565 131566 131567 131568 131569 ...
##   .. ..$ : int [1:961] 132440 132441 132442 132443 132444 132445 132446 132447 132448 132449 ...
##   .. ..$ : int [1:938] 133401 133402 133403 133404 133405 133406 133407 133408 133409 133410 ...
##   .. ..$ : int [1:945] 134339 134340 134341 134342 134343 134344 134345 134346 134347 134348 ...
##   .. ..$ : int [1:964] 135284 135285 135286 135287 135288 135289 135290 135291 135292 135293 ...
##   .. ..$ : int [1:958] 136248 136249 136250 136251 136252 136253 136254 136255 136256 136257 ...
##   .. ..$ : int [1:765] 137206 137207 137208 137209 137210 137211 137212 137213 137214 137215 ...
##   .. ..$ : int [1:913] 137971 137972 137973 137974 137975 137976 137977 137978 137979 137980 ...
##   .. ..$ : int [1:977] 138884 138885 138886 138887 138888 138889 138890 138891 138892 138893 ...
##   .. ..$ : int [1:965] 139861 139862 139863 139864 139865 139866 139867 139868 139869 139870 ...
##   .. ..$ : int [1:972] 140826 140827 140828 140829 140830 140831 140832 140833 140834 140835 ...
##   .. ..$ : int [1:980] 141798 141799 141800 141801 141802 141803 141804 141805 141806 141807 ...
##   .. ..$ : int [1:979] 142778 142779 142780 142781 142782 142783 142784 142785 142786 142787 ...
##   .. ..$ : int [1:765] 143757 143758 143759 143760 143761 143762 143763 143764 143765 143766 ...
##   .. ..$ : int [1:908] 144522 144523 144524 144525 144526 144527 144528 144529 144530 144531 ...
##   .. ..$ : int [1:980] 145430 145431 145432 145433 145434 145435 145436 145437 145438 145439 ...
##   .. ..$ : int [1:966] 146410 146411 146412 146413 146414 146415 146416 146417 146418 146419 ...
##   .. ..$ : int [1:974] 147376 147377 147378 147379 147380 147381 147382 147383 147384 147385 ...
##   .. ..$ : int [1:982] 148350 148351 148352 148353 148354 148355 148356 148357 148358 148359 ...
##   .. ..$ : int [1:979] 149332 149333 149334 149335 149336 149337 149338 149339 149340 149341 ...
##   .. ..$ : int [1:767] 150311 150312 150313 150314 150315 150316 150317 150318 150319 150320 ...
##   .. ..$ : int [1:907] 151078 151079 151080 151081 151082 151083 151084 151085 151086 151087 ...
##   .. ..$ : int [1:981] 151985 151986 151987 151988 151989 151990 151991 151992 151993 151994 ...
##   .. ..$ : int [1:967] 152966 152967 152968 152969 152970 152971 152972 152973 152974 152975 ...
##   .. ..$ : int [1:970] 153933 153934 153935 153936 153937 153938 153939 153940 153941 153942 ...
##   .. ..$ : int [1:980] 154903 154904 154905 154906 154907 154908 154909 154910 154911 154912 ...
##   .. ..$ : int [1:977] 155883 155884 155885 155886 155887 155888 155889 155890 155891 155892 ...
##   .. ..$ : int [1:767] 156860 156861 156862 156863 156864 156865 156866 156867 156868 156869 ...
##   .. ..$ : int [1:905] 157627 157628 157629 157630 157631 157632 157633 157634 157635 157636 ...
##   .. ..$ : int [1:978] 158532 158533 158534 158535 158536 158537 158538 158539 158540 158541 ...
##   .. ..$ : int [1:973] 159510 159511 159512 159513 159514 159515 159516 159517 159518 159519 ...
##   .. ..$ : int [1:977] 160483 160484 160485 160486 160487 160488 160489 160490 160491 160492 ...
##   .. ..$ : int [1:982] 161460 161461 161462 161463 161464 161465 161466 161467 161468 161469 ...
##   .. ..$ : int [1:974] 162442 162443 162444 162445 162446 162447 162448 162449 162450 162451 ...
##   .. ..$ : int [1:769] 163416 163417 163418 163419 163420 163421 163422 163423 163424 163425 ...
##   .. ..$ : int [1:897] 164185 164186 164187 164188 164189 164190 164191 164192 164193 164194 ...
##   .. ..$ : int [1:970] 165082 165083 165084 165085 165086 165087 165088 165089 165090 165091 ...
##   .. ..$ : int [1:983] 166052 166053 166054 166055 166056 166057 166058 166059 166060 166061 ...
##   .. ..$ : int [1:992] 167035 167036 167037 167038 167039 167040 167041 167042 167043 167044 ...
##   .. ..$ : int [1:985] 168027 168028 168029 168030 168031 168032 168033 168034 168035 168036 ...
##   .. ..$ : int [1:981] 169012 169013 169014 169015 169016 169017 169018 169019 169020 169021 ...
##   .. ..$ : int [1:770] 169993 169994 169995 169996 169997 169998 169999 170000 170001 170002 ...
##   .. ..$ : int [1:911] 170763 170764 170765 170766 170767 170768 170769 170770 170771 170772 ...
##   .. ..$ : int [1:981] 171674 171675 171676 171677 171678 171679 171680 171681 171682 171683 ...
##   .. ..$ : int [1:975] 172655 172656 172657 172658 172659 172660 172661 172662 172663 172664 ...
##   .. .. [list output truncated]
##   ..- attr(*, ".drop")= logi TRUE
```

```r
# delay by day
summarise(by_day, delay = mean(dep_delay, na.rm = T))
```

```
## # 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
```

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


# Explore the relationship between the distance and average delay for each location
delay <- flights %>%
  group_by(dest) %>%
  summarise(count = n(), dist = mean(distance, na.rm = T), delay = mean(arr_delay, na.rm = T)) %>%
  filter(count > 20 & dest != "HNL")

ggplot(data = delay, mapping = aes(x = dist, y = delay))+
  geom_point(aes(size=count), alpha=1/3)+
  geom_smooth(se=F)
```

