데이터 과학 외전

Day 1 – 데이터 핸들링 & 데이터 시각화

Contents

Data Manipulation

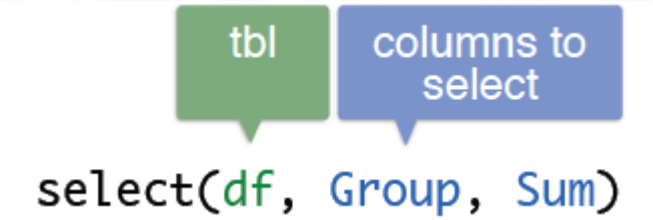
- Data science is to extract information that supports to make a critical decision
  - List up several questions that are important to answer
  - In many cases, manipulation and summarization are sufficient to get answers

Data to Practice

```
load(url('https://github.com/hbchoi/SampleData/raw/master/hflights.RData'))
```

hf_data data frame에는 2011년 미국 내 국내선 중 Houston 두 개 공항 IAH (George Bush Intercontinental) and HOU (Houston Hobby) 에서 출발한 비행 편에 대한 정보를 담고 있다. 전체 227,496편의 비행편에 대한 데이터를 담고 있으며, 각 비행편 당 21개의 변수를 담고 있다.

select




```r
# Print out a tbl with the four columns of hf_data related to delay
select(hf_data, ActualElapsedTime, AirTime, ArrDelay, DepDelay)
```

```
## # A tibble: 227,496 × 4
##    ActualElapsedTime AirTime ArrDelay DepDelay
## *             <int>   <int>    <int>    <int>
## 1                60      40      -10        0
## 2                60      45       -9        1
## 3                70      48       -8       -8
## 4                70      39        3        3
## 5                62      44       -3        5
## 6                64      45       -7       -1
## 7                70      43       -1       -1
## 8                59      40      -16       -5
## 9                71      41       44       43
## 10               70      45       43       43
## # ... with 227,486 more rows
```

```r
# Print out the columns Origin up to Cancelled of hf_data
select(hf_data, Origin:Cancelled)
```

```
## # A tibble: 227,496 × 6
##    Origin  Dest Distance TaxiIn TaxiOut Cancelled
## *   <chr> <chr>    <int>  <int>   <int>     <int>
## 1     IAH   DFW      224      7      13         0
## 2     IAH   DFW      224      6       9         0
## 3     IAH   DFW      224      5      17         0
## 4     IAH   DFW      224      9      22         0
## 5     IAH   DFW      224      9       9         0
## 6     IAH   DFW      224      6      13         0
## 7     IAH   DFW      224     12      15         0
## 8     IAH   DFW      224      7      12         0
## 9     IAH   DFW      224      8      22         0
## 10    IAH   DFW      224      6      19         0
## # ... with 227,486 more rows
```

```r
# Find the most concise way to select: columns Year up to and including
DayOfWeek, columns ArrDelay up to and including Diverted.
select(hf_data, Year:DayOfWeek, ArrDelay:Diverted)
```

```
## # A tibble: 227,496 × 14
##     Year Month DayofMonth DayOfWeek ArrDelay DepDelay Origin  Dest
## *  <int> <int>      <int>     <int>    <int>    <int>  <chr> <chr>
## 1   2011     1          1         6      -10        0    IAH   DFW
## 2   2011     1          2         7       -9        1    IAH   DFW
## 3   2011     1          3         1       -8       -8    IAH   DFW
## 4   2011     1          4         2        3        3    IAH   DFW
## 5   2011     1          5         3       -3        5    IAH   DFW
## 6   2011     1          6         4       -7       -1    IAH   DFW
## 7   2011     1          7         5       -1       -1    IAH   DFW
## 8   2011     1          8         6      -16       -5    IAH   DFW
## 9   2011     1          9         7       44       43    IAH   DFW
## 10  2011     1         10         1       43       43    IAH   DFW
## # ... with 227,486 more rows, and 6 more variables: Distance <int>,
## #   TaxiIn <int>, TaxiOut <int>, Cancelled <int>, CancellationCode <chr>,
## #   Diverted <int>
```

Helper functions for variable selection

- starts_with("X"): every name that starts with "X",
- ends_with("X"): every name that ends with "X",
- contains("X"): every name that contains "X",
- matches("X"): every name that matches "X", where "X"can be a regular expression,
- num_range("x", 1:5): the variables named x01, x02, x03, x04 and x05,
- one_of(x): every name that appears in x, which should be a character vector

```r
# Print out a tbl containing just ArrDelay and DepDelay
select(hf_data, ends_with("Delay"))
```

```
## # A tibble: 227,496 × 2
##    ArrDelay DepDelay
## *     <int>    <int>
## 1       -10        0
## 2        -9        1
## 3        -8       -8
## 4         3        3
## 5        -3        5
## 6        -7       -1
## 7        -1       -1
## 8       -16       -5
## 9        44       43
## 10       43       43
## # ... with 227,486 more rows
```

Helper functions for variable selection

- starts_with("X"): every name that starts with "X",
- ends_with("X"): every name that ends with "X",
- contains("X"): every name that contains "X",
- matches("X"): every name that matches "X", where "X"can be a regular expression,
- num_range("x", 1:5): the variables named x01, x02, x03, x04 and x05,
- one_of(x): every name that appears in x, which should be a character vector

```
select(hf_data, UniqueCarrier, ends_with("Num"), starts_with("Cancell"))
```

```
## # A tibble: 227,496 × 5
##    UniqueCarrier FlightNum TailNum Cancelled CancellationCode
## *          <chr>     <int>   <chr>     <int>            <chr>
## 1       American       428  N576AA         0             <NA>
## 2       American       428  N557AA         0             <NA>
## 3       American       428  N541AA         0             <NA>
## 4       American       428  N403AA         0             <NA>
## 5       American       428  N492AA         0             <NA>
## 6       American       428  N262AA         0             <NA>
## 7       American       428  N493AA         0             <NA>
## 8       American       428  N477AA         0             <NA>
## 9       American       428  N476AA         0             <NA>
## 10      American       428  N504AA         0             <NA>
## # ... with 227,486 more rows
```

```
select(hf_data, ends_with("Time"), ends_with("Delay"))
```

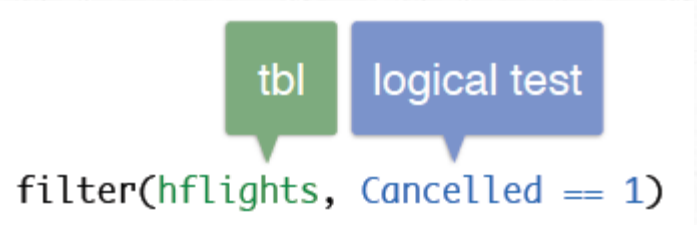
```
## # A tibble: 227,496 × 6
##    DepTime ArrTime ActualElapsedTime AirTime ArrDelay DepDelay
## *    <int>   <int>             <int>   <int>    <int>    <int>
## 1     1400    1500                60      40      -10        0
## 2     1401    1501                60      45       -9        1
## 3     1352    1502                70      48       -8       -8
## 4     1403    1513                70      39        3        3
## 5     1405    1507                62      44       -3        5
## 6     1359    1503                64      45       -7       -1
## 7     1359    1509                70      43       -1       -1
## 8     1355    1454                59      40      -16       -5
## 9     1443    1554                71      41       44       43
## 10    1443    1553                70      45       43       43
## # ... with 227,486 more rows
```

mutate


```r
# Add the new variable ActualGroundTime, the difference between
ActualElapsedTime and AirTime
g1 <- mutate(hf_data, ActualGroundTime = ActualElapsedTime - AirTime)

# Add the new variable GroundTime to g1. This column is the sum of the TaxiIn
and TaxiOut columns.
g2 <- mutate(g1, GroundTime = TaxiIn + TaxiOut)

# Add the new variable AverageSpeed to g2 that denotes the average speed that
each plane flew in miles per hour.
g3 <- mutate(g2, AverageSpeed = Distance / AirTime * 60)


# Print out g3
g3
```

```
## # A tibble: 227,496 × 24
##     Year Month DayofMonth DayOfWeek DepTime ArrTime UniqueCarrier
##    <int> <int>      <int>     <int>   <int>   <int>         <chr>
## 1   2011     1          1         6    1400    1500      American
## 2   2011     1          2         7    1401    1501      American
## 3   2011     1          3         1    1352    1502      American
## 4   2011     1          4         2    1403    1513      American
## 5   2011     1          5         3    1405    1507      American
## 6   2011     1          6         4    1359    1503      American
## 7   2011     1          7         5    1359    1509      American
## 8   2011     1          8         6    1355    1454      American
## 9   2011     1          9         7    1443    1554      American
## 10  2011     1         10         1    1443    1553      American
## # ... with 227,486 more rows, and 17 more variables: FlightNum <int>,
## #   TailNum <chr>, ActualElapsedTime <int>, AirTime <int>, ArrDelay <int>,
## #   DepDelay <int>, Origin <chr>, Dest <chr>, Distance <int>,
## #   TaxiIn <int>, TaxiOut <int>, Cancelled <int>, CancellationCode <chr>,
## #   Diverted <int>, ActualGroundTime <int>, GroundTime <int>,
## #   AverageSpeed <dbl>
```

filter




```r
# All flights flown by one of JetBlue, Southwest, or Delta
filter(hf_data, UniqueCarrier %in% c("JetBlue", "Southwest", "Delta"))
```

```
## # A tibble: 48,679 × 21
##     Year Month DayofMonth DayOfWeek DepTime ArrTime UniqueCarrier
##    <int> <int>      <int>     <int>   <int>   <int>         <chr>
## 1   2011     1          1         6     654    1124       JetBlue
## 2   2011     1          1         6    1639    2110       JetBlue
## 3   2011     1          2         7     703    1113       JetBlue
## 4   2011     1          2         7    1604    2040       JetBlue
## 5   2011     1          3         1     659    1100       JetBlue
## 6   2011     1          3         1    1801    2200       JetBlue
## 7   2011     1          4         2     654    1103       JetBlue
## 8   2011     1          4         2    1608    2034       JetBlue
## 9   2011     1          5         3     700    1103       JetBlue
## 10  2011     1          5         3    1544    1954       JetBlue
## # ... with 48,669 more rows, and 14 more variables: FlightNum <int>,
## #   TailNum <chr>, ActualElapsedTime <int>, AirTime <int>, ArrDelay <int>,
## #   DepDelay <int>, Origin <chr>, Dest <chr>, Distance <int>,
## #   TaxiIn <int>, TaxiOut <int>, Cancelled <int>, CancellationCode <chr>,
## #   Diverted <int>
```

```r
# All flights that traveled 3000 miles or more
filter(hf_data, Distance >= 3000)
```

```
## # A tibble: 527 × 21
##     Year Month DayofMonth DayOfWeek DepTime ArrTime UniqueCarrier
##    <int> <int>      <int>     <int>   <int>   <int>         <chr>
## 1   2011     1         31         1     924    1413   Continental
## 2   2011     1         30         7     925    1410   Continental
## 3   2011     1         29         6    1045    1445   Continental
## 4   2011     1         28         5    1516    1916   Continental
## 5   2011     1         27         4     950    1344   Continental
## 6   2011     1         26         3     944    1350   Continental
## 7   2011     1         25         2     924    1337   Continental
## 8   2011     1         24         1    1144    1605   Continental
## 9   2011     1         23         7     926    1335   Continental
## 10  2011     1         22         6     942    1340   Continental
## # ... with 517 more rows, and 14 more variables: FlightNum <int>,
## #   TailNum <chr>, ActualElapsedTime <int>, AirTime <int>, ArrDelay <int>,
## #   DepDelay <int>, Origin <chr>, Dest <chr>, Distance <int>,
## #   TaxiIn <int>, TaxiOut <int>, Cancelled <int>, CancellationCode <chr>,
## #   Diverted <int>
```

