Programming for Data Analytics

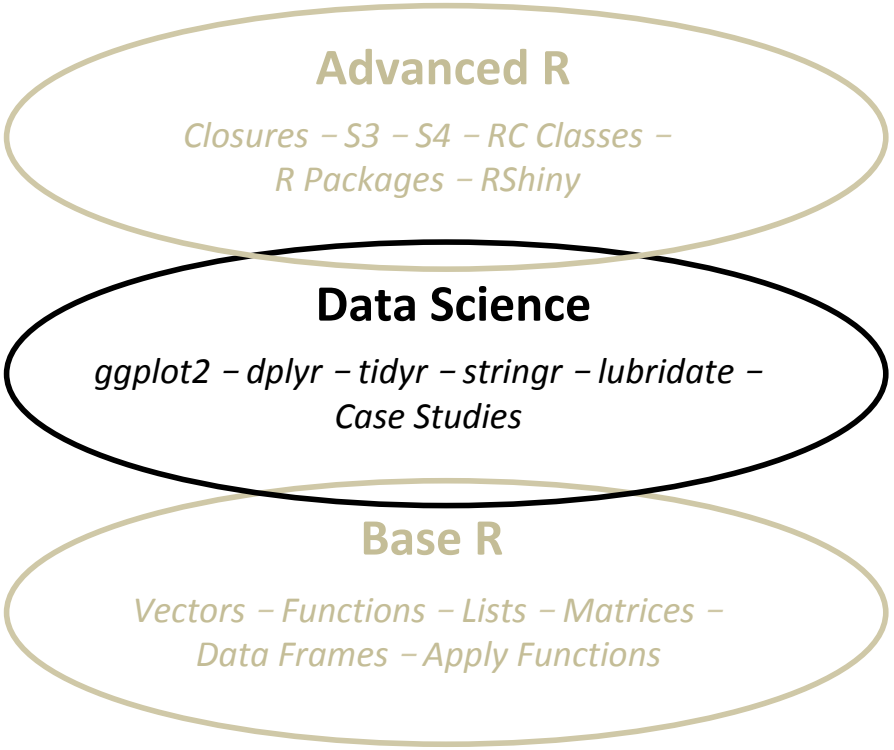
6. Data Transformation with dplyr

Prof. Jim Duggan,
School of Computer Science
National University of Ireland Galway.

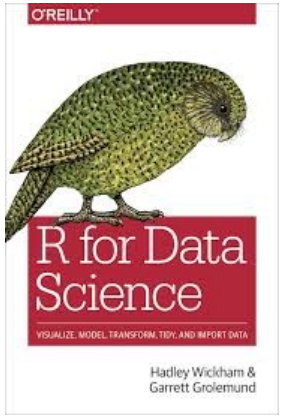
https://github.com/JimDuggan/CT5102

Lecture Overview

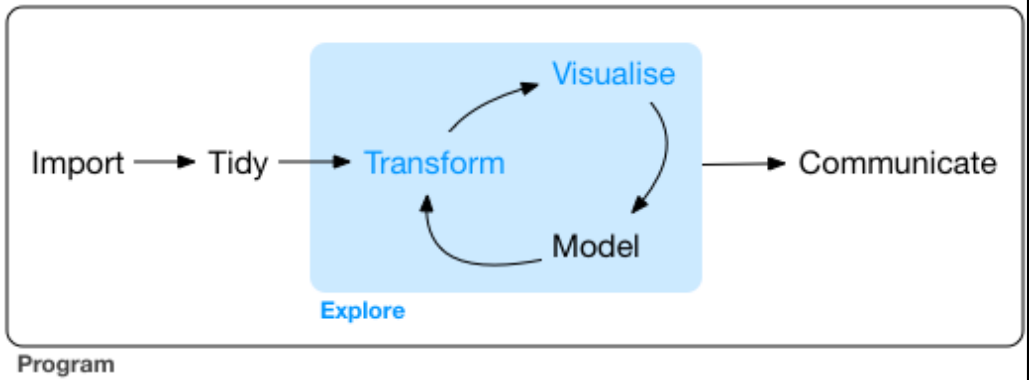
- filter()
- arrange()
- select()
- mutate()
- summarise()
- pull()
- case_when()



Overview



- Visualisation is an important tool for insight generation, but it's rare that you get the data in exactly the right form you need (Wickham and Grolemund 2017)
  - Create new variables
  - Create summaries
  - Order data
- **dplyr** package is designed for data transformation



dplyr Basics: 5 key functions

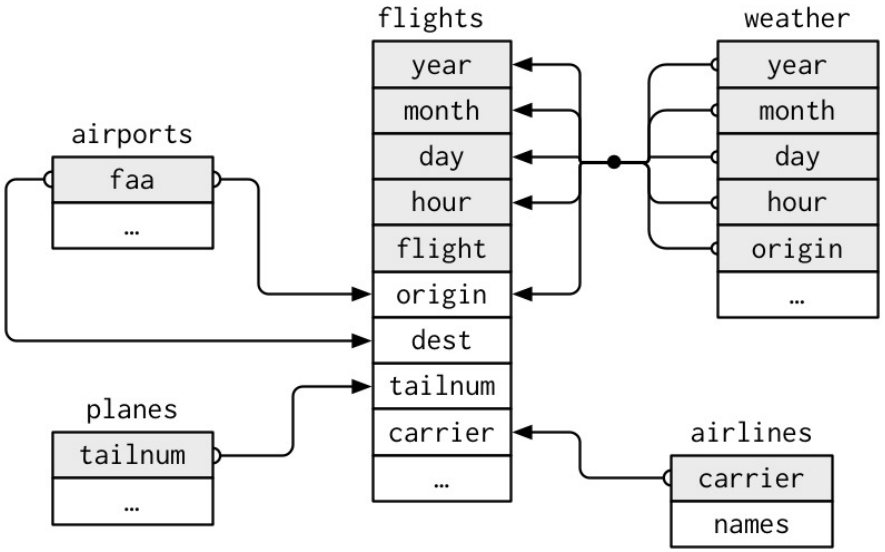
Function	Purpose
filter()	Pick observations by their values
arrange()	Reorder the rows
select()	Pick variables by their names
mutate()	Create new variables with functions of existing variables
summarise()	Collapse many values down to a single summary

- ”A grammar of data manipulation”
- All verbs (functions) work similarly
  - The first argument is a data frame/tibble
  - The subsequent arguments decide what to do with the data frame/tibble
  - The result (data frame/tibble) supports chaining of steps – NOTE the “pipe operator” which we will cover later.