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



```
# compare it with:
# by_destination <- group_by(flights, dest)
# (avgDelayByDest <- summarise(by_destination, count = n(), dist = mean(distance, na.rm = T), delay = m
# (delay <- filter(avgDelayByDest, count > 20 & dest != "HNL"))
```

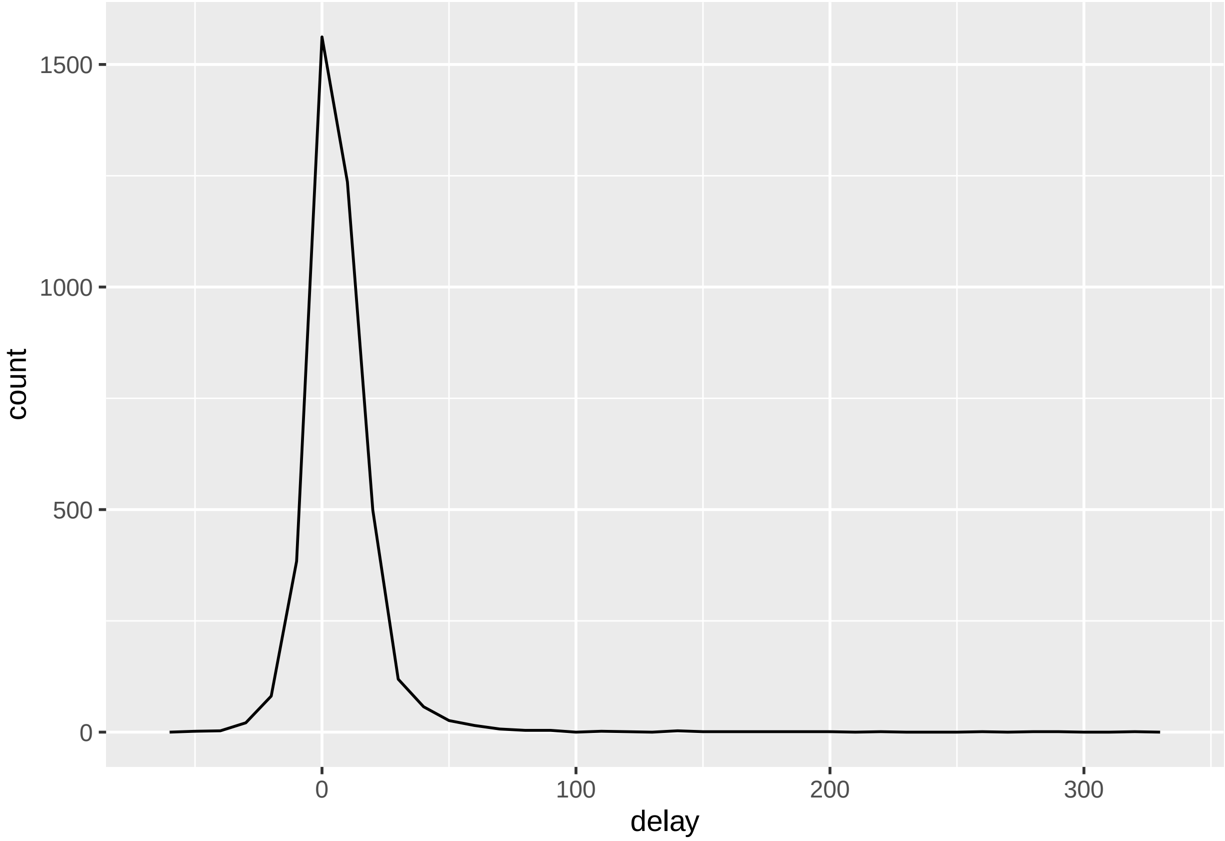
```
library(nycflights13)
library(tidyverse)

# count missing values in arr_delay
missing_arr_delay <- flights %>%
  filter(is.na(arr_delay) | is.na(dep_delay)) %>%
  group_by(arr_delay) %>%
  summarise(count = n())

sprintf("Number of records with missing arr_delay or dep_delay: %d",missing_arr_delay$count)
```

```
## [1] "Number of records with missing arr_delay or dep_delay: 9430"
```

```
# Remove records with missing arr_delay or dep_delay
(not_cancelled <- flights %>%
  filter(!is.na(arr_delay) & !is.na(dep_delay))
)
```

```
## # A tibble: 327,346 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 327,336 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>
```