arrange

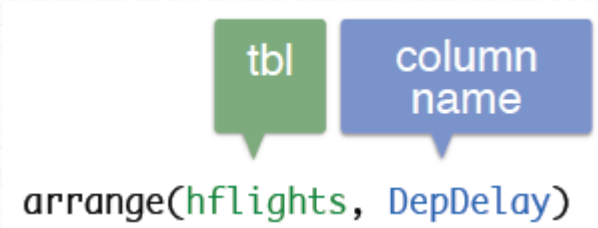

```r
# Arrange dtc so that cancellation reasons are grouped
arrange(dtc, CancellationCode)
```

```
## # A tibble: 68 × 21
##     Year Month DayofMonth DayOfWeek DepTime ArrTime  UniqueCarrier
##    <int> <int>      <int>     <int>   <int>   <int>          <chr>
## 1   2011     1         20         4    1413      NA         United
## 2   2011     1          7         5    2028      NA     ExpressJet
## 3   2011     2          4         5    1638      NA       American
## 4   2011     2          8         2    1057      NA    Continental
## 5   2011     2          1         2    1508      NA        SkyWest
## 6   2011     2         21         1    2257      NA        SkyWest
## 7   2011     2          9         3     555      NA American_Eagle
## 8   2011     3         18         5     727      NA         United
## 9   2011     4          4         1    1632      NA          Delta
## 10  2011     4          8         5    1608      NA      Southwest
## # ... with 58 more rows, and 14 more variables: FlightNum <int>,
## #   TailNum <chr>, ActualElapsedTime <int>, AirTime <int>, ArrDelay <int>,
## #   DepDelay <int>, Origin <chr>, Dest <chr>, Distance <int>,
## #   TaxiIn <int>, TaxiOut <int>, Cancelled <int>, CancellationCode <chr>,
## #   Diverted <int>
```

```r
# Definition of dtc
dtc <- filter(hf_data, Cancelled == 1, !is.na(DepDelay))

# Arrange dtc by departure delays
arrange(dtc, DepDelay)
```

```
## # A tibble: 68 × 21
##     Year Month DayofMonth DayOfWeek DepTime ArrTime  UniqueCarrier
##    <int> <int>      <int>     <int>   <int>   <int>          <chr>
## 1   2011     7         23         6     605      NA       Frontier
## 2   2011     1         17         1     916      NA     ExpressJet
## 3   2011    12          1         4     541      NA     US_Airways
## 4   2011    10         12         3    2022      NA American_Eagle
## 5   2011     7         29         5    1424      NA    Continental
## 6   2011     9         29         4    1639      NA        SkyWest
## 7   2011     2          9         3     555      NA American_Eagle
## 8   2011     5          9         1     715      NA        SkyWest
## 9   2011     1         20         4    1413      NA         United
## 10  2011     1         17         1     831      NA      Southwest
## # ... with 58 more rows, and 14 more variables: FlightNum <int>,
## #   TailNum <chr>, ActualElapsedTime <int>, AirTime <int>, ArrDelay <int>,
## #   DepDelay <int>, Origin <chr>, Dest <chr>, Distance <int>,
## #   TaxiIn <int>, TaxiOut <int>, Cancelled <int>, CancellationCode <chr>,
## #   Diverted <int>
```

arrange



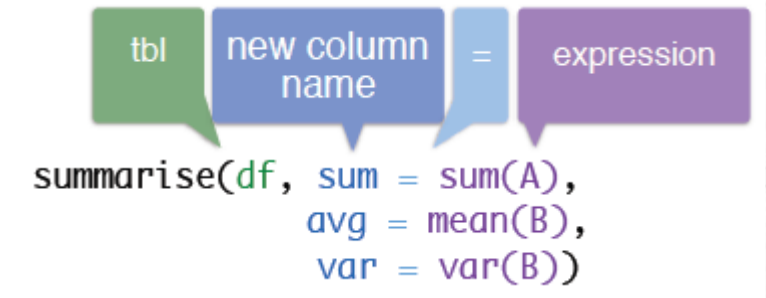

```r
# Arrange dtc according to carrier and departure delays
arrange(dtc, UniqueCarrier, DepDelay)
```

```
## # A tibble: 68 × 21
##     Year Month DayofMonth DayOfWeek DepTime ArrTime      UniqueCarrier
##    <int> <int>      <int>     <int>   <int>   <int>              <chr>
## 1   2011     6         11         6    1649      NA            AirTran
## 2   2011     8         18         4    1808      NA           American
## 3   2011     2          4         5    1638      NA           American
## 4   2011    10         12         3    2022      NA     American_Eagle
## 5   2011     2          9         3     555      NA     American_Eagle
## 6   2011     7         17         7    1917      NA     American_Eagle
## 7   2011     4         30         6     612      NA Atlantic_Southeast
## 8   2011     4         10         7    1147      NA Atlantic_Southeast
## 9   2011     5         23         1     657      NA Atlantic_Southeast
## 10  2011     9         29         4     723      NA Atlantic_Southeast
## # ... with 58 more rows, and 14 more variables: FlightNum <int>,
## #   TailNum <chr>, ActualElapsedTime <int>, AirTime <int>, ArrDelay <int>,
## #   DepDelay <int>, Origin <chr>, Dest <chr>, Distance <int>,
## #   TaxiIn <int>, TaxiOut <int>, Cancelled <int>, CancellationCode <chr>,
## #   Diverted <int>
```

```r
# Arrange according to carrier and decreasing departure delays
arrange(hf_data, UniqueCarrier, desc(DepDelay))
```

```
## # A tibble: 227,496 × 21
##     Year Month DayofMonth DayOfWeek DepTime ArrTime UniqueCarrier
##    <int> <int>      <int>     <int>   <int>   <int>          <chr>
## 1   2011     2         19         6    1902    2143       AirTran
## 2   2011     3         14         1    2024    2309       AirTran
## 3   2011     2         16         3    2349     227       AirTran
## 4   2011    11         13         7    2312     213       AirTran
## 5   2011     5         26         4    2353     305       AirTran
## 6   2011     5         26         4    1922    2229       AirTran
## 7   2011     4         28         4    1045    1328       AirTran
## 8   2011     6          5         7    2207      52       AirTran
## 9   2011     5          7         6    1009    1256       AirTran
## 10  2011     7         25         1    2107      14       AirTran
## # ... with 227,486 more rows, and 14 more variables: FlightNum <int>,
## #   TailNum <chr>, ActualElapsedTime <int>, AirTime <int>, ArrDelay <int>,
## #   DepDelay <int>, Origin <chr>, Dest <chr>, Distance <int>,
## #   TaxiIn <int>, TaxiOut <int>, Cancelled <int>, CancellationCode <chr>,
## #   Diverted <int>
```

summarise



Aggregate functions

- `min(x)` - minimum value of vector `x`.
- `max(x)` - maximum value of vector `x`.
- `mean(x)` - mean value of vector `x`.
- `median(x)` - median value of vector `x`.
- `quantile(x, p)` - `p`th quantile of vector `x`.
- `sd(x)` - standard deviation of vector `x`.
- `var(x)` - variance of vector `x`.
- `IQR(x)` - Inter Quartile Range (IQR) of vector `x`.

- `diff(range(x))` - total range of vector `x`.

dplyr aggregate functions

- `first(x)` - The first element of vector `x`.
- `last(x)` - The last element of vector `x`.
- `nth(x, n)` - The `n`th element of vector `x`.
- `n()` - The number of rows in the data.frame or group of observations that `summarise()` describes.

- `n_distinct(x)` - The number of unique values in vector `x`.

```r
summarise(hf_data, min_dist = min(Distance), max_dist = max(Distance))
```

```
## # A tibble: 1 × 2
##   min_dist max_dist
##      <int>    <int>
## 1       79     3904
```

```r
# Print out a summary with variable max_div: the longest Distance for
diverted flights.
summarise(filter(hf_data, Diverted == 1), max_div = max(Distance))
```

```
## # A tibble: 1 × 1
##   max_div
##     <int>
## 1    3904
```

```r
# Remove rows that have NA ArrDelay: temp1
temp1 <- filter(hf_data, !is.na(ArrDelay))

# Generate summary about ArrDelay column of temp1
summarise(temp1, earliest = min(ArrDelay), average = mean(ArrDelay), latest=
max(ArrDelay), sd = sd(ArrDelay))
```

```
## # A tibble: 1 × 4
##   earliest  average latest       sd
##      <int>    <dbl>  <int>    <dbl>
## 1      -70 7.094334    978 30.70852
```

```r
# Keep rows that have no NA TaxiIn and no NA TaxiOut: temp2
temp2 <- filter(hf_data, !is.na(TaxiIn), !is.na(TaxiOut))

# Print the maximum taxiing difference of temp2 with summarise()
summarise(temp2, max_taxi_diff = max(abs(TaxiIn - TaxiOut)))
```

```
## # A tibble: 1 × 1
##   max_taxi_diff
##           <int>
## 1           160
```

```r
# Generate summarizing statistics for hf_data
summarise(hf_data,
          n_obs = n(),
          n_carrier = n_distinct(UniqueCarrier),
          n_dest = n_distinct(Dest))
```

```
## # A tibble: 1 × 3
##    n_obs n_carrier n_dest
##    <int>     <int>  <int>
## 1 227496        15    116
```

```r
# All American Airline flights
aa <- filter(hf_data, UniqueCarrier == "American")

# Generate summarizing statistics for aa
summarise(aa,
          n_flights = n(),
          n_canc = sum(Cancelled),
          avg_delay = mean(ArrDelay, na.rm = T))
```

```
## # A tibble: 1 × 3
##   n_flights n_canc avg_delay
##       <int>  <int>     <dbl>
## 1      3244     60 0.8917558
```

pipe operation %>%

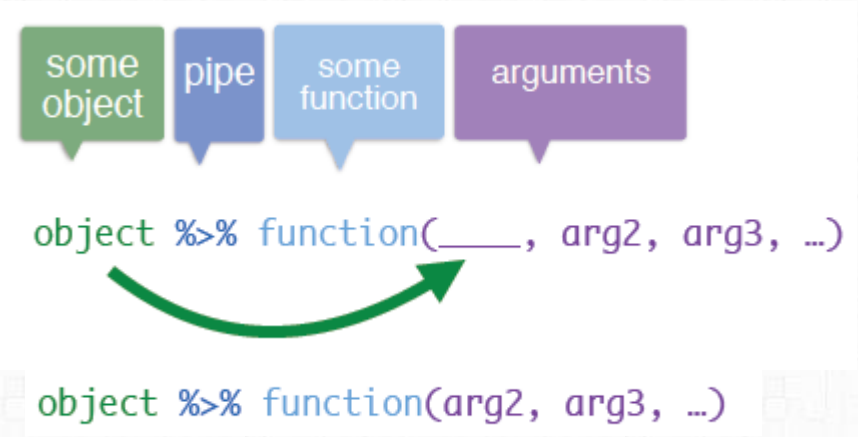
Option 1

```
a1 <- select(a, X, Y, Z)
a2 <- filter(a1, X > Y)
a3 <- mutate(a2, Q = X + Y + Z)
a4 <- summarise(a3, all = sum(Q))
```

Option 2

```
summarise(
    mutate(
        filter(
            select(a, X, Y, Z),
        X > Y),
        Q = X + Y + Z),
    all = sum(Q))
)
```

pipe operation %>%




```
a %>%
  select(X, Y, Z) %>%
  filter(X > Y) %>%
  mutate(Q = X + Y + Z) %>%
  summarise(all = sum(Q))
```

```
hf_data %>%
  mutate(diff = TaxiOut - TaxiIn) %>%
  filter(!is.na(diff)) %>%
  summarise(avg = mean(diff))
```

```
## # A tibble: 1 × 1
##        avg
##      <dbl>
## 1 8.992064
```