https://dplyr.tidyverse.org

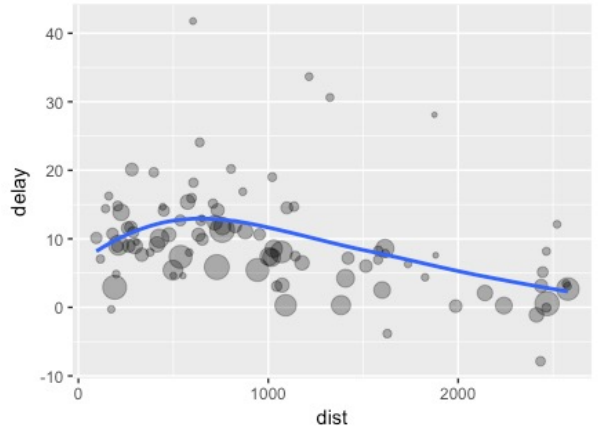
Airports Case Study

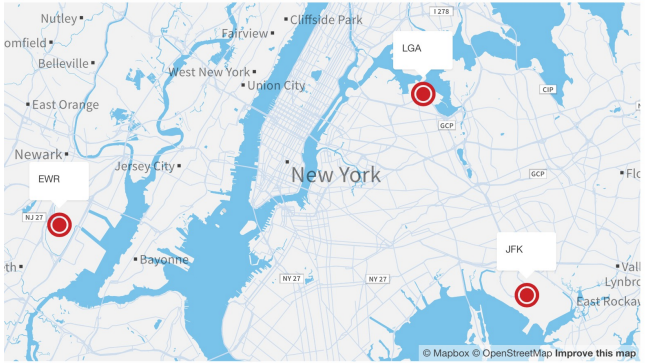
- flights connects to planes via a single variable, **tailnum**.
- flights connects to airlines through the **carrier** variable.
- flights connects to airports in two ways: via the **origin** and **dest** variables.
- flights connects to weather via **origin** (the location), and **year, month, day and hour** (the time).



Data Set: nycflights13

```
> flights
# A tibble: 336,776 × 19
    year month   day dep_time sched_dep_time dep_delay
   <int> <int> <int>    <int>          <int>     <dbl>
1   2013     1     1      517            515         2
2   2013     1     1      533            529         4
3   2013     1     1      542            540         2
4   2013     1     1      544            545        -1
5   2013     1     1      554            600        -6
6   2013     1     1      554            558        -4
7   2013     1     1      555            600        -5
8   2013     1     1      557            600        -3
9   2013     1     1      557            600        -3
10  2013     1     1      558            600        -2
# ... with 336,766 more rows, and 13 more variables:
#   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>
```





1. filter()

- Subset observations based on their values.
- First argument the name of the data frame
- Subsequent arguments are expressions that filter the data frame

```
> filter(flights,month==1,day==1)
# A tibble: 842 × 19
    year month   day dep_time sched_dep_time
   <int> <int> <int>    <int>          <int>
1   2013     1     1      517            515
2   2013     1     1      533            529
3   2013     1     1      542            540
4   2013     1     1      544            545
5   2013     1     1      554            600
6   2013     1     1      554            558
7   2013     1     1      555            600
8   2013     1     1      557            600
9   2013     1     1      557            600
10  2013     1     1      558            600
# ... with 832 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>
```

Relational operators in R

Operators	Description
<	less than
<=	less than or equal to
>	greater than
>=	greater than or equal to
==	exactly equal to
!=	not equal to
!x	not x
x | y	x OR y
x & y	x AND y

```
> bel <- filter(observations,station=="BELMULLET")
> bel
# A tibble: 8,760 x 12
   station  year month   day  hour date                 rain
   <chr>   <dbl> <dbl> <int> <int> <dttm>              <dbl>
 1 BELMUL… 2017     1     1     0 2017-01-01 00:00:00   0
 2 BELMUL… 2017     1     1     1 2017-01-01 01:00:00   0.5
 3 BELMUL… 2017     1     1     2 2017-01-01 02:00:00   0
 4 BELMUL… 2017     1     1     3 2017-01-01 03:00:00   0.4
 5 BELMUL… 2017     1     1     4 2017-01-01 04:00:00   0.6
 6 BELMUL… 2017     1     1     5 2017-01-01 05:00:00   0.1
 7 BELMUL… 2017     1     1     6 2017-01-01 06:00:00   0
 8 BELMUL… 2017     1     1     7 2017-01-01 07:00:00   0
 9 BELMUL… 2017     1     1     8 2017-01-01 08:00:00   0
10 BELMUL… 2017     1     1     9 2017-01-01 09:00:00   0
# … with 8,750 more rows, and 5 more variables: temp <dbl>,
#   rhum <dbl>, msl <dbl>, wdsp <dbl>, wddir <dbl>
```

Earliest Flight(s) that left all year?

```
> filter(flights,dep_time == min(dep_time,na.rm = T))
# A tibble: 25 × 19
    year month   day dep_time sched_dep_time dep_delay arr_time
   <int> <int> <int>    <int>          <int>     <dbl>    <int>
1   2013     1    13        1           2249        72      108
2   2013     1    31        1           2100       181      124
3   2013    11    13        1           2359         2      442
4   2013    12    16        1           2359         2      447
5   2013    12    20        1           2359         2      430
6   2013    12    26        1           2359         2      437
7   2013    12    30        1           2359         2      441
8   2013     2    11        1           2100       181      111
9   2013     2    24        1           2245        76      121
10  2013     3     8        1           2355         6      431
# ... with 15 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>
```

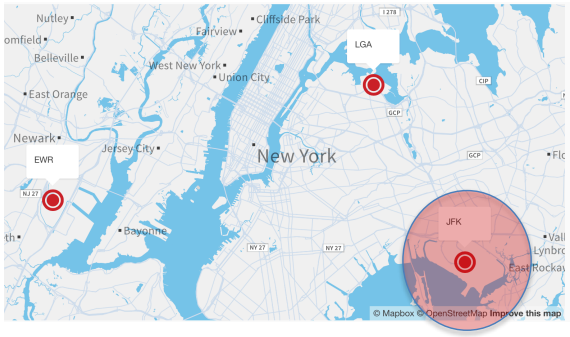
Flights with longest dep delay?

```
> filter(flights,dep_delay == max(dep_delay,na.rm = T))
# A tibble: 1 × 19
   year month   day dep_time sched_dep_time dep_delay arr_time
  <int> <int> <int>    <int>          <int>     <dbl>    <int>
1  2013     1     9      641            900      1301     1242
# ... with 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>
```

```
> x<-filter(flights,dep_delay == max(dep_delay,na.rm = T))
```

More details…

```
> glimpse(x)
Observations: 1
Variables: 19
$ year           <int> 2013
$ month          <int> 1
$ day            <int> 9
$ dep_time       <int> 641
$ sched_dep_time <int> 900
$ dep_delay      <dbl> 1301
$ arr_time       <int> 1242
$ sched_arr_time <int> 1530
$ arr_delay      <dbl> 1272
$ carrier        <chr> "HA"
$ flight         <int> 51
$ tailnum        <chr> "N384HA"
$ origin         <chr> "JFK"
$ dest           <chr> "HNL"
$ air_time       <dbl> 640
$ distance       <dbl> 4983
$ hour           <dbl> 9
$ minute         <dbl> 0
$ time_hour      <dttm> 2013-01-09 09:00:00
```




```
> flights
# A tibble: 336,776 x 19
    year month   day dep_time sched_dep_time dep_delay arr_time sched_arr_time
   <int> <int> <int>    <int>          <int>     <dbl>    <int>          <int>
 1  2013     1     1      517            515         2      830            819
 2  2013     1     1      533            529         4      850            830
 3  2013     1     1      542            540         2      923            850
 4  2013     1     1      544            545        -1     1004           1022
 5  2013     1     1      554            600        -6      812            837
 6  2013     1     1      554            558        -4      740            728
 7  2013     1     1      555            600        -5      913            854
 8  2013     1     1      557            600        -3      709            723
 9  2013     1     1      557            600        -3      838            846
10  2013     1     1      558            600        -2      753            745
# … with 336,766 more rows, and 11 more variables: arr_delay <dbl>, carrier <chr>,
#   flight <int>, tailnum <chr>, origin <chr>, dest <chr>, air_time <dbl>,
#   distance <dbl>, hour <dbl>, minute <dbl>, time_hour <dttm>
```