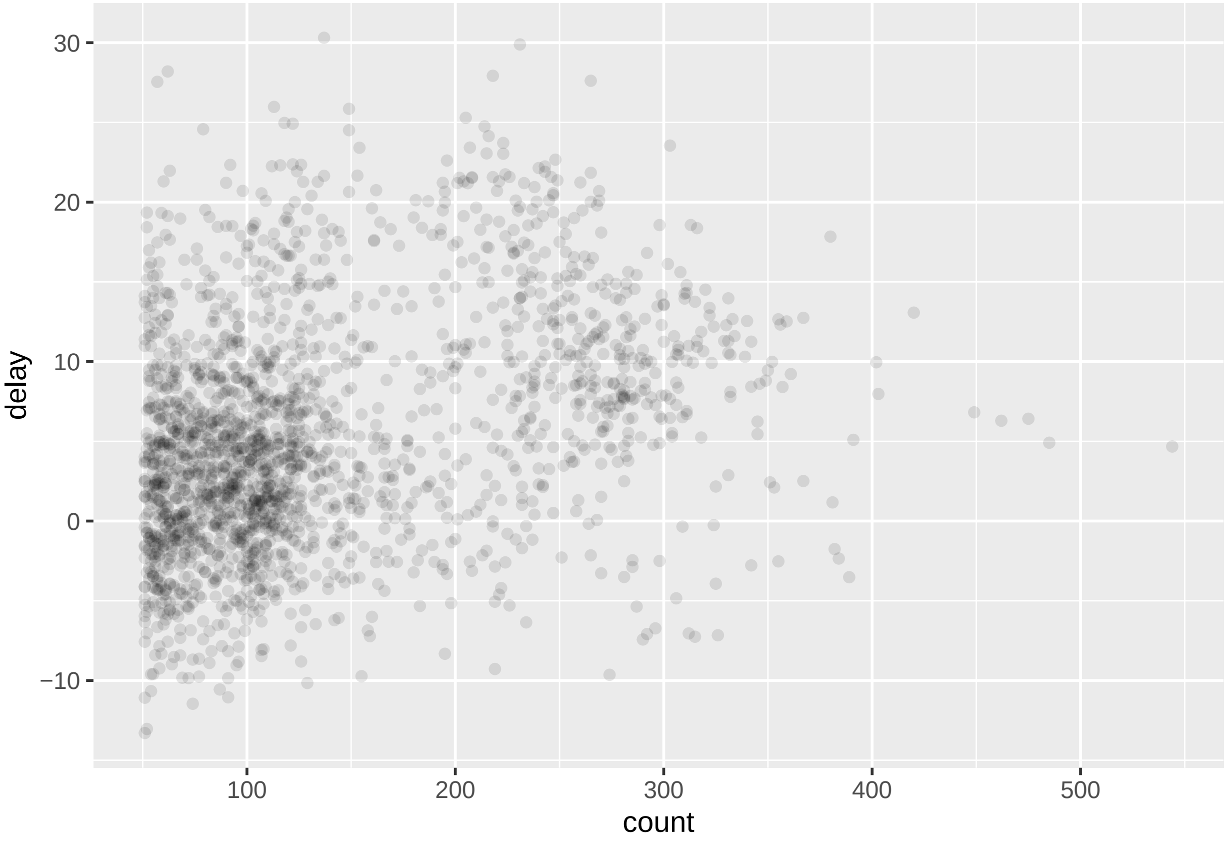
```
# let’s look at the planes (identified by their tail number) that have the highest average delays:

delays <- not_cancelled %>%
  group_by(tailnum) %>%
  summarise(delay = mean(arr_delay))

ggplot(data = delays, mapping = aes(x = delay)) +
  geom_freqpoly(binwidth = 10)
```



```
# Wow, there are some planes that have an average delay of 5 hours (300 minutes)!
# Let's draw a scatterplot of number of flights vs. average delay:

(delays1 <- not_cancelled %>%
  group_by(tailnum) %>%
  summarise(count = n(), delay = mean (arr_delay))
)
```

```
## # A tibble: 4,037 x 3
##    tailnum count  delay
##    <chr>   <int>  <dbl>
##  1 D942DN      4 31.5
##  2 N0EGMQ    352  9.98
##  3 N10156    145 12.7
##  4 N102UW     48  2.94
##  5 N103US     46 -6.93
##  6 N104UW     46  1.80
##  7 N10575    269 20.7
##  8 N105UW     45 -0.267
##  9 N107US     41 -5.73
## 10 N108UW     60 -1.25
## # ... with 4,027 more rows
```

```
# filter(delays1 , is.na(delay) | is.na(count))
delays1 %>%
  # it's often useful to filter out the groups with the smallest numbers of observations to see pattern
  filter (count > 50) %>%
  ggplot(mapping = aes(x=count, y=delay))+
  geom_point(alpha=1/10)
```

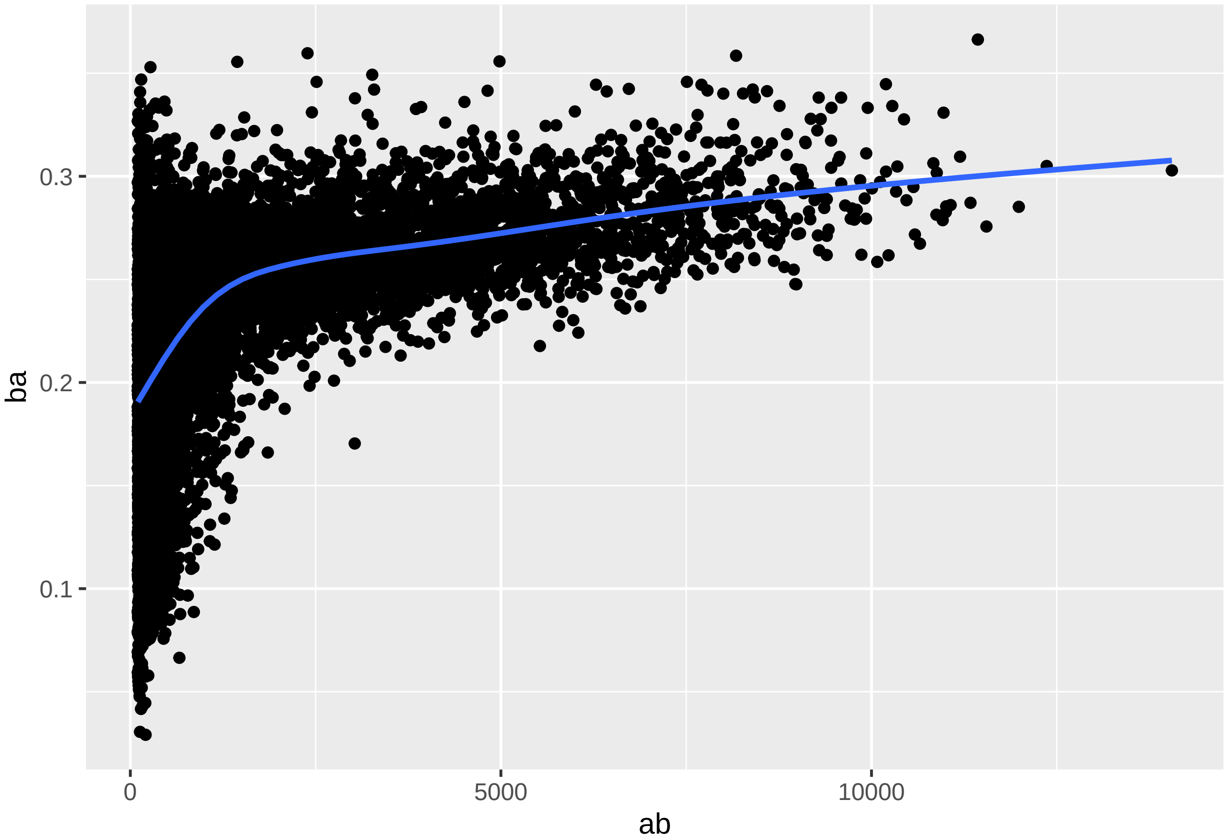


```
# ANORTHER EXAMPLE:
# When I plot the skill of the batter (measured by the batting average, ba) against the number of oppor
# (measured by at bat, ab), you see two patterns:

batting <- as_tibble(Lahman::Batting)
(batters <- batting %>%
  group_by(playerID) %>%
  summarise(ba = sum(H, na.rm = T)/sum(AB, na.rm = T), ab = sum(AB, na.rm = T))
)
```

```
## # A tibble: 19,428 x 3
##    playerID      ba    ab
##    <chr>      <dbl> <int>
##  1 aardsda01 0          4
##  2 aaronha01 0.305  12364
##  3 aaronto01 0.229    944
##  4 aasedo01  0          5
##  5 abadan01  0.0952    21
##  6 abadfe01  0.111      9
##  7 abadijo01 0.224     49
##  8 abbated01 0.254   3044
##  9 abbeybe01 0.169    225
## 10 abbeych01 0.281   1756
## # ... with 19,418 more rows
```

```
# Variation in our aggregate decreases as we get more data points.
# There's a positive correlation between skill (ba) and opportunities to hit the ball (ab).

batters %>%
 filter (ab > 100) %>%
  ggplot(mapping = aes(x=ab, y = ba)) +
  geom_point()+
  geom_smooth(se=F)
```

```
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```