Pipe operator

```
> x <- 1:10

> x %>% sum()
[1] 55
```

```
> sum(x)
[1] 55
```

```
> abs(diff(range(x)))
[1] 9

> x %>%
>  range() %>%
>  diff() %>%
>  abs()
[1] 9
```

group_by

tbl

column to group by

```
group_by(df, Group)

df %>%
  group_by(Group)
```

```
hf_data %>% group_by(UniqueCarrier)

## Source: local data frame [227,496 x 21]
## Groups: UniqueCarrier [15]
##
##     Year Month DayofMonth DayOfWeek DepTime ArrTime UniqueCarrier
## *  <int> <int>      <int>     <int>   <int>   <int>         <chr>
## 1   2011     1          1         6    1400    1500      American
## 2   2011     1          2         7    1401    1501      American
## 3   2011     1          3         1    1352    1502      American
## 4   2011     1          4         2    1403    1513      American
## 5   2011     1          5         3    1405    1507      American
## 6   2011     1          6         4    1359    1503      American
## 7   2011     1          7         5    1359    1509      American
## 8   2011     1          8         6    1355    1454      American
## 9   2011     1          9         7    1443    1554      American
## 10  2011     1         10         1    1443    1553      American
## # ... with 227,486 more rows, and 14 more variables: FlightNum <int>,
## #   TailNum <chr>, ActualElapsedTime <int>, AirTime <int>, ArrDelay <int>,
## #   DepDelay <int>, Origin <chr>, Dest <chr>, Distance <int>,
## #   TaxiIn <int>, TaxiOut <int>, Cancelled <int>, CancellationCode <chr>,
## #   Diverted <int>
```

group_by

tbl

column to group by

```
group_by(df, Group)

df %>%
  group_by(Group)
```

```
hf_data %>%
  group_by(UniqueCarrier) %>%
  summarise(avgDep = mean(DepDelay, na.rm = T),
            avgArr = mean(ArrDelay, na.rm = T))
```

```
## # A tibble: 15 × 3
##          UniqueCarrier    avgDep     avgArr
##                  <chr>     <dbl>      <dbl>
## 1              AirTran  4.716376  1.8536239
## 2               Alaska  3.712329  3.1923077
## 3             American  6.390144  0.8917558
## 4       American_Eagle 11.071745  7.1529751
## 5   Atlantic_Southeast 12.482193  7.2569543
## 6          Continental  9.261313  6.0986983
## 7                Delta  9.370627  6.0841374
## 8            ExpressJet  7.713728  8.1865242
## 9              Frontier  5.093637  7.6682692
## 10              JetBlue 13.320532  9.8588410
## 11                 Mesa  1.538462  4.0128205
## 12              SkyWest  8.885482  8.6934922
## 13            Southwest 13.488241  7.5871430
## 14               United 12.918707 10.4628628
## 15           US_Airways  1.622926 -0.6307692
```

group_by

tbl

column to group by

```
group_by(df, Group)
df %>%
  group_by(Group)
```

```
hf_data %>%
  group_by(UniqueCarrier) %>%
  summarise(avgDep = mean(DepDelay, na.rm = T),
            avgArr = mean(ArrDelay, na.rm = T)) %>%
  arrange(avgArr, avgDep)
```

```
## # A tibble: 15 × 3
##          UniqueCarrier    avgDep     avgArr
##                  <chr>     <dbl>      <dbl>
## 1           US_Airways  1.622926 -0.6307692
## 2             American  6.390144  0.8917558
## 3              AirTran  4.716376  1.8536239
## 4               Alaska  3.712329  3.1923077
## 5                 Mesa  1.538462  4.0128205
## 6                Delta  9.370627  6.0841374
## 7          Continental  9.261313  6.0986983
## 8       American_Eagle 11.071745  7.1529751
## 9   Atlantic_Southeast 12.482193  7.2569543
## 10           Southwest 13.488241  7.5871430
## 11             Frontier  5.093637  7.6682692
## 12           ExpressJet  7.713728  8.1865242
## 13              SkyWest  8.885482  8.6934922
## 14              JetBlue 13.320532  9.8588410
## 15               United 12.918707 10.4628628
```

연습문제

① 각 항공사(unique carrier)별로 비행편수를 계산해서, 해당 기간동안 가장 많은 비행편수가 많은 항공사부터 정렬하여라. 가장 비행편을 많이 운행한 항공사는 어디이며, 가장 적게 운행한 항공사는 어디인가

② 각 목적지 별로(Dest) 운항시간(ActualElapsedTime)의 평균을 계산하여, 평균적으로 가장 오래걸리는 목적지는 어디이며 가장 빨리도착하는 목적지는 어디인지 찾아보자

③ 각 항공사별로 cancel이 된 항공편의 비율을 계산해보고, 비율이 높은 항공사부터 정렬하여라, cancel이 될 확률이 가장 높은 항공사는 어디이며 확률이 얼마나 되는가?

Data Integration

- Most of Real world data comes across multiple tables (or data frame)
- Data Integration is to combine the datasets into single data table for further processing
- It is easier to process and analyze data in a single table

Data Join

- Combines data into one data frame from two or more different data table

sid	dept	GPA
1000	ICT	4.1
1001	GE	3.9
1002	CSEE	3.5

sid	name	hometown
1001	KIM	Pohang
1002	LEE	Seoul
1000	PARK	Busan

sid	dept	GPA	name	hometown
1000	ICT	4.1	PARK	Busan
1001	GE	3.9	KIM	Pohang
1002	CSEE	3.5	LEE	Seoul

Join Key

Primary Table

sid	dept	GPA
1000	ICT	4.1
1001	GE	3.9
1002	CSEE	3.5

Primary Key

Secondary Table

sid	name	hometown
1001	KIM	Pohang
1002	LEE	Seoul
1000	PARK	Busan

Foreign Key

sid	dept	GPA	name	hometown
1000	ICT	4.1	PARK	Busan
1001	GE	3.9	KIM	Pohang
1002	CSEE	3.5	LEE	Seoul

Keys

primary key

foreign key

```
> names2
  name    surname    band
1 John   Coltrane      NA
2 John     Lennon Beatles
3 Paul McCartney Beatles
```

```
> plays2
   name    surname  plays
1  John     Lennon Guitar
2  Paul McCartney   Bass
3 Keith   Richards Guitar
```

```
# Example join output
   name    surname    band  plays
1  John   Coltrane    <NA>   <NA>
2  John     Lennon Beatles Guitar
3  Paul McCartney Beatles   Bass
4 Keith   Richards    <NA> Guitar
```

- Primary key should uniquely identify records
- Key may involves multiple variables

left_join

```
> names
  name     band
1 Mick   Stones
2 John Beatles
3 Paul Beatles
```

```
> plays
   name  plays
1  John Guitar
2  Paul   Bass
3 Keith Guitar
```

```
> left_join(names, plays, by = "name")
  name     band  plays
1 Mick   Stones   <NA>
2 John Beatles Guitar
3 Paul Beatles   Bass
```

- returns a copy of primary dataset with one or more data added to it from secondary dataset
- all rows from first dataset retains in the result dataset

right_join

```
> names
  name     band
1 Mick   Stones
2 John Beatles
3 Paul Beatles
```

```
> plays
   name  plays
1  John Guitar
2  Paul   Bass
3 Keith Guitar
```

```
> right_join(names, plays, by = "name")
   name     band  plays
1  John Beatles Guitar
2  Paul Beatles   Bass
3 Keith    <NA> Guitar
```

- returns a copy of secondary dataset with one or more data added to it from primary dataset
- all rows from second dataset retains in the result dataset

inner_join

```
> names
  name    band
1 Mick  Stones
2 John Beatles
3 Paul Beatles
```

```
> plays
   name  plays
1  John Guitar
2  Paul   Bass
3 Keith Guitar
```

```
> inner_join(names, plays, by = "name")
  name    band  plays
1 John Beatles Guitar
2 Paul Beatles   Bass
```

- returns rows contained in both datasets
- most exclusive join

full_join

```
> names
  name     band
1 Mick   Stones
2 John Beatles
3 Paul Beatles
```

```
> plays
   name  plays
1  John Guitar
2  Paul   Bass
3 Keith Guitar
```

```
> full_join(names, plays, by = "name")
   name     band  plays
1  Mick   Stones   <NA>
2  John Beatles Guitar
3  Paul Beatles   Bass
4 Keith     <NA> Guitar
```

- returns every rows contained in either dataset
- most inclusive join

Syntax

```
> left_join( names, plays, by = "name")
> right_join(names, plays, by = "name")
> inner_join(names, plays, by = "name")
> full_join( names, plays, by = "name")
```

x y by

Data for practice

```r
load(url('https://github.com/hbchoi/SampleData/raw/master/join_practice.RData'))
```

```r
head(bands)
```

```
## # A tibble: 6 x 3
##   first      last      band
##   <chr>      <chr>     <chr>
## 1 John       Bonham    Led Zeppelin
## 2 John Paul  Jones     Led Zeppelin
## 3 Jimmy      Page      Led Zeppelin
## 4 Robert     Plant     Led Zeppelin
## 5 George     Harrison  The Beatles
## 6 John       Lennon    The Beatles
```

```r
head(artists)
```

```
## # A tibble: 6 x 3
##   first  last     instrument
##   <chr>  <chr>    <chr>
## 1 Jimmy  Buffett  Guitar
## 2 George Harrison Guitar
## 3 Mick   Jagger   Vocals
## 4 Tom    Jones    Vocals
## 5 Davy   Jones    Vocals
## 6 John   Lennon   Guitar
```

연습문제

- bands 와 artists 테이블을 left join, right join 해보고 결과를 비교하시오 (key 는 first, last 로 함)
- artists, bands, songs, albums 테이블을 full join을 이용하여 모두 합쳐라

```
## # A tibble: 29 x 7
##    first  last      instrument band                song     album      year
##    <chr>  <chr>     <chr>      <chr>               <chr>    <chr>     <dbl>
##  1 Jimmy  Buffett   Guitar     The Coral Reefers   <NA>     <NA>         NA
##  2 George Harrison  Guitar     The Beatles         <NA>     <NA>         NA
##  3 Mick   Jagger    Vocals     The Rolling Stones  <NA>     <NA>         NA
##  4 Tom    Jones     Vocals     <NA>                It's No~ Along Ca~    NA
##  5 Davy   Jones     Vocals     <NA>                <NA>     <NA>         NA
##  6 John   Lennon    Guitar     The Beatles         Come To~ Abbey Ro~  1969
##  7 Paul   McCartney Bass       The Beatles         Hello, ~ Magical ~  1967
##  8 Jimmy  Page      Guitar     Led Zeppelin        <NA>     <NA>         NA
##  9 Joe    Perry     Guitar     <NA>                <NA>     <NA>         NA
## 10 Elvis  Presley   Vocals     <NA>                <NA>     <NA>         NA
## # ... with 19 more rows
```

tidyr 리뷰

- **Gather columns into key-value pairs**

```
library(tidyr)
wide_df <- data.frame(col = c('X', 'Y'), A = c(1,4), B = c(2,5), C = c(3,6))

# Look at wide_df
wide_df

##   col A B C
## 1   X 1 2 3
## 2   Y 4 5 6

# Gather the columns of wide_df
gather(wide_df, my_key, my_val, -col)

##   col my_key my_val
## 1   X      A      1
## 2   Y      A      4
## 3   X      B      2
## 4   Y      B      5
## 5   X      C      3
## 6   Y      C      6
```

gather(data, key, value, …)

data: a data frame

key: bare name of new key column

value: bare name of new value column

…: bare names of columns to gather (or not)

Spread key-value pairs into columns

```
long_df <- gather(wide_df, my_key, my_val, -col)

# Look at long_df
long_df

##   col my_key my_val
## 1   X      A      1
## 2   Y      A      4
## 3   X      B      2
## 4   Y      B      5
## 5   X      C      3
## 6   Y      C      6

# Spread the key-value pairs of long_df
spread(long_df, my_key, my_val)

##   col A B C
## 1   X 1 2 3
## 2   Y 4 5 6
```

spread(data, key, value)

data: a data frame

key: bare name of column containing keys

value: bare name of column containing values

Separate columns

```
treatments <- data.frame(patient = rep(c('X', 'Y'),3) ,
                         treatment = rep(c('A', 'B'), each = 3),
                         year_mo = rep(c('2010-10', '2012-08', '2014-12'), each = 2),
                         response = c(1,4,2,5,3,6))

# View the treatments data
treatments

##   patient treatment year_mo response
## 1       X         A 2010-10        1
## 2       Y         A 2010-10        4
## 3       X         A 2012-08        2
## 4       Y         B 2012-08        5
## 5       X         B 2014-12        3
## 6       Y         B 2014-12        6

# Separate year_mo into two columns
separate(treatments, year_mo, c("year", "month"))

##   patient treatment year month response
## 1       X         A 2010    10        1
## 2       Y         A 2010    10        4
## 3       X         A 2012    08        2
## 4       Y         B 2012    08        5
## 5       X         B 2014    12        3
## 6       Y         B 2014    12        6
```

separate(data, col, into)

data: a data frame **sep = "-"**

col: bare name of column to separate

into: character vector of new column names

Unite columns

```
treatments2 <- separate(treatments, year_mo, c("year", "month"))

# View treatments2 data
treatments2

##   patient treatment year month response
## 1       X         A 2010    10        1
## 2       Y         A 2010    10        4
## 3       X         A 2012    08        2
## 4       Y         B 2012    08        5
## 5       X         B 2014    12        3
## 6       Y         B 2014    12        6

# Unite year and month to form year_mo column
unite(treatments2, year_mo, year, month)

##   patient treatment year_mo response
## 1       X         A 2010_10        1
## 2       Y         A 2010_10        4
## 3       X         A 2012_08        2
## 4       Y         B 2012_08        5
## 5       X         B 2014_12        3
## 6       Y         B 2014_12        6
```

unite(data, col, …)

data: a data frame **sep = "-"**

col: bare name of new column

…: bare names of columns to unite

Try different sep option (by default sep = '_')