Challenge 6.1

- Find all flights that:
  - Had an arrival delay of two or more hours
  - Flew to Houston (IAH or HOU)
  - Departed in the summer (July, August and September)
  - Arrived more than 2 hours late, but didn't leave late

2. arrange()

Function	Purpose
filter()	Pick observations by their values
arrange()	Reorder the rows
select()	Pick variables by their names
mutate()	Create new variables with functions of existing variables
summarise()	Collapse many values down to a single summary

- Changes the order of rows.
- Takes a data frame and a set of column names to order by

```
> arrange(flights,dep_delay)
# A tibble: 336,776 x 19
    year month   day dep_time sched_dep_time dep_delay arr_time
   <int> <int> <int>    <int>          <int>     <dbl>    <int>
1   2013    12     7     2040           2123       -43       40
2   2013     2     3     2022           2055       -33     2240
3   2013    11    10     1408           1440       -32     1549
```

Using desc()

```
> arrange(flights,desc(dep_delay))
# A tibble: 336,776 × 19
    year month   day dep_time sched_dep_time dep_delay arr_time
   <int> <int> <int>    <int>          <int>     <dbl>    <int>
1   2013     1     9      641            900      1301     1242
2   2013     6    15     1432           1935      1137     1607
3   2013     1    10     1121           1635      1126     1239
4   2013     9    20     1139           1845      1014     1457
5   2013     7    22      845           1600      1005     1044
6   2013     4    10     1100           1900       960     1342
7   2013     3    17     2321            810       911      135
```

Mean Sea Level Pressure

```
> arrange(observations,msl)
# A tibble: 219,000 x 12
   station              year month   day  hour date                rain  temp  rhum   msl  wdsp wddir
   <chr>               <dbl> <dbl> <int> <int> <dttm>              <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
 1 VALENTIA OBSERVATORY 2017    10    16    11 2017-10-16 11:00:00   9.8  14.6    95  962.    24   100
 2 BELMULLET            2017     2     2    20 2017-02-02 20:00:00   2.5   9.4    94  964.    25   140
 3 BELMULLET            2017     2     2    19 2017-02-02 19:00:00   0     9.3    89  964.    15   140
 4 BELMULLET            2017     2     2    18 2017-02-02 18:00:00   0.1   9.4    87  965.    17   140
 5 MACE HEAD            2017     2     2    15 2017-02-02 15:00:00   0.2  10.1    86  965.    23   120
 6 BELMULLET            2017     2     2    17 2017-02-02 17:00:00   0.3   9.6    88  965     18   140
 7 MACE HEAD            2017     2     2    16 2017-02-02 16:00:00   0.4   9.7    90  965.    19   140
 8 MACE HEAD            2017     2     2    17 2017-02-02 17:00:00   0.2   9.5    90  965.    17   140
 9 BELMULLET            2017     2     2    16 2017-02-02 16:00:00   0    10.6    79  965.    18   140
10 MACE HEAD            2017     2     2    14 2017-02-02 14:00:00   0    10.8    82  966.    22   120
# … with 218,990 more rows
```

Humidity

```
> arrange(observations,rhum)
# A tibble: 219,000 x 12
   station        year month   day  hour date                rain  temp  rhum   msl  wdsp wddir
   <chr>         <dbl> <dbl> <int> <int> <dttm>              <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
 1 SherkinIsland  2017    11    23     5 2017-11-23 05:00:00   0     8.6    20  991.    29   260
 2 SherkinIsland  2017    11    28    13 2017-11-28 13:00:00   0     7.9    20 1019.    11   320
 3 SherkinIsland  2017    11    28    14 2017-11-28 14:00:00   0     8.1    20 1018.    11   330
 4 SherkinIsland  2017    11    18    23 2017-11-18 23:00:00   0    11.9    21 1024.    11   260
 5 SherkinIsland  2017    11    19     5 2017-11-19 05:00:00   0    11.5    21 1024.     8   260
 6 SherkinIsland  2017    11    19     7 2017-11-19 07:00:00   0    10.4    21 1024.     4   220
 7 SherkinIsland  2017    11    21     8 2017-11-21 08:00:00   1.4  12.8    21 1006.    20   200
 8 SherkinIsland  2017    11    22     1 2017-11-22 01:00:00   2.5  12.8    21  995.    19   210
 9 SherkinIsland  2017    11    23    18 2017-11-23 18:00:00   0     8.2    21 1005.     6    10
10 SherkinIsland  2017    11    24    15 2017-11-24 15:00:00   0     6.1    21 1015.     8   320
# … with 218,990 more rows
```

More than one value

```
> arrange(observations,month,temp)
# A tibble: 219,000 x 12
   station     year month   day  hour date                rain  temp  rhum   msl  wdsp wddir
   <chr>      <dbl> <dbl> <int> <int> <dttm>              <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
 1 MOORE PARK  2017     1     3     9 2017-01-03 09:00:00     0  -5.6    91 1033.     1   330
 2 MOORE PARK  2017     1     3     8 2017-01-03 08:00:00     0  -5.4    91 1033.     1   160
 3 MARKREE     2017     1    23     4 2017-01-23 04:00:00     0  -5.1    96 1024.    NA    NA
 4 MOORE PARK  2017     1     3     7 2017-01-03 07:00:00     0  -5.1    92 1033.     1   250
 5 MARKREE     2017     1    23     5 2017-01-23 05:00:00     0  -5      98 1024.    NA    NA
 6 MARKREE     2017     1    23     2 2017-01-23 02:00:00     0  -4.8    97 1025.    NA    NA
 7 MARKREE     2017     1    23     3 2017-01-23 03:00:00     0  -4.8    98 1025.    NA    NA
 8 MOORE PARK  2017     1     3     6 2017-01-03 06:00:00     0  -4.8    92 1033.     1   270
 9 MT DILLON   2017     1    21     8 2017-01-21 08:00:00     0  -4.6    96 1027.     2   350
10 MARKREE     2017     1    23     1 2017-01-23 01:00:00     0  -4.4    96 1026.    NA    NA
# … with 218,990 more rows
```

In descending order - desc()

```
> arrange(observations,desc(temp))
# A tibble: 219,000 x 12
   station       year month   day  hour date                rain  temp  rhum   msl  wdsp wddir
   <chr>        <dbl> <dbl> <int> <int> <dttm>              <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
 1 PHOENIX PARK  2017     6    21    13 2017-06-21 13:00:00   0.1  28.3    51 1010     NA    NA
 2 PHOENIX PARK  2017     6    21    12 2017-06-21 12:00:00   0    27.5    54 1011.    NA    NA
 3 PHOENIX PARK  2017     6    21    14 2017-06-21 14:00:00   0    27.5    49 1010.    NA    NA
 4 PHOENIX PARK  2017     6    21    16 2017-06-21 16:00:00   0    26.8    61 1009.    NA    NA
 5 CASEMENT      2017     6    21    12 2017-06-21 12:00:00   0    26.6    54 1011.    11   150
 6 MOORE PARK    2017     6    19    16 2017-06-19 16:00:00   0    26.6    50 1018.     3   200
 7 DUNSANY       2017     6    21    12 2017-06-21 12:00:00   0    26.5    55 1010.     8   150
 8 PHOENIX PARK  2017     6    21    11 2017-06-21 11:00:00   0    26.5    56 1011.    NA    NA
 9 PHOENIX PARK  2017     6    17    16 2017-06-17 16:00:00   0    26.4    42 1024.    NA    NA
10 PHOENIX PARK  2017     6    21    15 2017-06-21 15:00:00   0    26.4    61 1009.    NA    NA
# … with 218,990 more rows
```