```r
# If you naively sort on desc(ba), the people with the best batting averages are clearly lucky, not ski

batters %>%
  arrange(desc(ba))
```

```
## # A tibble: 19,428 x 3
##    playerID     ba    ab
##    <chr>     <dbl> <int>
##  1 abramge01     1     1
##  2 alberan01     1     1
##  3 allarko01     1     1
##  4 banisje01     1     1
##  5 bartocl01     1     1
##  6 bassdo01      1     1
##  7 birasst01     1     2
##  8 bruneju01     1     1
##  9 burnscb01     1     1
## 10 cammaer01     1     1
## # ... with 19,418 more rows
```

```r
# sum(x > 10) can take a logical expression that filters in certain records and then it adds them (sum
# mean (x > 60) can filters in records whose column 'x' value is greater than 60 and  then calculate th
# Measures of location: mean(x), median(x) (Half of the values of x is less than median(x) and other ha
# Measures of spread: sd(x), IQR(x), mad(x)
# Measures of rank: min(x), quantile(x, 0.25), max(x)
# Measures of position: first(x), nth(x, 2), last(x). These work similarly to x[1], x[2], and x[length(

# Counts:
```

```
#   n() : returns the size of the current group.
#   count(x) : counts number of repeatitions of each element in a qualitative column x
#   sum(!is.na(x)) : number of non-missing values in current group
#   n_distinct(x) : number of distinct (unique) values in current group

# Counts and proportions of logical values: sum(x > 10), mean(y == 0)

# quantile(x, 0.25) will find a value of x that is greater than 25% of the values, and less than the re
# IQR is 3rd Quartile - 1st Quartile (i.e the box plot)
# mad is median absolute deviation mad(x) may be more useful if you have outliers


library(nycflights13)
library(tidyverse)

(not_cancelled <- flights %>%
  filter(!is.na(arr_delay) & !is.na(dep_delay))
)
```

```
## # A tibble: 327,346 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 327,336 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>
```

```
(not_cancelled %>%
  group_by(year, month, day) %>%
  summarise(avg_arr_delay = mean(arr_delay), avg_pos_arr_delay = mean(arr_delay[arr_delay > 0]))
)
```

```
## # A tibble: 365 x 5
## # Groups:   year, month [12]
##     year month   day avg_arr_delay avg_pos_arr_delay
##    <int> <int> <int>         <dbl>             <dbl>
##  1  2013     1     1        12.7                32.5
##  2  2013     1     2        12.7                32.0
##  3  2013     1     3         5.73               27.7
##  4  2013     1     4        -1.93               28.3
##  5  2013     1     5        -1.53               22.6
##  6  2013     1     6         4.24               24.4
##  7  2013     1     7        -4.95               27.8
##  8  2013     1     8        -3.23               20.8
##  9  2013     1     9        -0.264              25.6
## 10  2013     1    10        -5.90               27.3
## # ... with 355 more rows
```

```
not_cancelled %>%
  group_by(dest) %>%
  summarise(distance_sd = sd(distance)) %>%
  arrange(desc(distance_sd))
```

```
## # A tibble: 104 x 2
##    dest  distance_sd
##    <chr>       <dbl>
##  1 EGE         10.5
##  2 SAN         10.4
##  3 SFO         10.2
##  4 HNL         10.0
##  5 SEA          9.98
##  6 LAS          9.91
##  7 PDX          9.87
##  8 PHX          9.86
##  9 LAX          9.66
## 10 IND          9.46
## # ... with 94 more rows
```

```
not_cancelled %>%
  group_by(year, month, day) %>%
  summarise(first = min(dep_time), last=max(dep_time))
```