연습용 데이터

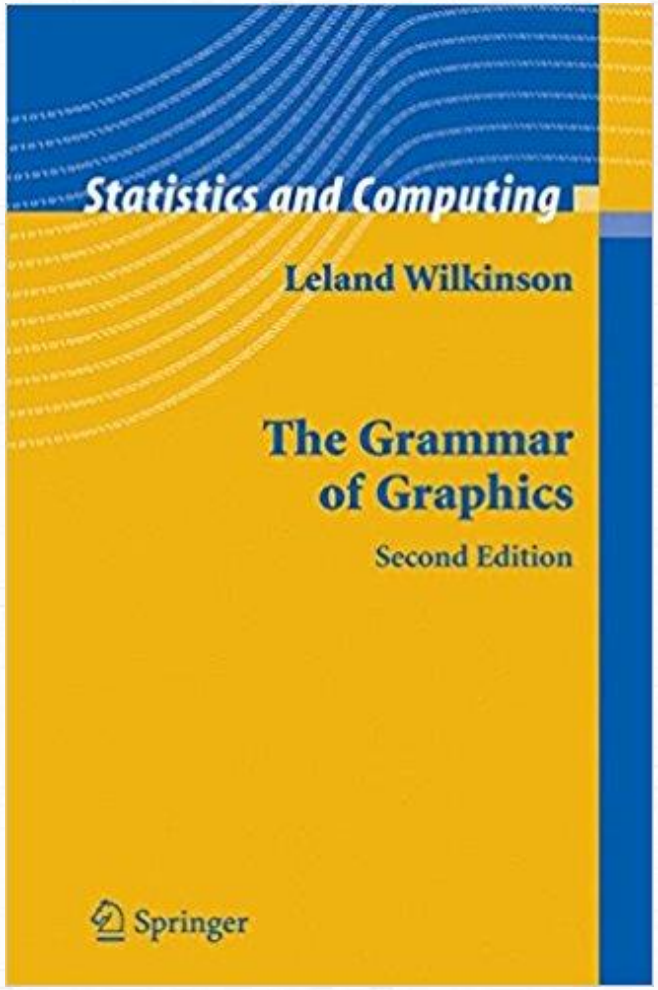
```
load(url('https://github.com/hbchoi/SampleData/raw/master/weather.RData'))
```

- weather2 테이블로부터 월별 min, max, mean.temperature의 평균을 계산하시오. (tidyr, dplyr을 pipe로 연동)

데이터 시각화 with ggplot

- Exploratory Visualization
  - Help you see what is in the data
- Explanatory Visualization
  - Shows others what you've found in your data

Grammar of Graphics



All Grammatical Elements

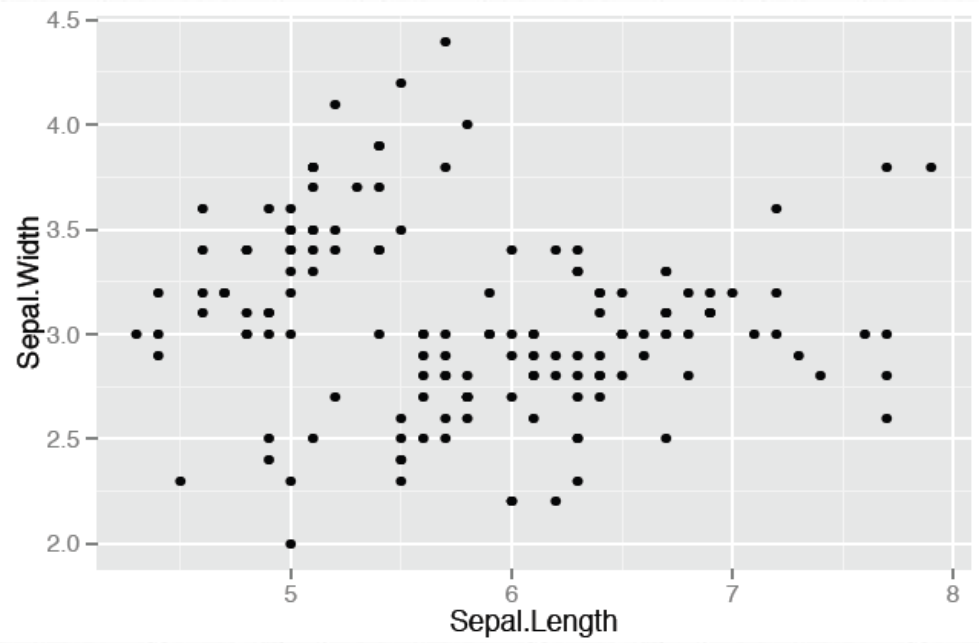
Element	Description
Data	**The dataset being plotted.**
Aesthetics	**The scales onto which we *map* our data.**
Geometries	**The visual elements used for our data.**
Facets	Plotting small multiples
Statistics	Representations of our data to aid understanding.
Coordinates	The space on which the data will be plotted.
Themes	All non-data ink.

Diagram

Data	*{variables of interest}*				
Aesthetics	*x-axis* *y-axis*	*colour* *fill*	*size* *labels*	*alpha* *shape*	*line width* *line type*
Geometries	*point*	*line*	*histogram*	*bar*	*boxplot*
Facets	*columns*	*rows*			
Statistics	*binning*	*smoothing*	*descriptive*	*inferential*	
Coordinates	*cartesian*	*fixed*	*polar*	*limits*	
Themes	*non-data ink*				

Scatter Plot

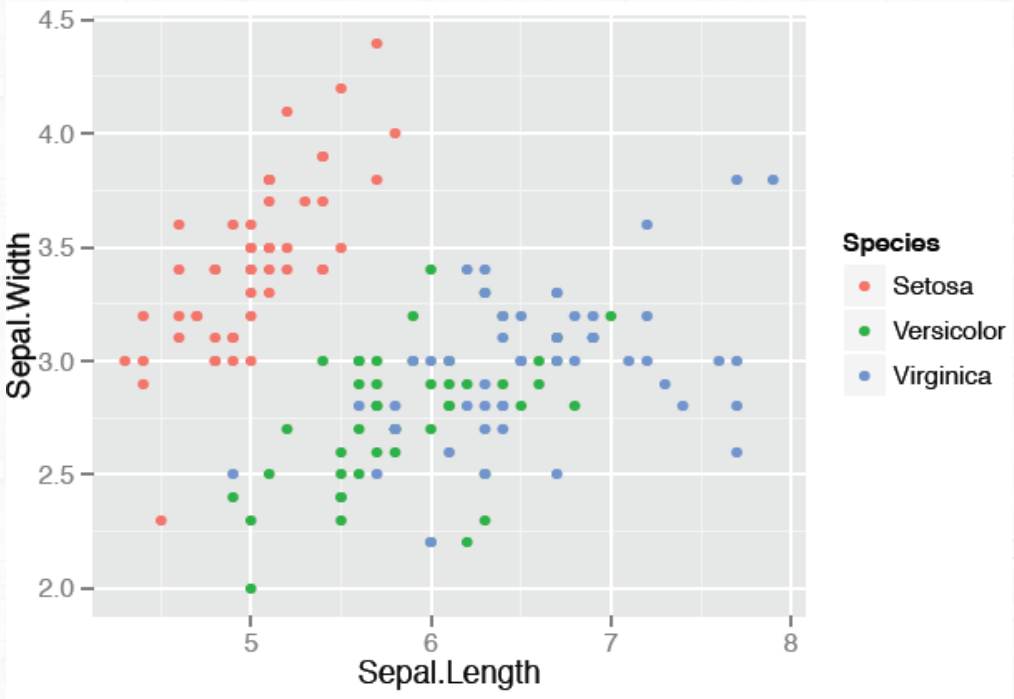
- Each geom has specific aesthetic mappings
- geom_point()
  - Essential: x, y




```
> ggplot(iris, aes(x = Sepal.Length, y = Sepal.Width)) +
    geom_point()
```

Scatter Plot

- geom_point()
  - Essential: x, y
  - Optional: alpha, colour, fill, shape, size




```
> ggplot(iris, aes(x = Sepal.Length, y = Sepal.Width, col = Species)) +
  geom_point()
```

Summary Statistics

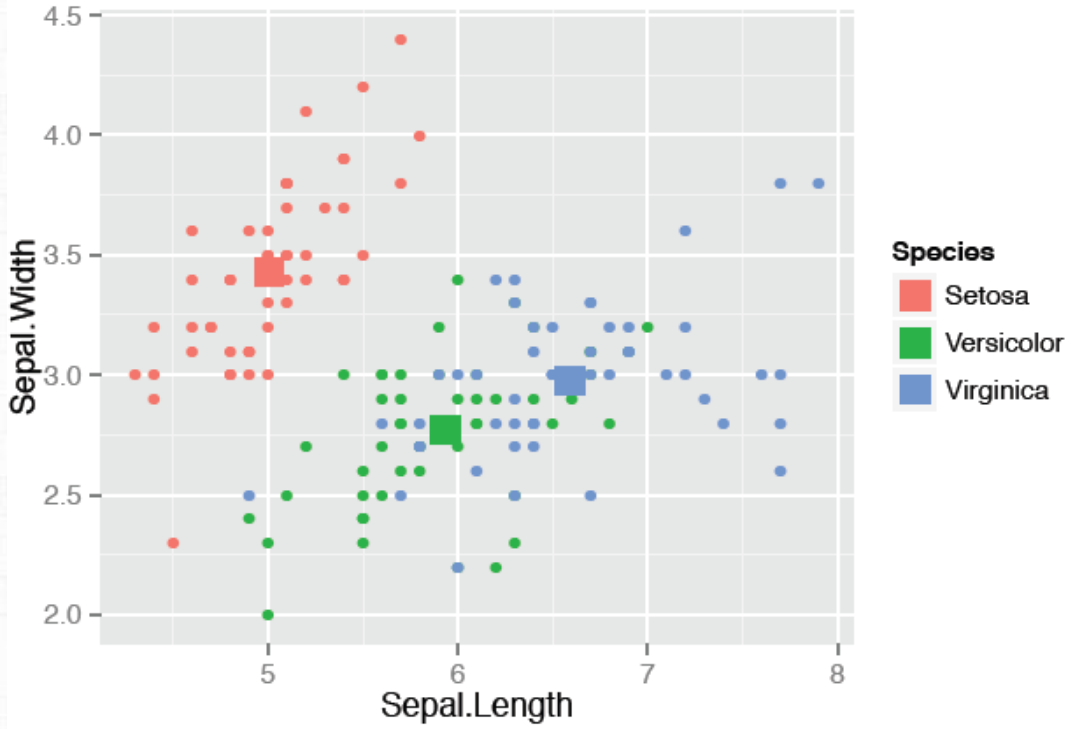
```
> head(iris)
  Species Sepal.Length Sepal.Width Petal.Length Petal.Width
1  Setosa          5.1         3.5          1.4         0.2
2  Setosa          4.9         3.0          1.4         0.2
3  Setosa          4.7         3.2          1.3         0.2
4  Setosa          4.6         3.1          1.5         0.2
5  Setosa          5.0         3.6          1.4         0.2
6  Setosa          5.4         3.9          1.7         0.4

> iris.summary <- aggregate(iris[2:5], list(iris$Species), mean)
> names(iris.summary)[1] <- "Species"
> iris.summary
     Species Sepal.Length Sepal.Width Petal.Length Petal.Width
1     Setosa        5.006       3.428        1.462       0.246
2 Versicolor        5.936       2.770        4.260       1.326
3  Virginica        6.588       2.974        5.552       2.026
```