Mean Sea Level Pressure

```
> arrange(observations,desc(msl))
# A tibble: 219,000 x 12
   station              year month   day  hour date                rain  temp  rhum   msl  wdsp wddir
   <chr>               <dbl> <dbl> <int> <int> <dttm>              <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
 1 VALENTIA OBSERVATORY 2017    12    22    19 2017-12-22 19:00:00   0     9.7    97 1039.    NA    NA
 2 VALENTIA OBSERVATORY 2017    12    22    18 2017-12-22 18:00:00   0     9.9    98 1039     NA    NA
 3 VALENTIA OBSERVATORY 2017    12    22    11 2017-12-22 11:00:00   0    10.3    97 1039.    NA    NA
 4 VALENTIA OBSERVATORY 2017    12    22    20 2017-12-22 20:00:00   0.2   9.5    98 1039.    NA    NA
 5 VALENTIA OBSERVATORY 2017    12    22    21 2017-12-22 21:00:00   0.2   9.5    97 1039.    NA    NA
 6 CORK AIRPORT         2017    12    22    21 2017-12-22 21:00:00   0     8.9   100 1039.     4   260
 7 CORK AIRPORT         2017    12    22    20 2017-12-22 20:00:00   0     9.4    99 1039.     3   290
 8 SherkinIsland        2017    12    22    19 2017-12-22 19:00:00   0     9      95 1039.     6   250
 9 SherkinIsland        2017    12    22    20 2017-12-22 20:00:00   0.1   9      96 1039.     3   280
10 VALENTIA OBSERVATORY 2017    12    22    12 2017-12-22 12:00:00   0    10.4    98 1039.    NA    NA
# … with 218,990 more rows
```

Windspeed

```
> arrange(observations,desc(wdsp))
# A tibble: 219,000 x 12
   station        year month   day  hour date                rain  temp  rhum   msl  wdsp wddir
   <chr>         <dbl> <dbl> <int> <int> <dttm>              <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
 1 ROCHES POINT   2017    10    16    12 2017-10-16 12:00:00   1.3  12      96  983.    59   180
 2 ROCHES POINT   2017    10    16    11 2017-10-16 11:00:00   0.2  11.7    88  983.    55   160
 3 SherkinIsland  2017    10    16    11 2017-10-16 11:00:00   0    13.4    92  975.    52   170
 4 MACE HEAD      2017     2    23     2 2017-02-23 02:00:00   0     7.6    86  985.    50   250
 5 ROCHES POINT   2017    10    16    13 2017-10-16 13:00:00   1    12.9    98  986.    50   190
 6 MACE HEAD      2017     2    23     3 2017-02-23 03:00:00   0     7      84  987     48   270
 7 MALIN HEAD     2017    12    31     7 2017-12-31 07:00:00   0.1   7      84  974.    48   250
 8 SherkinIsland  2017    10    16    10 2017-10-16 10:00:00   0.7  11.4    97  974.    47   150
 9 MACE HEAD      2017     2    23     4 2017-02-23 04:00:00   0     7.2    86  990.    46   290
10 MACE HEAD      2017    12    31     2 2017-12-31 02:00:00   0     8.2    78  979.    46   240
# … with 218,990 more rows
```

Challenge 6.2

- Which 10 flights travelled the longest?
- Which 10 flights traveled the shortest?

3. select()

Function	Purpose
filter()	Pick observations by their values
arrange()	Reorder the rows
select()	Pick variables by their names
mutate()	Create new variables with functions of existing variables
summarise()	Collapse many values down to a single summary

- It is not uncommon to get datasets with hundreds, or even thousands, of variables
- A challenge is to narrow down on the variables of you're interested in
- select() allows you to rapidly zoom in on a useful subset using operations based on the variable names
- Number of rows does not change

```
> new_obs <- select(observations,station,year,month,day,hour,temp)
> new_obs
# A tibble: 219,000 x 6
   station  year month   day  hour  temp
   <chr>   <dbl> <dbl> <int> <int> <dbl>
 1 ATHENRY  2017     1     1     0   5.2
 2 ATHENRY  2017     1     1     1   4.7
 3 ATHENRY  2017     1     1     2   4.2
 4 ATHENRY  2017     1     1     3   3.5
 5 ATHENRY  2017     1     1     4   3.2
 6 ATHENRY  2017     1     1     5   2.1
 7 ATHENRY  2017     1     1     6   2
 8 ATHENRY  2017     1     1     7   1.7
 9 ATHENRY  2017     1     1     8   1
10 ATHENRY  2017     1     1     9   1.1
# … with 218,990 more rows
```

Useful options with select()

```
> select(observations,station:rain)
# A tibble: 219,000 x 7
   station  year month   day  hour date                rain
   <chr>   <dbl> <dbl> <int> <int> <dttm>              <dbl>
 1 ATHENRY  2017     1     1     0 2017-01-01 00:00:00   0
 2 ATHENRY  2017     1     1     1 2017-01-01 01:00:00   0
 3 ATHENRY  2017     1     1     2 2017-01-01 02:00:00   0
 4 ATHENRY  2017     1     1     3 2017-01-01 03:00:00   0.1
 5 ATHENRY  2017     1     1     4 2017-01-01 04:00:00   0.1
 6 ATHENRY  2017     1     1     5 2017-01-01 05:00:00   0
 7 ATHENRY  2017     1     1     6 2017-01-01 06:00:00   0
 8 ATHENRY  2017     1     1     7 2017-01-01 07:00:00   0
 9 ATHENRY  2017     1     1     8 2017-01-01 08:00:00   0
10 ATHENRY  2017     1     1     9 2017-01-01 09:00:00   0
# … with 218,990 more rows
```

```
> select(observations,-(station:rain))
# A tibble: 219,000 x 5
    temp  rhum   msl  wdsp wddir
   <dbl> <dbl> <dbl> <dbl> <dbl>
 1   5.2    89 1022.     8   320
 2   4.7    89 1022      9   320
 3   4.2    90 1022.     8   320
 4   3.5    87 1022.     9   330
 5   3.2    89 1023.     8   330
 6   2.1    91 1023.     8   330
 7   2      89 1024.     7   330
 8   1.7    89 1024.     7   340
 9   1      91 1025      7   330
10   1.1    91 1026.     8   330
# … with 218,990 more rows
```

Special functions with select()

Special functions

As well as using existing functions like `:` and `c`, there are a number of special functions that only work inside `select`

- `starts_with(x, ignore.case = TRUE)`: names starts with `x`
- `ends_with(x, ignore.case = TRUE)`: names ends in `x`
- `contains(x, ignore.case = TRUE)`: selects all variables whose name contains `x`
- `matches(x, ignore.case = TRUE)`: selects all variables whose name matches the regular expression `x`
- `num_range("x", 1:5, width = 2)`: selects all variables (numerically) from x01 to x05.
- `one_of("x", "y", "z")`: selects variables provided in a character vector.
- `everything()`: selects all variables.