```
## # A tibble: 365 x 5
## # Groups:   year, month [12]
##     year month   day first  last
##    <int> <int> <int> <int> <int>
##  1  2013     1     1   517  2356
##  2  2013     1     2    42  2354
##  3  2013     1     3    32  2349
##  4  2013     1     4    25  2358
##  5  2013     1     5    14  2357
##  6  2013     1     6    16  2355
##  7  2013     1     7    49  2359
##  8  2013     1     8   454  2351
##  9  2013     1     9     2  2252
## 10  2013     1    10     3  2320
## # ... with 355 more rows
```

```
not_cancelled %>%
  group_by(year, month, day) %>%
  summarise(first_dep = first(dep_time), last_dep = last(dep_time))
```

```
## # A tibble: 365 x 5
## # Groups:   year, month [12]
##     year month   day first_dep last_dep
##    <int> <int> <int>     <int>    <int>
##  1  2013     1     1       517     2356
##  2  2013     1     2        42     2354
##  3  2013     1     3        32     2349
##  4  2013     1     4        25     2358
##  5  2013     1     5        14     2357
##  6  2013     1     6        16     2355
##  7  2013     1     7       49      2359
##  8  2013     1     8      454      2351
##  9  2013     1     9        2      2252
## 10  2013     1    10        3      2320
## # ... with 355 more rows
```

```
# Filtering on ranks gives you all variables, with each observation in a separate row:
not_cancelled %>%
  group_by(year, month, day) %>%
  mutate(rank = min_rank(desc(dep_time))) %>%
  filter(rank %in% range(rank))
```

```
## # A tibble: 770 x 20
## # Groups:   year, month, day [365]
##     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     2356           2359        -3      425            437
##  3  2013     1     2       42           2359        43      518            442
##  4  2013     1     2     2354           2359        -5      413            437
##  5  2013     1     3       32           2359        33      504            442
##  6  2013     1     3     2349           2359       -10      434            445
##  7  2013     1     4       25           2359        26      505            442
##  8  2013     1     4     2358           2359        -1      429            437
##  9  2013     1     4     2358           2359        -1      436            445
## 10  2013     1     5       14           2359        15      503            445
## # ... with 760 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>,
## #   rank <int>
```

```
# which destination have the most carriers
not_cancelled %>%
  filter(!is.na(dest) & !is.na(carrier)) %>%
  group_by(dest) %>%
  summarize(max_carriers = max(n_distinct(carrier))) %>%
  arrange(desc(max_carriers))
```

```
## # A tibble: 104 x 2
##    dest  max_carriers
##    <chr>        <int>
##  1 ATL              7
##  2 BOS              7
##  3 CLT              7
##  4 ORD              7
##  5 TPA              7
##  6 AUS              6
##  7 DCA              6
##  8 DTW              6
##  9 IAD              6
## 10 MSP              6
## # ... with 94 more rows
```

```
# Give a count of each destinations separately
not_cancelled %>%
  count(dest)
```

```
## # A tibble: 104 x 2
##    dest      n
##    <chr> <int>
##  1 ABQ     254
##  2 ACK     264
##  3 ALB     418
##  4 ANC       8
##  5 ATL   16837
##  6 AUS    2411
##  7 AVL     261
##  8 BDL     412
##  9 BGR     358
## 10 BHM     269
## # ... with 94 more rows
```

```
# You can optionally provide a weight variable.
# "count" (sum) the total number of miles a plane flew:

not_cancelled %>%
  count(tailnum, wt=distance)
```

```
## # A tibble: 4,037 x 2
##    tailnum      n
##    <chr>    <dbl>
##  1 D942DN    3418
##  2 N0EGMQ  239143
##  3 N10156  109664
##  4 N102UW   25722
##  5 N103US   24619
##  6 N104UW   24616
##  7 N10575  139903
##  8 N105UW   23618
##  9 N107US   21677
## 10 N108UW   32070
## # ... with 4,027 more rows
```

```
# Whcih is the same as
not_cancelled %>%
  group_by(tailnum) %>%
  summarise(n=sum(distance))
```

```
## # A tibble: 4,037 x 2
##    tailnum      n
##    <chr>    <dbl>
##  1 D942DN    3418
##  2 N0EGMQ  239143
##  3 N10156  109664
##  4 N102UW   25722
##  5 N103US   24619
##  6 N104UW   24616
##  7 N10575  139903
##  8 N105UW   23618
##  9 N107US   21677
## 10 N108UW   32070
## # ... with 4,027 more rows
```

```r
# How many flights left before 5am?
not_cancelled %>%
  group_by(year, month, day) %>%
  summarise(n_early = sum(dep_time < 500))
```

```
## # A tibble: 365 x 4
## # Groups:   year, month [12]
##     year month   day n_early
##    <int> <int> <int>   <int>
##  1  2013     1     1       0
##  2  2013     1     2       3
##  3  2013     1     3       4
##  4  2013     1     4       3
##  5  2013     1     5       3
##  6  2013     1     6       2
##  7  2013     1     7       2
##  8  2013     1     8       1
##  9  2013     1     9       3
## 10  2013     1    10       3
## # ... with 355 more rows
```

```r
# What proportion of flights are delayed by more than an hour?
not_cancelled %>%
  group_by(year, month, day) %>%
  summarize(proportion = mean(arr_delay > 60))
```