Add Layers

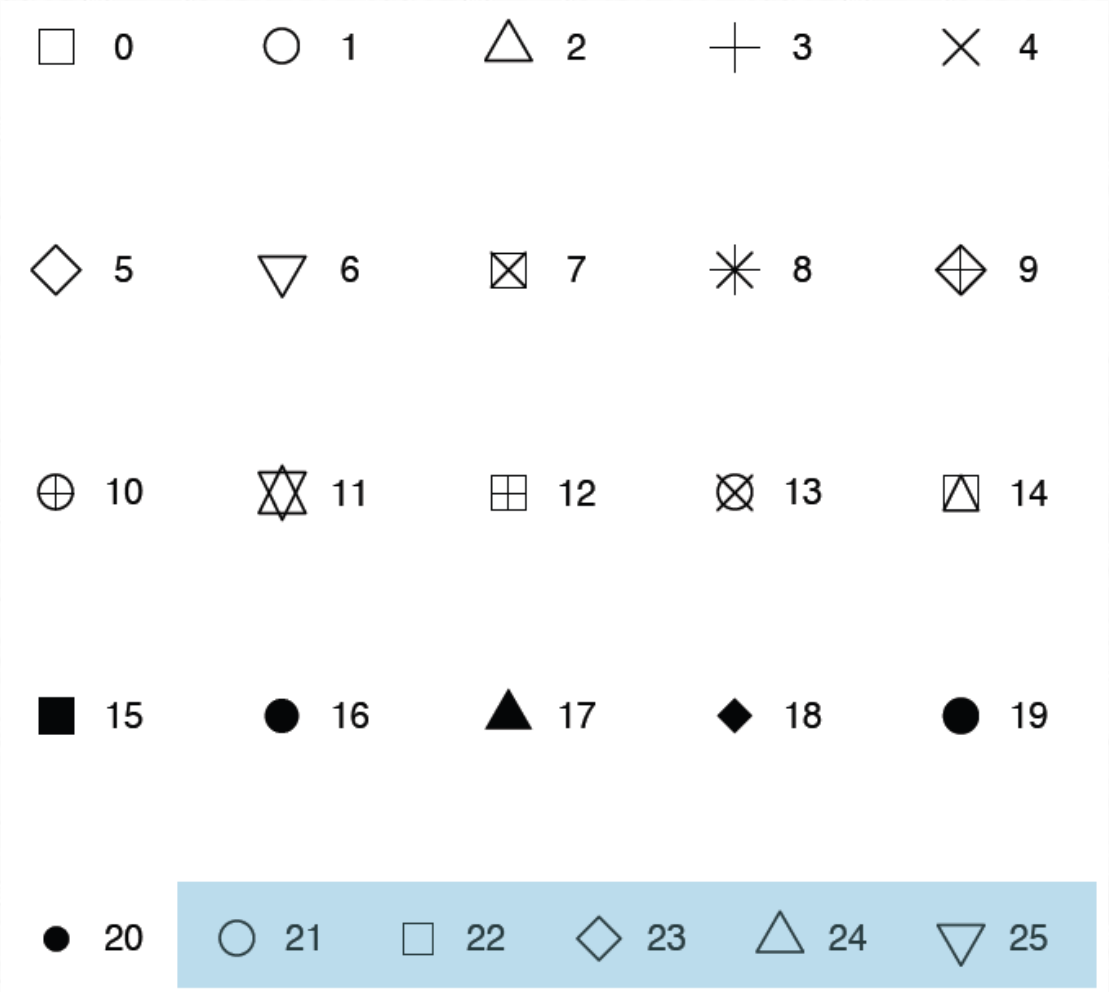
```
> ggplot(iris, aes(x = Sepal.Length, y = Sepal.Width, col = Species)) +
    geom_point() +
    geom_point(data = iris.summary, shape = 15, size = 5)
```

inherits data and aes from ggplot()

different data
inherits aes



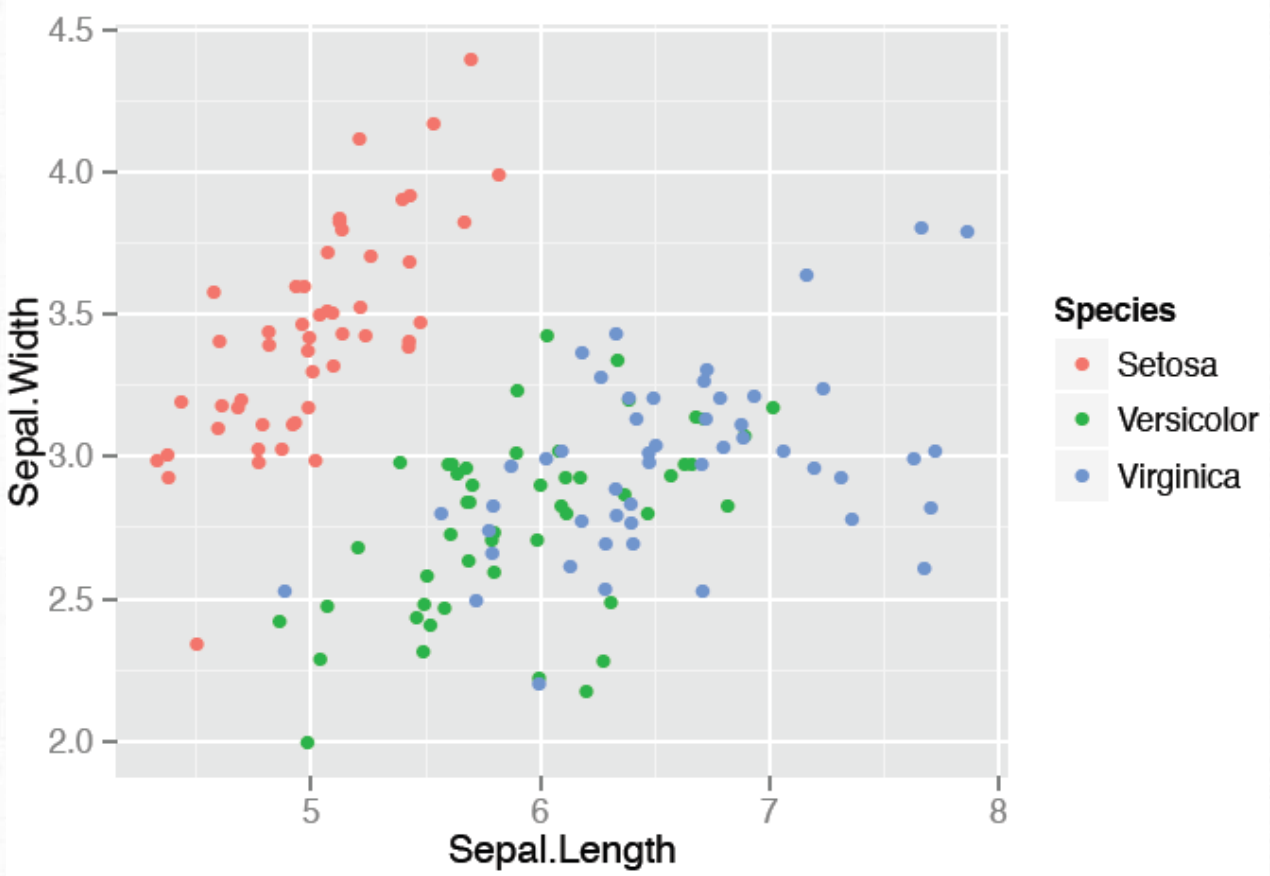
shape



both fill and color

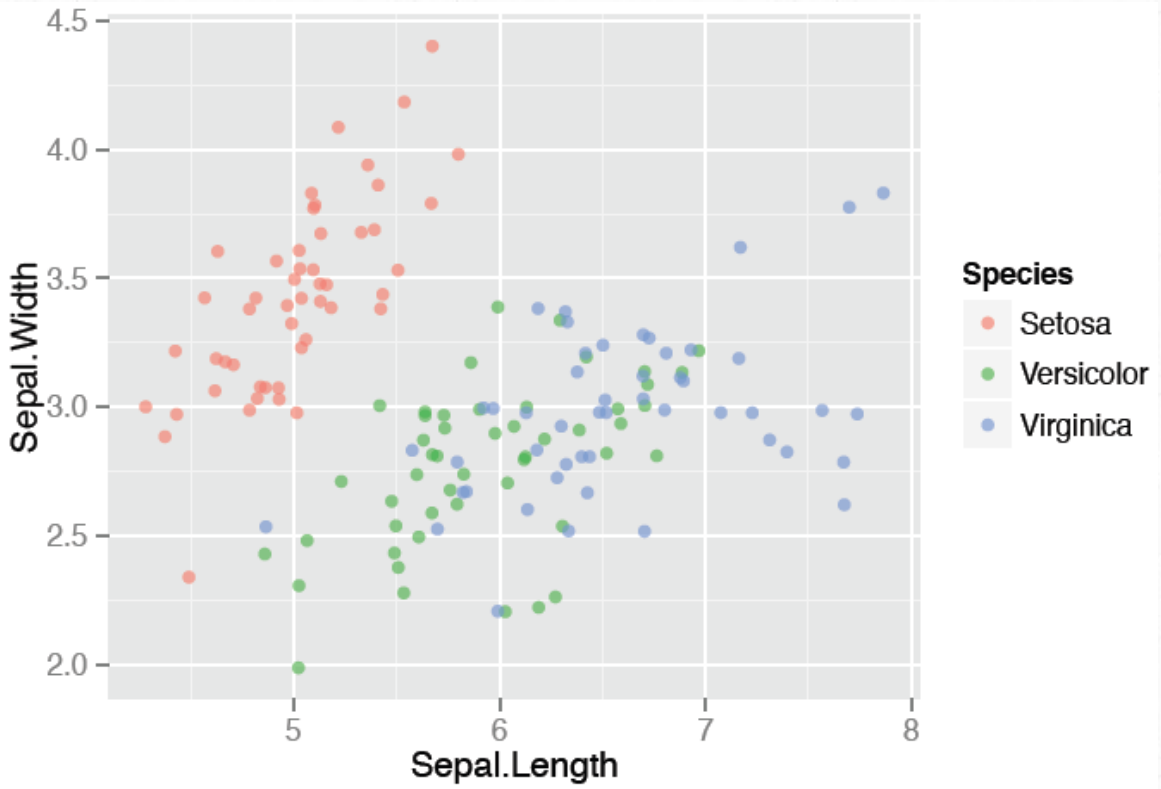
Jittering

```
> ggplot(iris, aes(x = Sepal.Length, y = Sepal.Width, col = Species)) +
    geom_point(position = "jitter")
```



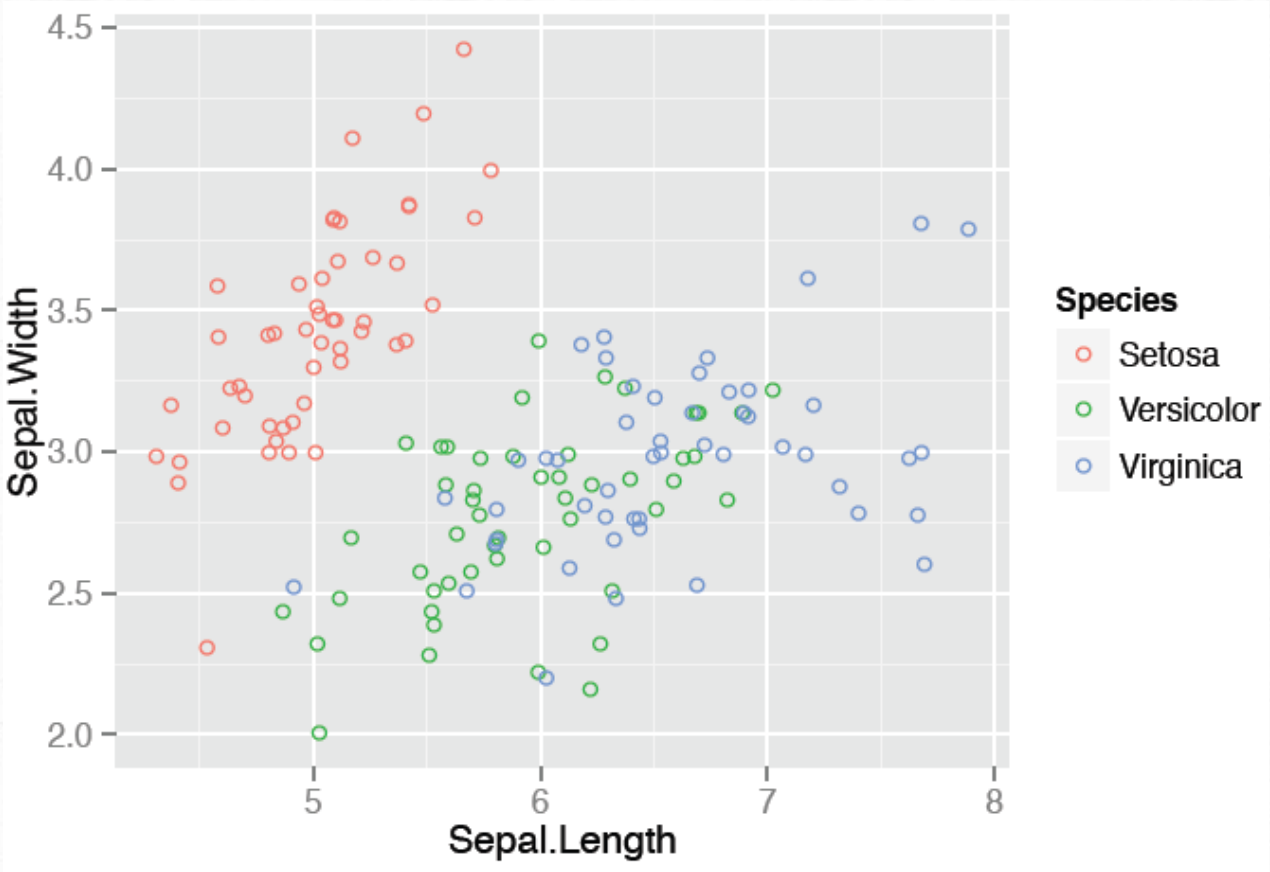
jittering

```
> ggplot(iris, aes(x = Sepal.Length, y = Sepal.Width, col = Species)) +
    geom_jitter(alpha = 0.6)
```



jittering

```
> ggplot(iris, aes(x = Sepal.Length, y = Sepal.Width, col = Species)) +
    geom_jitter(shape = 1)
```