Examples

```
> select(observations,starts_with("w"))
# A tibble: 219,000 x 2
    wdsp wddir
   <dbl> <dbl>
 1     8   320
 2     9   320
 3     8   320
 4     9   330
 5     8   330
 6     8   330
 7     7   330
 8     7   340
 9     7   330
10     8   330
# … with 218,990 more rows
```

```
> select(observations,ends_with("p"))
# A tibble: 219,000 x 2
    temp  wdsp
   <dbl> <dbl>
 1   5.2     8
 2   4.7     9
 3   4.2     8
 4   3.5     9
 5   3.2     8
 6   2.1     8
 7   2       7
 8   1.7     7
 9   1       7
10   1.1     8
# … with 218,990 more rows
```

everything()

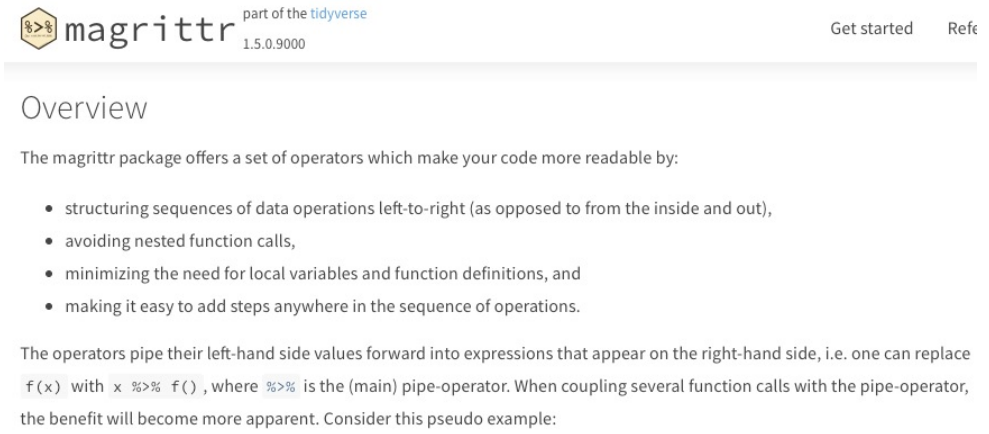
```
> select(observations,ends_with("p"),everything())
# A tibble: 219,000 x 12
    temp  wdsp station  year month   day  hour date                rain  rhum   msl wddir
   <dbl> <dbl> <chr>   <dbl> <dbl> <int> <int> <dttm>              <dbl> <dbl> <dbl> <dbl>
 1   5.2     8 ATHENRY  2017     1     1     0 2017-01-01 00:00:00   0      89 1022.   320
 2   4.7     9 ATHENRY  2017     1     1     1 2017-01-01 01:00:00   0      89 1022    320
 3   4.2     8 ATHENRY  2017     1     1     2 2017-01-01 02:00:00   0      90 1022.   320
 4   3.5     9 ATHENRY  2017     1     1     3 2017-01-01 03:00:00   0.1    87 1022.   330
 5   3.2     8 ATHENRY  2017     1     1     4 2017-01-01 04:00:00   0.1    89 1023.   330
 6   2.1     8 ATHENRY  2017     1     1     5 2017-01-01 05:00:00   0      91 1023.   330
 7   2       7 ATHENRY  2017     1     1     6 2017-01-01 06:00:00   0      89 1024.   330
 8   1.7     7 ATHENRY  2017     1     1     7 2017-01-01 07:00:00   0      89 1024.   340
 9   1       7 ATHENRY  2017     1     1     8 2017-01-01 08:00:00   0      91 1025    330
10   1.1     8 ATHENRY  2017     1     1     9 2017-01-01 09:00:00   0      91 1026.   330
# … with 218,990 more rows
```

Challenge 6.3

- Select all columns from flights that contains the substring "time"

Combining operations with the Pipe

- The pipe %>% comes from the magrittr package (Stefan Milton Bache)
- Helps to write code that is easier to read and understand
- x %>% f(y) turns into f(x, y)
- x %>% f(y) %>% g(z) turns into g(f(x, y),z)



https://magrittr.tidyverse.org

```
> sqrt(1:5)
[1] 1.000000 1.414214 1.732051 2.000000 2.236068
> 1:5 %>% sqrt()
[1] 1.000000 1.414214 1.732051 2.000000 2.236068
```

Examples

```
> observations %>% filter(day==1,station=="ATHENRY",hour==12,month==1)
# A tibble: 1 x 12
  station  year month   day  hour date                rain  temp  rhum   msl
  <chr>   <dbl> <dbl> <int> <int> <dttm>             <dbl> <dbl> <dbl> <dbl>
1 ATHENRY  2017     1     1    12 2017-01-01 12:00:00     0   5.1    75 1027.
# … with 2 more variables: wdsp <dbl>, wddir <dbl>
> observations %>% filter(station=="MACE HEAD") %>% arrange(desc(temp)) %>% head()
# A tibble: 6 x 12
  station  year month   day  hour date                rain  temp  rhum   msl
  <chr>   <dbl> <dbl> <int> <int> <dttm>             <dbl> <dbl> <dbl> <dbl>
1 MACE H…  2017     6    20    17 2017-06-20 17:00:00     0  22.7    69 1015.
2 MACE H…  2017     6    20    16 2017-06-20 16:00:00     0  22.6    67 1016.
3 MACE H…  2017     6    20    18 2017-06-20 18:00:00     0  22.3    71 1015.
4 MACE H…  2017     7    18    16 2017-07-18 16:00:00     0  22.3    61 1008.
5 MACE H…  2017     7    18    18 2017-07-18 18:00:00     0  22.2    65 1007.
6 MACE H…  2017     6    20    15 2017-06-20 15:00:00     0  22.1    68 1017.
# … with 2 more variables: wdsp <dbl>, wddir <dbl>
```

Challenge 6.4

- Organise the following into a pipeline command
  - Subset all observations from aimsir17 from October 2017
  - Select all those from “ROCHES POINT”
  - Sort the observations by wind speed (descending)
  - Select the top five (see the dplyr function slice)
  - Store in a result tibble

4. mutate()

Function	Purpose
filter()	Pick observations by their values
arrange()	Reorder the rows
select()	Pick variables by their names
mutate()	Create new variables with functions of existing variables
summarise()	Collapse many values down to a single summary

- It is often useful to add new columns that are functions of existing columns
- mutate() always adds new columns at the end of your data set.

```
sml <- select(flights,
              year:day,
              ends_with("delay"),
              distance,
              air_time)
```

Calculate any gain during flight…

```
> mutate(sml,gain=dep_delay-arr_delay)
# A tibble: 336,776 × 8
    year month   day dep_delay arr_delay distance air_time  gain
   <int> <int> <int>     <dbl>     <dbl>    <dbl>    <dbl> <dbl>
1   2013     1     1         2        11     1400      227    -9
2   2013     1     1         4        20     1416      227   -16
3   2013     1     1         2        33     1089      160   -31
4   2013     1     1        -1       -18     1576      183    17
5   2013     1     1        -6       -25      762      116    19
6   2013     1     1        -4        12      719      150   -16
7   2013     1     1        -5        19     1065      158   -24
8   2013     1     1        -3       -14      229       53    11
9   2013     1     1        -3        -8      944      140     5
10  2013     1     1        -2         8      733      138   -10
# ... with 336,766 more rows
```