```
## # A tibble: 365 x 4
## # Groups:   year, month [12]
##     year month   day proportion
##    <int> <int> <int>      <dbl>
##  1  2013     1     1     0.0722
##  2  2013     1     2     0.0851
##  3  2013     1     3     0.0567
##  4  2013     1     4     0.0396
##  5  2013     1     5     0.0349
##  6  2013     1     6     0.0470
##  7  2013     1     7     0.0333
##  8  2013     1     8     0.0213
##  9  2013     1     9     0.0202
## 10  2013     1    10     0.0183
## # ... with 355 more rows
```

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

(per_day <- flights %>%
  group_by(year, month, day) %>%
   summarise(flights=n())
)
```

```
## # A tibble: 365 x 4
## # Groups:   year, month [12]
##     year month   day flights
##    <int> <int> <int>   <int>
##  1  2013     1     1     842
##  2  2013     1     2     943
##  3  2013     1     3     914
##  4  2013     1     4     915
##  5  2013     1     5     720
##  6  2013     1     6     832
##  7  2013     1     7     933
##  8  2013     1     8     899
##  9  2013     1     9     902
## 10  2013     1    10     932
## # ... with 355 more rows
```

```
(per_month <- per_day %>%
  summarize(flights = sum(flights)))
```

```
## # A tibble: 12 x 3
## # Groups:   year [1]
##     year month flights
##    <int> <int>   <int>
##  1  2013     1   27004
##  2  2013     2   24951
##  3  2013     3   28834
##  4  2013     4   28330
##  5  2013     5   28796
##  6  2013     6   28243
##  7  2013     7   29425
##  8  2013     8   29327
##  9  2013     9   27574
## 10  2013    10   28889
## 11  2013    11   27268
## 12  2013    12   28135
```

```
(per_year <- per_month %>%
    summarize(flights = sum(flights)))
```

```
## # A tibble: 1 x 2
##    year flights
##   <int>   <int>
## 1  2013  336776
```

```
# Equivalently

(per_day <- flights %>%
  group_by(year, month, day) %>%
   summarise(per_day_flights=n()) %>%
    summarise(per_month_flights = sum(per_day_flights)) %>%
    summarise(per_year_flights = sum(per_month_flights))
)
```

```
## # A tibble: 1 x 2
##    year per_year_flights
##   <int>            <int>
## 1  2013           336776
```

```
# Be careful when progressively rolling up summaries: it's OK for sums and counts, but you need to thin
# I.e. the sum of groupwise sums is the overall sum, but the median of groupwise medians is not the ove

# If you need to remove grouping, and return to operations on ungrouped data, use ungroup()
```

```r
daily <- group_by(flights, year, month, day)

daily %>%
  ungroup() %>% # no longer grouped by year-month-day
  summarise(flights = n())
```

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

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

not_cancelled <- flights %>%
  filter(!is.na(arr_delay) & (!is.na(dep_delay)))
# 1)
# A flight is 15 minutes early 50% of the time, and 15 minutes late 50% of the time.
(not_cancelled %>%
    group_by(flight) %>%
    summarize(total = n(),
              early15 = sum(arr_delay == -15),
              late15 = sum (arr_delay == 15)) %>%
    filter(total != 0 & early15 != 0 & late15 != 0 &  near(total / 2, 0.5))

  )
```

```
## # A tibble: 0 x 4
## # ... with 4 variables: flight <int>, total <int>, early15 <int>, late15 <int>
```

```r
# Another way :
(not_cancelled %>%
    group_by(flight) %>%
    summarize(total = n(),
              early15 = mean(arr_delay == -15, na.rm = T),
              late15 = mean (arr_delay == 15, na.rm = T)) %>%
    filter(total != 0 & early15 == 0.5 & late15 == 0.5)

  )
```

```
## # A tibble: 0 x 4
## # ... with 4 variables: flight <int>, total <int>, early15 <dbl>, late15 <dbl>
```

```r
# A flight is always 10 minutes late.
(not_cancelled %>%
    group_by(flight) %>%
    filter (arr_delay == 10)
)
```

```
## # A tibble: 3,373 x 19
## # Groups:   flight [1,475]
##     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      624            630        -6      840            830
##  2  2013     1     1      717            720        -3      850            840
##  3  2013     1     1      745            745         0     1135           1125
##  4  2013     1     1      805            805         0     1015           1005
##  5  2013     1     1      811            815        -4     1026           1016
##  6  2013     1     1      921            900        21     1237           1227
##  7  2013     1     1     1158           1205        -7     1530           1520
##  8  2013     1     1     1211           1215        -4     1423           1413
##  9  2013     1     1     1455           1459        -4     1655           1645
## 10  2013     1     1     1554           1600        -6     1830           1820
## # ... with 3,363 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>
```

```
# 99% of the time a flight is on time. 1% of the time it's 2 hours late.

not_cancelled %>%
    group_by(flight) %>%
    summarize (total = n(), ontime = sum(arr_delay == 0), late = sum(arr_delay == 2)) %>%
  filter((ontime %/% total)*100 == 99 && (late %/% total)*100 == 1)
```

```
## # A tibble: 0 x 4
## # ... with 4 variables: flight <int>, total <int>, ontime <int>, late <int>
```