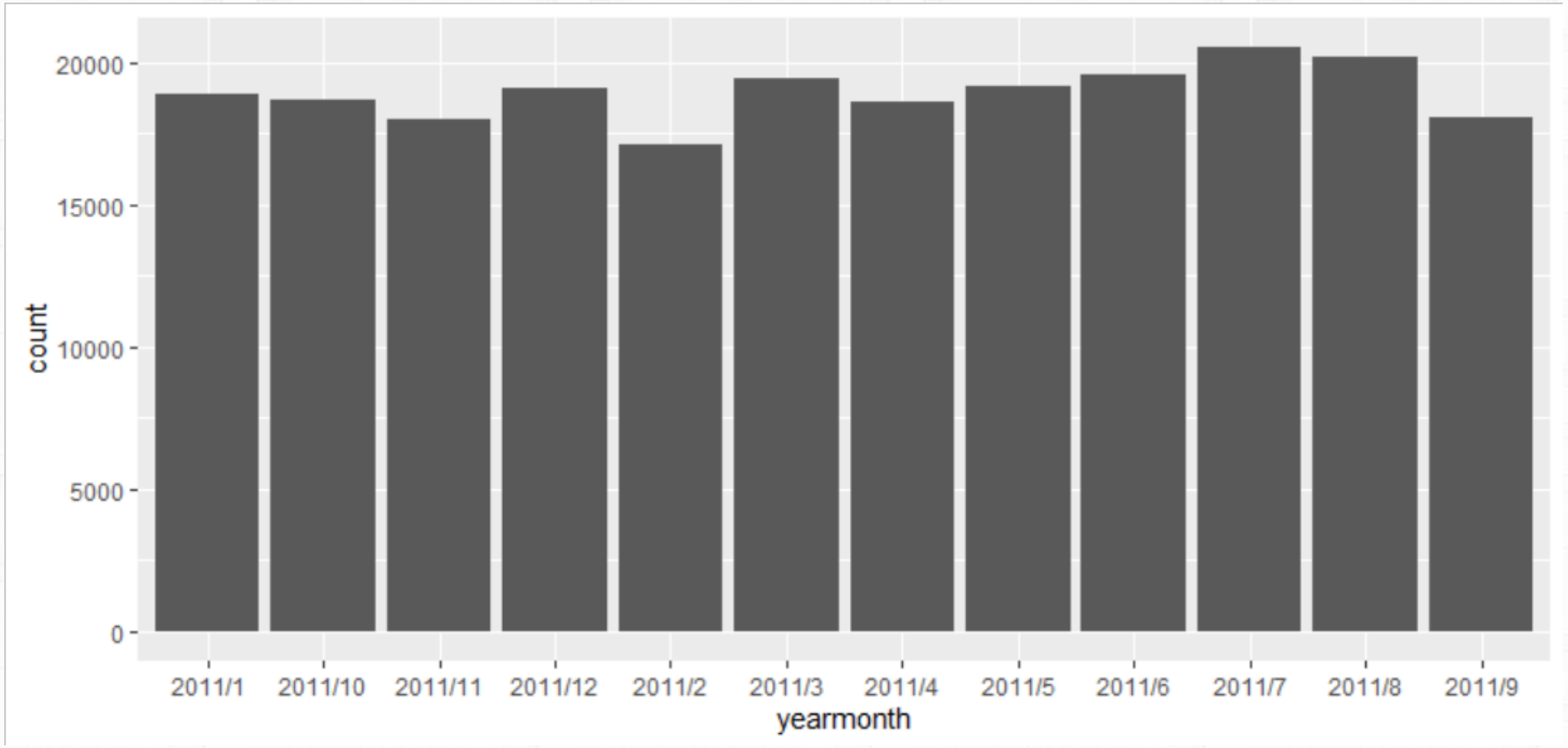


bar plot

- geom_bar()
- All positions from before available
- Two types
  - stat counts
  - absolute values

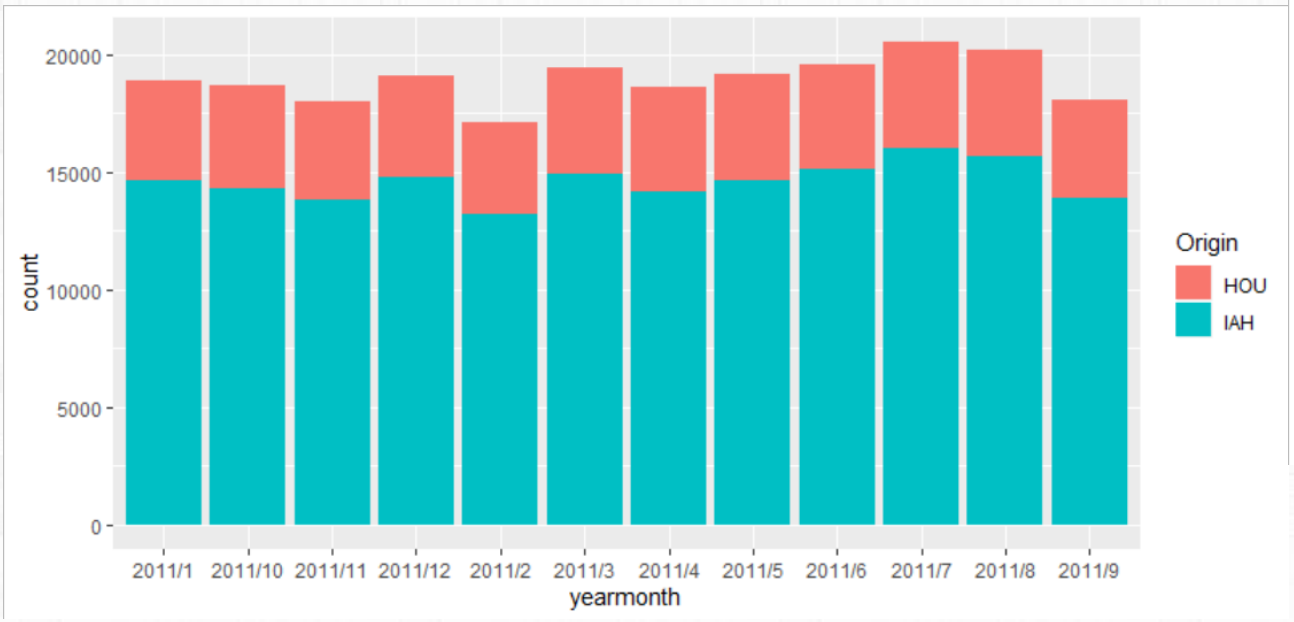
bar plot with stat count

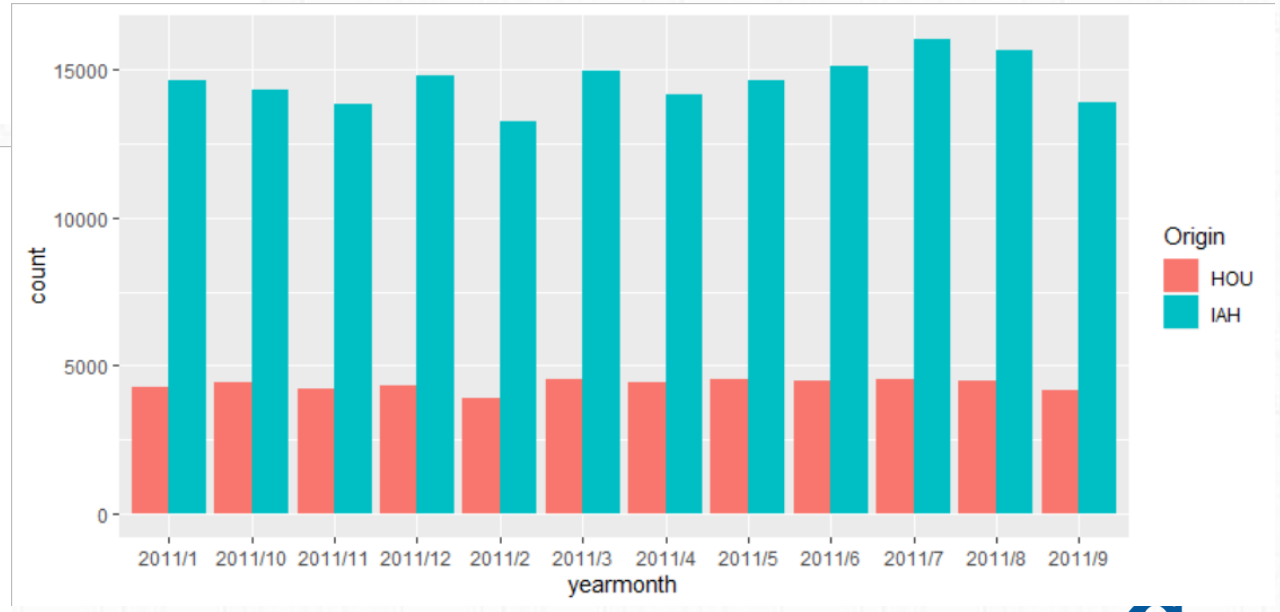
```
hf_data %>% mutate(yearmonth = paste(Year, Month, sep = '/')) %>%
  ggplot(aes(x = yearmonth)) +
  geom_bar()
```



bar plot with stat count

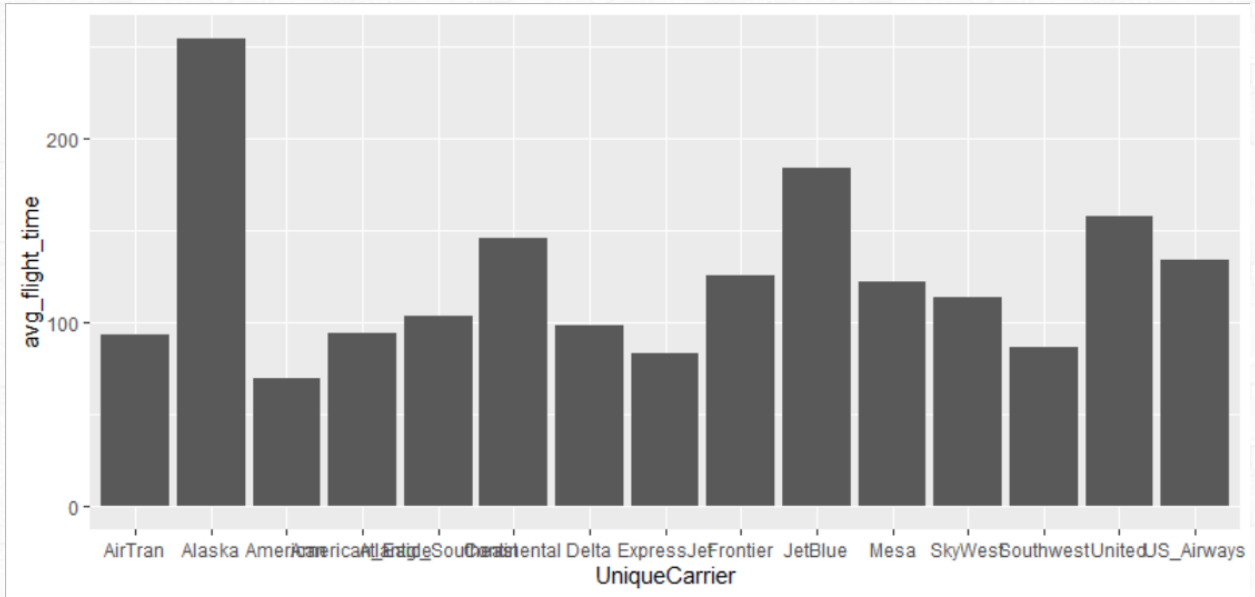
```
hf_data %>% mutate(yearmonth = paste(Year, Month, sep = '/')) %>%
  ggplot(aes(x = yearmonth, fill = Origin)) +
  geom_bar()

hf_data %>% mutate(yearmonth = paste(Year, Month, sep = '/')) %>%
  ggplot(aes(x = yearmonth, fill = Origin)) +
  geom_bar(position = position_dodge())
```