Useful Creation Functions

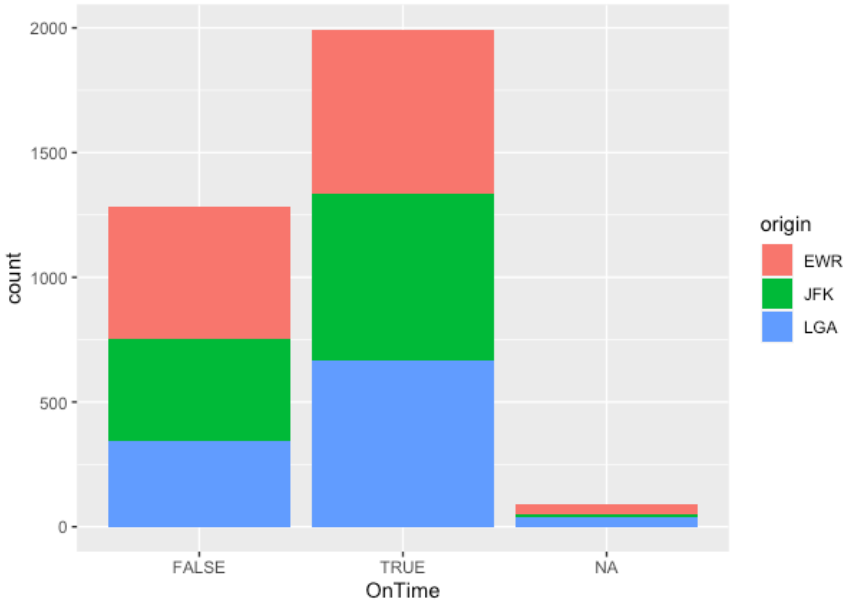
- There are many functions for creating new variables that can be used with mutate()
- The key property is that the function **must be vectorised:**
  - It must take a vector of values as input, and,
  - Return a vector with the same number of values as output
- Useful functions are now summarised

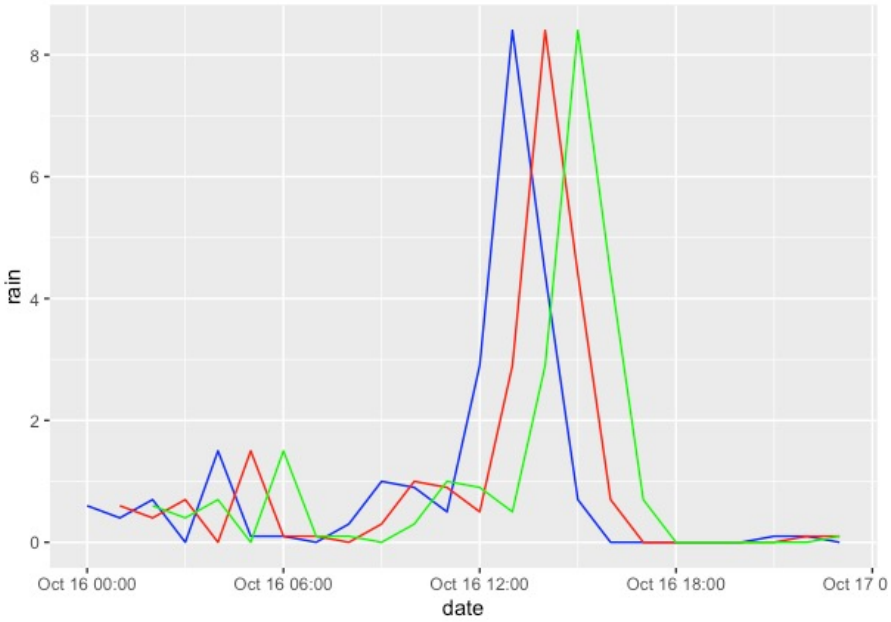
Useful Vectorised Functions

Grouping	Examples
Arithmetic Operators	+, -, *, /, ^
Modular Arithmetic	%/% - Integer division && - Remainder
Logs	log(), log2(), log10()
Offsets	lead() and lag() Find when values change x!=lag(x)
Cumulative and rolling aggregates	cumsum(), cumprod(), cummin(), cummax(), cummean()
Logical comparisons	<, <=, >, >=, !=
Ranking	min_rank()

Challenge 6.5

- Use mutate to generate the following graphs (Mace Head is the station for plot 2)





5. summarise()

- The last key verb is summarise()
- It collapses a data frame into a single row
- Not very useful unless paired with group_by()
- Very useful to combine with the pipe operator

```
> summarise(flights,
+           AvrDelay=mean(dep_delay,na.rm=TRUE))
# A tibble: 1 × 1
  AvrDelay
     <dbl>
1 12.63907
```

group_by()

- Most data operations are useful done on groups defined by variables in the the dataset.
- The group_by function takes an existing tbl and converts it into a grouped tbl where operations are performed "by group".