```
# 2)
#  not_cancelled %>% count(dest)
not_cancelled %>%
  group_by(dest) %>%
  summarise(n = n())
```

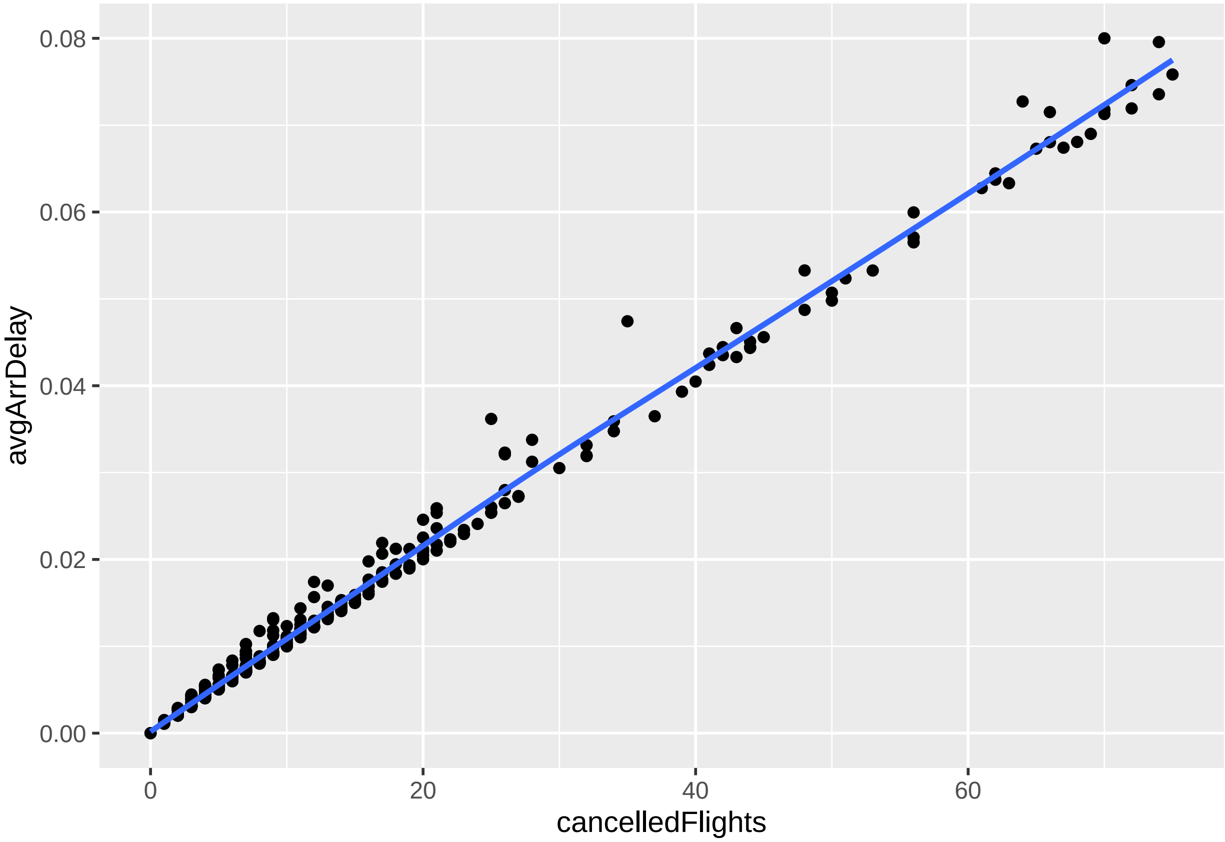
```
## # A tibble: 104 x 2
##    dest      n
##    <chr> <int>
##  1 ABQ     254
##  2 ACK     264
##  3 ALB     418
##  4 ANC       8
##  5 ATL   16837
##  6 AUS    2411
##  7 AVL     261
##  8 BDL     412
##  9 BGR     358
## 10 BHM     269
## # ... with 94 more rows
```

```
# not_cancelled %>% count(tailnum, wt = distance)
not_cancelled %>%
  group_by(tailnum) %>%
  summarise(wt = sum(distance))
```

```
## # A tibble: 4,037 x 2
##    tailnum      wt
##    <chr>     <dbl>
##  1 D942DN     3418
##  2 N0EGMQ   239143
##  3 N10156   109664
##  4 N102UW    25722
##  5 N103US    24619
##  6 N104UW    24616
##   7 N10575  139903
##   8 N105UW   23618
##   9 N107US   21677
##  10 N108UW   32070
## # ... with 4,027 more rows
```

```
# 4)
# Look at the number of cancelled flights per day. Is there a pattern? Is the proportion of cancelled f

flights %>%
  group_by(year, month, day) %>%
  summarize(cancelledFlights = sum(is.na(arr_delay) | (is.na(dep_delay))), avgArrDelay = mean(is.na(arr
  arrange(desc(avgArrDelay)) %>%
  filter(cancelledFlights <= 75) %>%
  ggplot(mapping = aes(x=cancelledFlights, y = avgArrDelay)) +
  geom_point()+
  geom_smooth(se=F)
```

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



```
# 5)
# Which carrier has the worst delays?
# Challenge: can you disentangle the effects of bad airports vs. bad carriers? Why/why not?
# (Hint: think about flights %>% group_by(carrier, dest) %>% summarise(n()))


# not_cancelled %>%
#   group_by(carrier, dest) %>%
```

```r
#    summarise(max_delay_per_dest = max(arr_delay, na.rm = T)) %>%
#      summarise(max_delay_per_carr = max(max_delay_per_dest, na.rm = T)) %>%
#   arrange(desc(max_delay_per_carr))

not_cancelled %>%
  group_by(carrier, dest) %>%
  summarise(max_delay_per_dest = max(arr_delay, na.rm = T)) %>%
  group_by(dest) %>%
  mutate(rank = min_rank(desc(max_delay_per_dest))) %>%
  filter(rank %in% range(rank)) %>%
  arrange(carrier, dest)
```

```
## # A tibble: 180 x 4
## # Groups:   dest [104]
##    carrier dest  max_delay_per_dest  rank
##    <chr>   <chr>              <dbl> <int>
##  1 9E      ATL                   55     7
##  2 9E      AUS                   25     6
##  3 9E      AVL                   13     2
##  4 9E      BTV                   -1     3
##  5 9E      BUF                  396     1
##  6 9E      CAE                   55     2
##  7 9E      CLT                  744     1
##  8 9E      CMH                   70     3
##  9 9E      DAY                  292     1
## 10 9E      DCA                  384     1
## # ... with 170 more rows
```

```r
# 6) What does the sort argument to count() do?
not_cancelled %>%
  count(tailnum, wt = distance, sort = T)
```

```
## # A tibble: 4,037 x 2
##    tailnum      n
##    <chr>    <dbl>
##  1 N328AA  929090
##  2 N338AA  921172
##  3 N335AA  902271
##  4 N327AA  900482
##  5 N323AA  839468
##  6 N319AA  837924
##  7 N336AA  833136
##  8 N329AA  825826
##  9 N324AA  786159
## 10 N339AA  783648
## # ... with 4,027 more rows
```