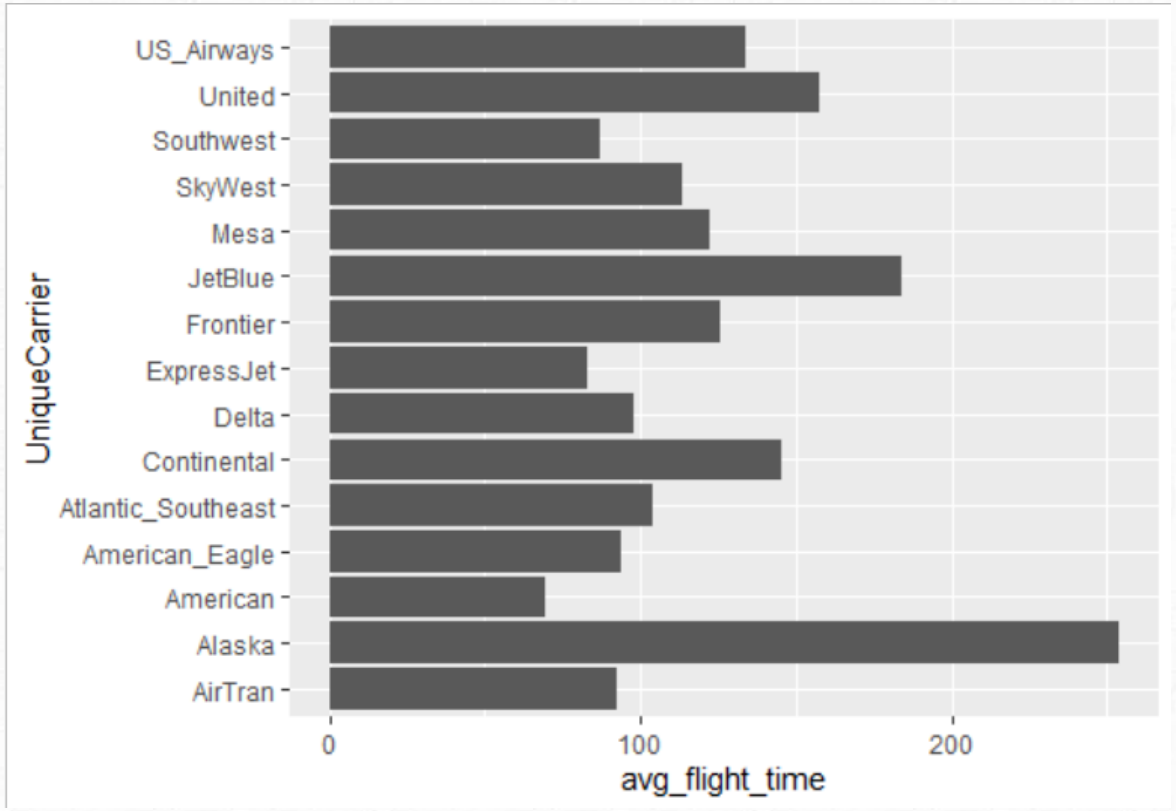




bar plot with absolute value

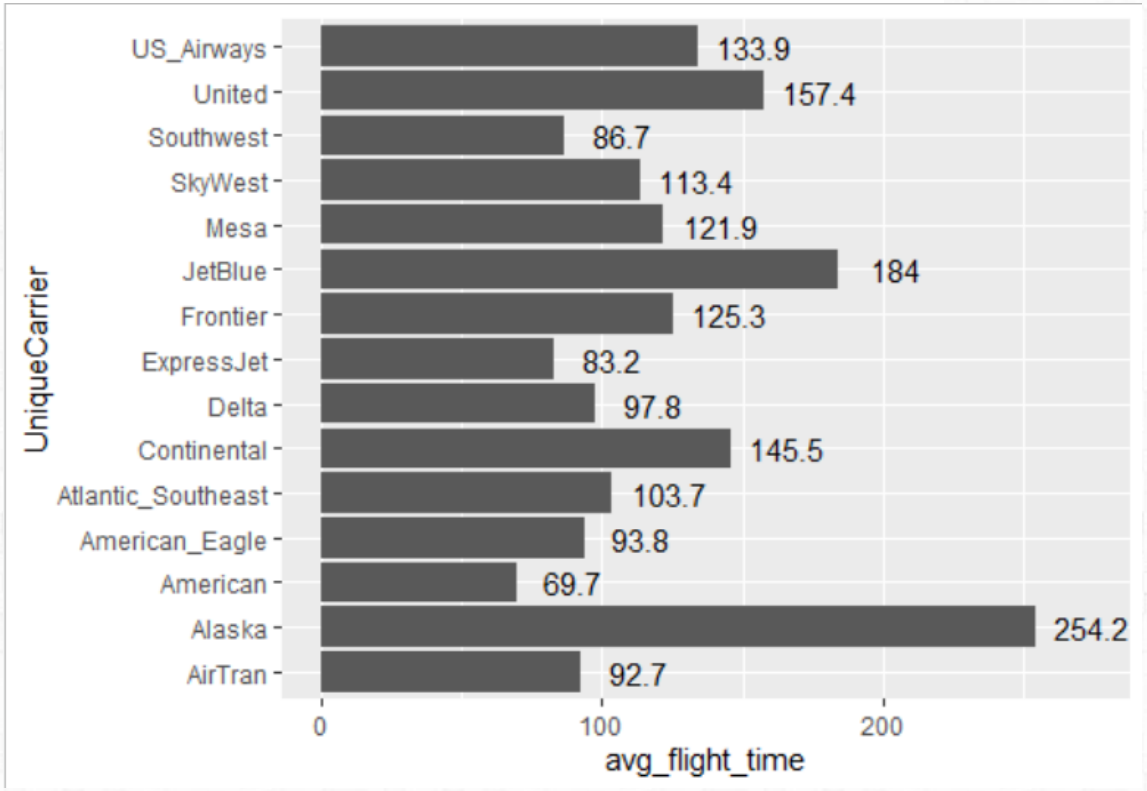
```
hf_data %>%
  group_by(UniqueCarrier) %>%
  summarise(avg_flight_time = mean(AirTime, na.rm = T)) %>%
  ggplot(aes(x = UniqueCarrier, y = avg_flight_time)) +
  geom_bar(stat = 'identity')

hf_data %>%
  group_by(UniqueCarrier) %>%
  summarise(avg_flight_time = mean(AirTime, na.rm = T)) %>%
  ggplot(aes(x = UniqueCarrier, y = avg_flight_time)) +
  geom_bar(stat = 'identity') +
  coord_flip()
```





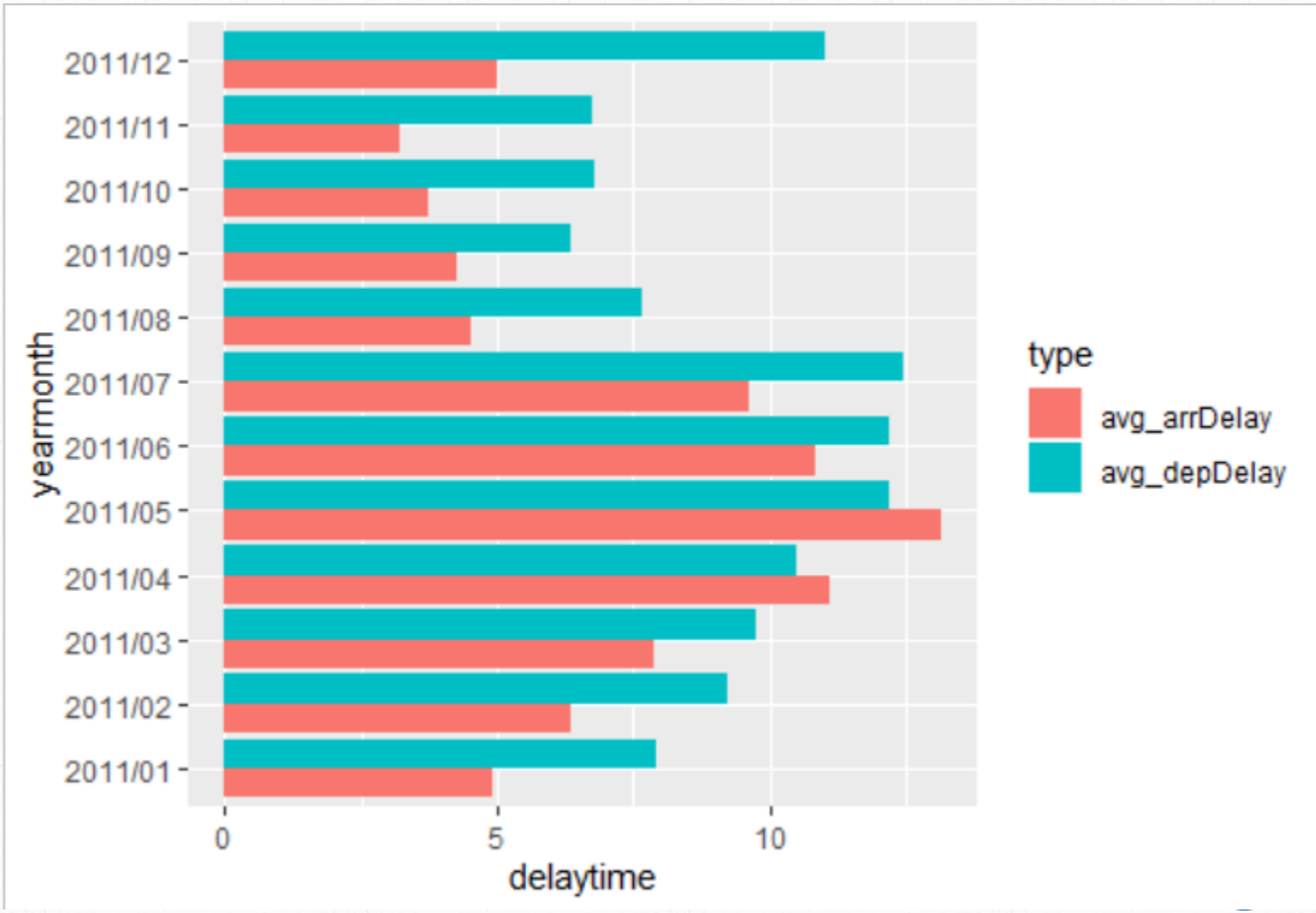
adding labels

```
hf_data %>%
  group_by(UniqueCarrier) %>%
  summarise(avg_flight_time = mean(AirTime, na.rm = T)) %>%
  ggplot(aes(x = UniqueCarrier, y = avg_flight_time)) +
  geom_bar(stat = 'identity') +
  geom_text(aes(label = round(avg_flight_time,1)), nudge_y = 20) +
  coord_flip()
```