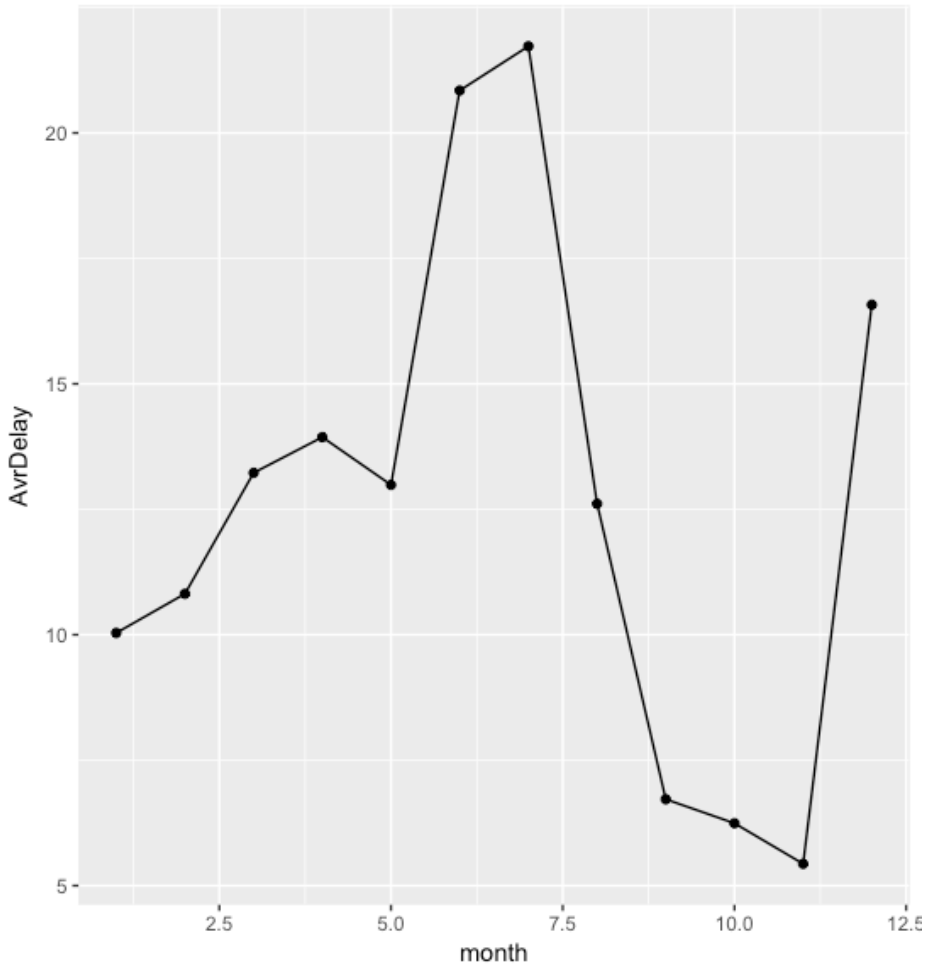
```
by_month <- group_by(flights,month)

ans <- summarise(by_month,
                 AvrDelay=mean(dep_delay,na.rm=T))
```

```
> by_month
Source: local data frame [336,776 x 19]
Groups: month [12]

    year month   day dep_time sched_dep_time dep_delay arr_time
   <int> <int> <int>    <int>          <int>     <dbl>    <int>
1   2013     1     1      517            515         2      830
2   2013     1     1      533            529         4      850
3   2013     1     1      542            540         2      923
4   2013     1     1      544            545        -1     1004
5   2013     1     1      554            600        -6      812
6   2013     1     1      554            558        -4      740
7   2013     1     1      555            600        -5      913
8   2013     1     1      557            600        -3      709
9   2013     1     1      557            600        -3      838
10  2013     1     1      558            600        -2      753
# ... with 336,766 more rows, and 12 more variables:
#   sched_arr_time <int>, arr_delay <dbl>, carrier <chr>,
#   flight <int>, tailnum <chr>, origin <chr>, dest <chr>,
#   air_time <dbl>, distance <dbl>, hour <dbl>, minute <dbl>,
#   time_hour <dttm>
```

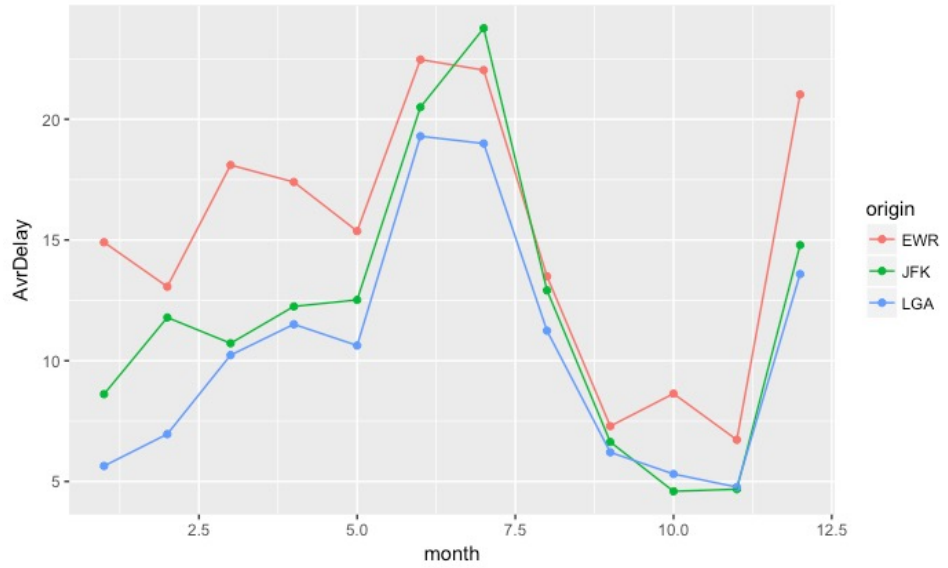
```
> ans
# A tibble: 12 × 2
   month  AvrDelay
   <int>     <dbl>
1      1 10.036665
2      2 10.816843
3      3 13.227076
4      4 13.938038
5      5 12.986859
6      6 20.846332
7      7 21.727787
8      8 12.611040
9      9  6.722476
10    10  6.243988
11    11  5.435362
12    12 16.576688
```



```
by_month <- group_by(flights,month,origin)

ans <- summarise(by_month,AvrDelay=mean(dep_delay,na.rm=T))

ggplot(ans,mapping=aes(x=month,y=AvrDelay,colour=origin))+
        geom_point()+geom_path()
```