```r
# assume we want to count (number, letter) pair
(data = tibble(
  letter = sample(LETTERS, 50000, replace = TRUE),
  number = sample (1:10, 50000, replace = TRUE)
  ))
```

```
## # A tibble: 50,000 x 2
##    letter number
##    <chr>   <int>
##  1 D           4
##  2 R          10
##  3 E          10
##  4 R           6
##  5 M           3
##  6 G           4
##  7 F           8
##  8 A           1
##  9 J           2
## 10 U           9
## # ... with 49,990 more rows
```

```
data %>%
  count(letter, number, sort = TRUE)
```

```
## # A tibble: 260 x 3
##    letter number     n
##    <chr>   <int> <int>
##  1 E           8   233
##  2 F           5   229
##  3 T           9   227
##  4 Q           9   224
##  5 J           4   222
##  6 I           4   221
##  7 P           8   221
##  8 F           2   220
##  9 R           3   220
## 10 S           7   220
## # ... with 250 more rows
```

```
data %>%
  group_by(letter, number) %>%
  summarise(n = n()) %>%
  ungroup() %>%
  arrange(desc(n))
```

```
## # A tibble: 260 x 3
##    letter number     n
##    <chr>   <int> <int>
##  1 E           8   233
##  2 F           5   229
##  3 T           9   227
##  4 Q           9   224
##  5 J           4   222
##  6 I           4   221
##  7 P           8   221
##  8 F           2   220
##  9 R           3   220
## 10 S           7   220
## # ... with 250 more rows
```

```
data %>%
  count(letter, number) %>%
  ungroup() %>%
  arrange(desc(n))
```

```
## # A tibble: 260 x 3
##    letter number     n
##    <chr>   <int> <int>
##  1 E           8   233
##  2 F           5   229
##  3 T           9   227
##  4 Q           9   224
##  5 J           4   222
##  6 I           4   221
##  7 P           8   221
##  8 F           2   220
##  9 R           3   220
## 10 S           7   220
## # ... with 250 more rows
```

```
library(nycflights13)
library(tidyverse)

# Find the worst members of each group
flights %>%
  group_by(year, month, day) %>%
  filter(rank(desc(arr_delay)) < 4)
```

```
## # A tibble: 1,105 x 19
## # Groups:   year, month, day [365]
##     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      848           1835       853     1001           1950
##  2  2013     1     1     1815           1325       290     2120           1542
##  3  2013     1     1     2343           1724       379      314           1938
##  4  2013     1     2     1412            838       334     1710           1147
##  5  2013     1     2     1607           1030       337     2003           1355
##  6  2013     1     2     2131           1512       379     2340           1741
##  7  2013     1     3     2008           1540       268     2339           1909
##  8  2013     1     3     2012           1600       252     2314           1857
##  9  2013     1     3     2056           1605       291     2239           1754
## 10  2013     1     4     1305           1030       155     1452           1210
## # ... with 1,095 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>
```

```
(r1 <- rank(x1 <- c(3, 1, 4, 15, 92)))
```

```
## [1] 2 1 3 4 5
```

```
(r2 <- min_rank(x1 <- c(3, 1, 4, 15, 92)))
```

```
## [1] 2 1 3 4 5
```

```
# Find all groups bigger than a threshold
(poular_dests <-
  flights %>%
  group_by(dest) %>%
  filter(n() > 365))
```

```
## # A tibble: 332,577 x 19
## # Groups:   dest [77]
##     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 332,567 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>
```

```
# A grouped filter is a grouped mutate followed by an ungrouped filter. I generally avoid them except f

poular_dests %>%
  filter(arr_delay > 0) %>%
  mutate(prop_delay = arr_delay / sum(arr_delay)) %>%
  select(year:day, dest, arr_delay, prop_delay)
```

```
## # A tibble: 131,106 x 6
## # Groups:   dest [77]
##     year month   day dest  arr_delay prop_delay
##    <int> <int> <int> <chr>     <dbl>      <dbl>
##  1  2013     1     1 IAH          11  0.000111
##  2  2013     1     1 IAH          20  0.000201
##  3  2013     1     1 MIA          33  0.000235
##  4  2013     1     1 ORD          12  0.0000424
##  5  2013     1     1 FLL          19  0.0000938
##  6  2013     1     1 ORD           8  0.0000283
##  7  2013     1     1 LAX           7  0.0000344
##  8  2013     1     1 DFW          31  0.000282
##  9  2013     1     1 ATL          12  0.0000400
## 10  2013     1     1 DTW          16  0.000116
## # ... with 131,096 more rows
```

```
# 1)
# Filter function is appalied to each group and shrinks the elements of each group
flights %>%
  group_by(year, month, day) %>%
  filter (air_time == 320 & carrier=="US")
```

```
## # A tibble: 15 x 19
## # Groups:   year, month, day [15]
##     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    11     1619           1620        -1     2003           2003
##  2  2013     1    14     1346           1350        -4     1724           1715
##  3  2013    10    30      625            630        -5      931            918
##  4  2013    11    17     1010           1015        -5     1406           1342
##  5  2013    11    22     1639           1630         9     2022           2011
##  6  2013    11    23     1030           1030         0     1417           1410
##  7  2013    12     4     1352           1355        -3     1729           1715
##  8  2013    12     5     1402           1355         7     1736           1715
##  9  2013    12     6     1406           1355        11     1746           1715
## 10  2013    12    20     1631           1630         1     2009           1957
## 11  2013     2    23      954            959        -5     1345           1333
## 12  2013     3    23     1007           1015        -8     1252           1240
## 13  2013     4     1      623            630        -7      922            913
## 14  2013     4     2     1628           1635        -7     1902           1856
## 15  2013     4    26     1632           1630         2     1910           1851
## # ... 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>
```

```r
# mutate function with group
# Arithmetic operators with group_by
# flights %>%
```

```r
#Functions that work most naturally in grouped mutates and filters are known as window functions (vs. t

vignette("window-functions")
```

```
## starting httpd help server ... done
```