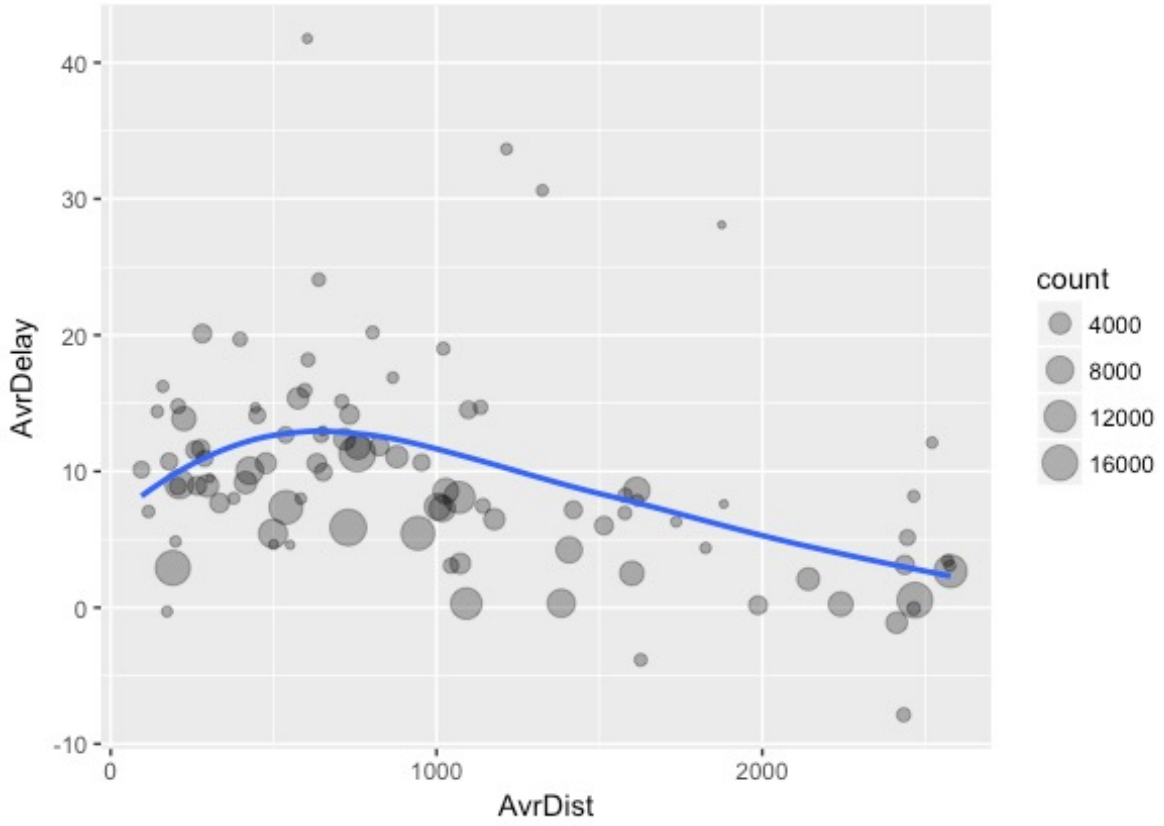
Example

- We want to explore the relationship between distance and average delay for each destination

```
delay <- flights %>% group_by(dest) %>%
           summarize(count=n(),
                     AvrDist=mean(distance,na.rm=T),
                     AvrDelay=mean(arr_delay,na.rm=T)) %>%
           arrange(dest) %>% filter(count>20,dest!="HNL")
```

```
> delay
# A tibble: 96 × 4
    dest count   AvrDist  AvrDelay
   <chr> <int>     <dbl>     <dbl>
1    ABQ   254 1826.0000  4.381890
2    ACK   265  199.0000  4.852273
3    ALB   439  143.0000 14.397129
4    ATL 17215  757.1082 11.300113
5    AUS  2439 1514.2530  6.019909
6    AVL   275  583.5818  8.003831
7    BDL   443  116.0000  7.048544
8    BGR   375  378.0000  8.027933
9    BHM   297  865.9966 16.877323
10   BNA  6333  758.2135 11.812459
# ... with 86 more rows
```

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

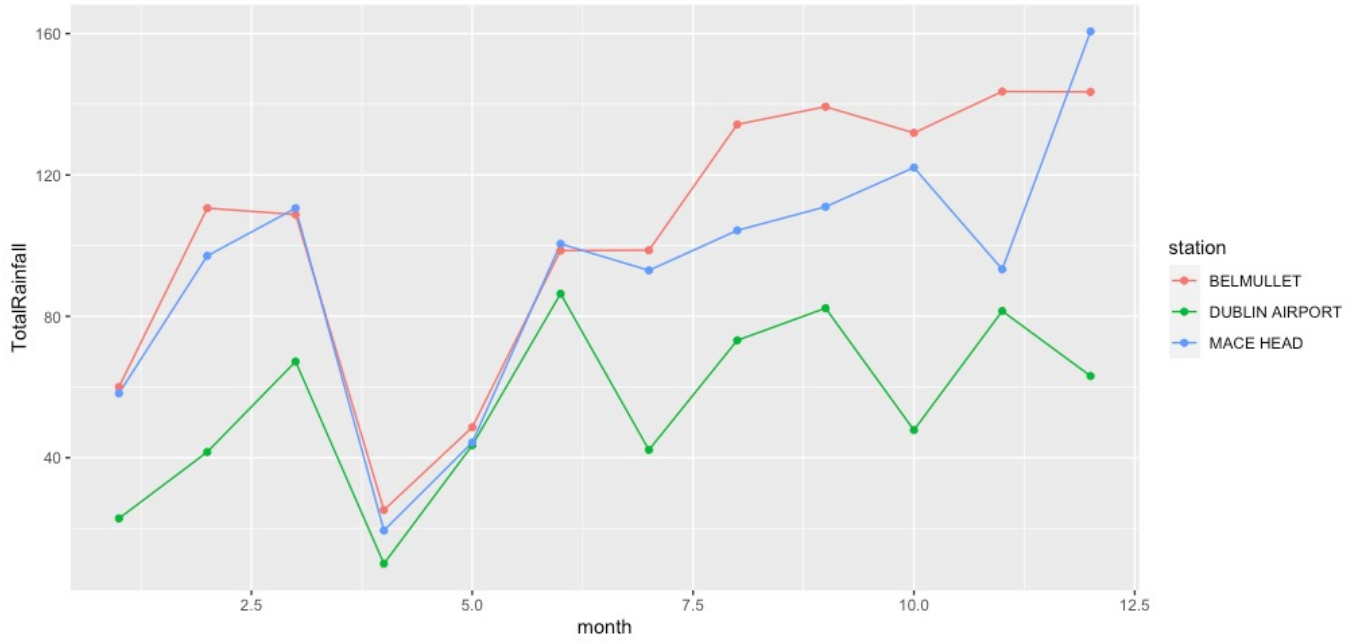


Useful Summary Functions

Grouping	Examples
Measures of location	mean(), median()
Measures of spread	sd(), IQR(),mad()
Measures of rank	min((), quantile(), max()
Measures of position	first(), nth(), last()
Counts	n(), n_distinct()
Counts and proportions of logical values	sum(x>0) when used with numeric functions, (T,F) converted to (1,0)

Challenge 6.6

- *Generate the following summary plot*



pull() – Select 1 column as a vector

```
> select(mpg,hwy)
# A tibble: 234 x 1
     hwy
   <int>
 1    29
 2    29
 3    31
 4    30
 5    26
 6    26
 7    27
 8    26
 9    25
10    28
# ... with 224 more rows
```

```
> pull(mpg,hwy)
  [1] 29 29 31 30 26 26 27 26 25 28 27 25
 [13] 25 25 25 24 25 23 20 15 20 17 17 26
 [25] 23 26 25 24 19 14 15 17 27 30 26 29
 [37] 26 24 24 22 22 24 24 17 22 21 23 23
 [49] 19 18 17 17 19 19 12 17 15 17 17 12
 [61] 17 16 18 15 16 12 17 17 16 12 15 16
 [73] 17 15 17 17 18 17 19 17 19 19 17 17
 [85] 17 16 16 17 15 17 26 25 26 24 21 22
 [97] 23 22 20 33 32 32 29 32 34 36 36 29
[109] 26 27 30 31 26 26 28 26 29 28 27 24
[121] 24 24 22 19 20 17 12 19 18 14 15 18
[133] 18 15 17 16 18 17 19 19 17 29 27 31
[145] 32 27 26 26 25 25 17 17 20 18 26 26
[157] 27 28 25 25 24 27 25 26 23 26 26 26
[169] 26 25 27 25 27 20 20 19 17 20 17 29
```

pull() – Also works with column numbers, default is last column

```
> head(pull(mpg))
[1] "compact" "compact" "compact" "compact"
[5] "compact" "compact"
>
> head(pull(mpg,1))
[1] "audi" "audi" "audi" "audi" "audi"
[6] "audi"
>
> head(pull(mpg,-1))
[1] "compact" "compact" "compact" "compact"
[5] "compact" "compact"
```

case_when()

- This function allows you to vectorise multiple if and else if statements. It is an R equivalent of the SQL CASE WHEN statement. Arguments:
  - A sequence of two-sided formulas. The left hand side (LHS) determines which values match this case. The right hand side (RHS) provides the replacement value.
  - The LHS must evaluate to a logical vector. Each logical vector can either have length 1 or a common length. All RHSs must evaluate to the same type of vector.
  - Proceed from the most specific to the most general

Very useful in mutate()

```
starwars %>%
  select(name:mass, gender, species) %>%
  mutate(
    type = case_when(
      height > 200 | mass > 200 ~ "large",
      species == "Droid"        ~ "robot",
      TRUE                      ~  "other"
    )
  )
```

```
# A tibble: 87 x 6
                  name height  mass gender species  type
                 <chr>  <int> <dbl>  <chr>   <chr> <chr>
1     Luke Skywalker    172    77   male   Human other
2              C-3PO    167    75   <NA>   Droid robot
```