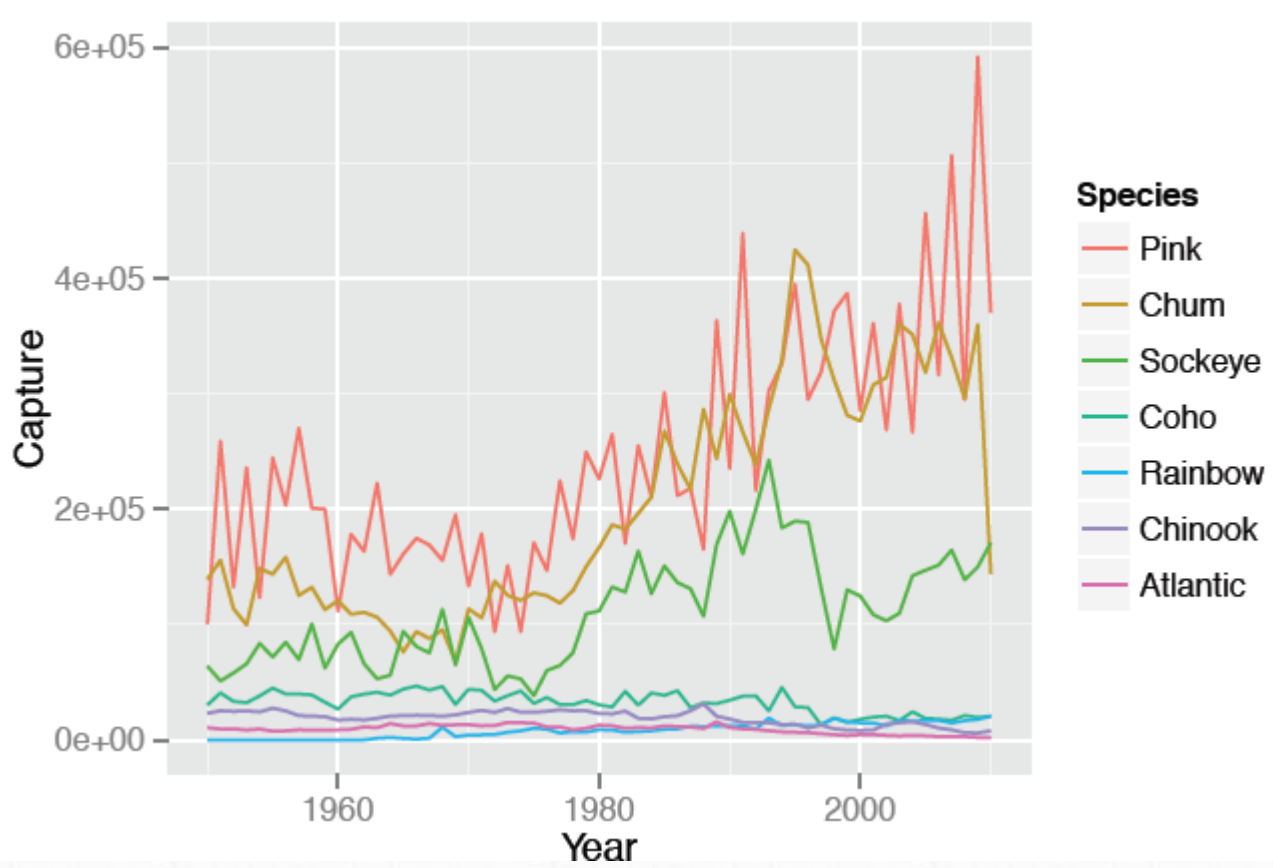
연습문제

- 다음시각화를 수행해보라
- 월별 도착/출발 딜레이 평균을 막대그래프로 표현



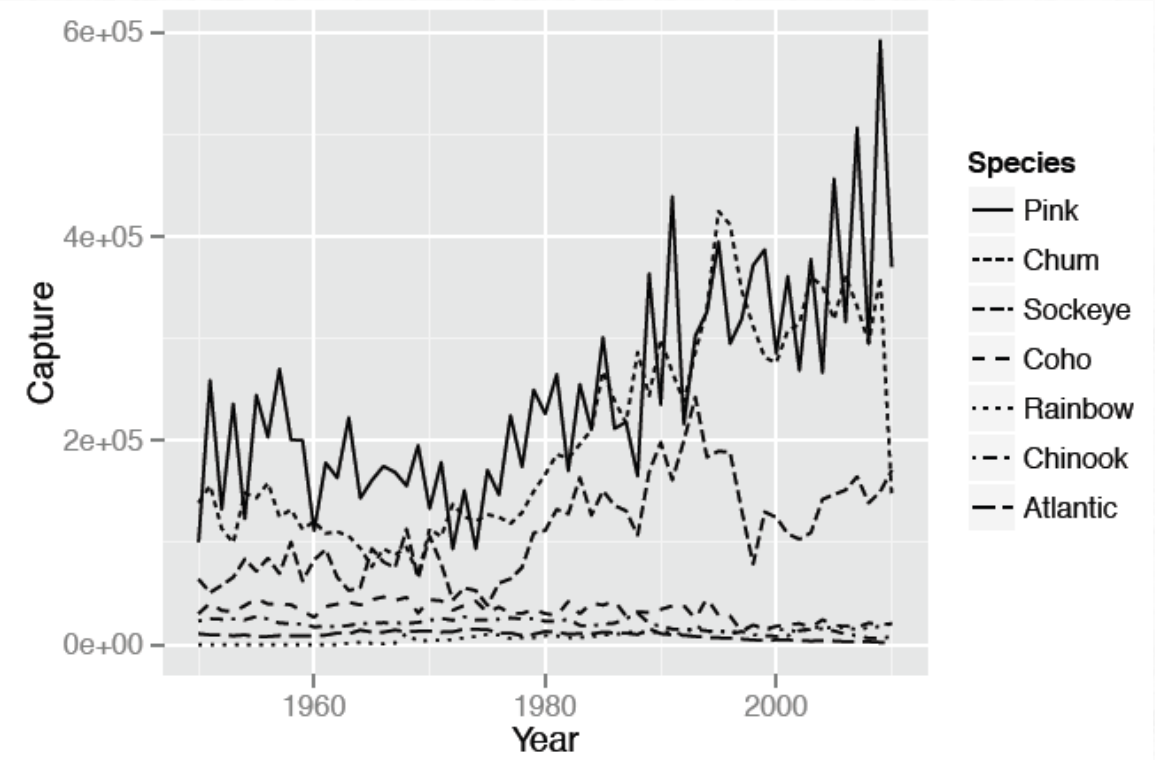
line graph

- 시간에 따른 변화량/추세를 표현하기 좋음



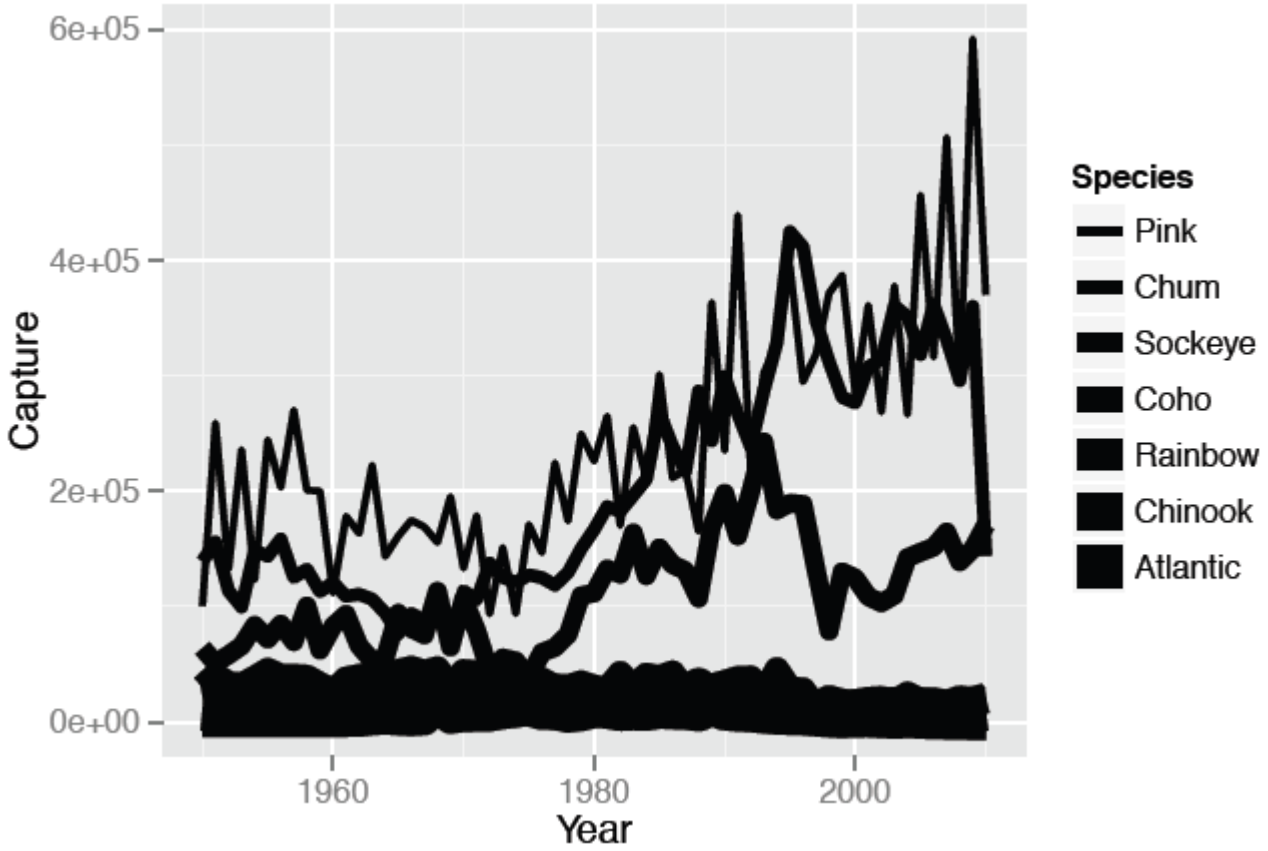
linetype

```
> ggplot(fish, aes(x = Year, y = Capture, linetype = Species)) +
    geom_line()
> names(fish)
[1] "Species" "Year"    "Capture"
```



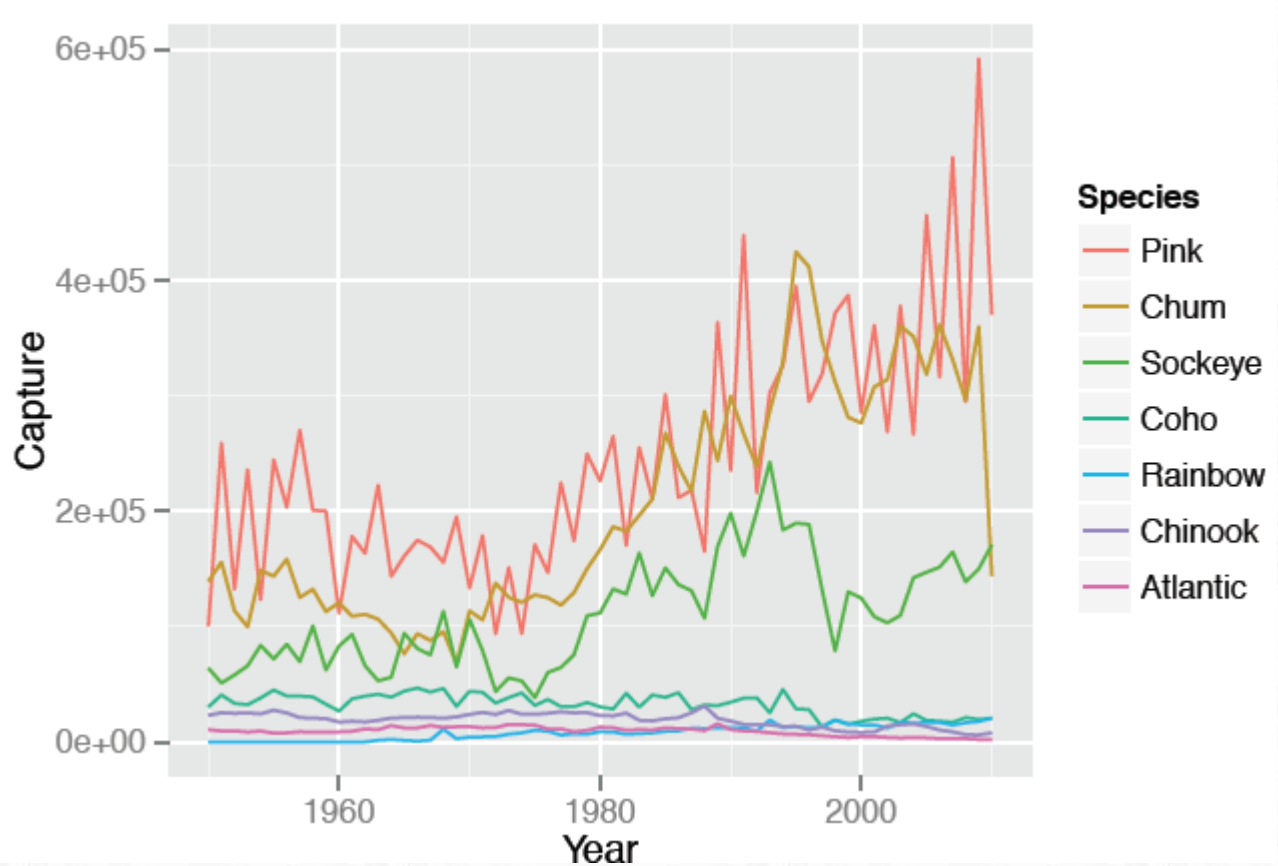
size

```
> ggplot(fish, aes(x = Year, y = Capture, size = Species)) +
    geom_line()
```

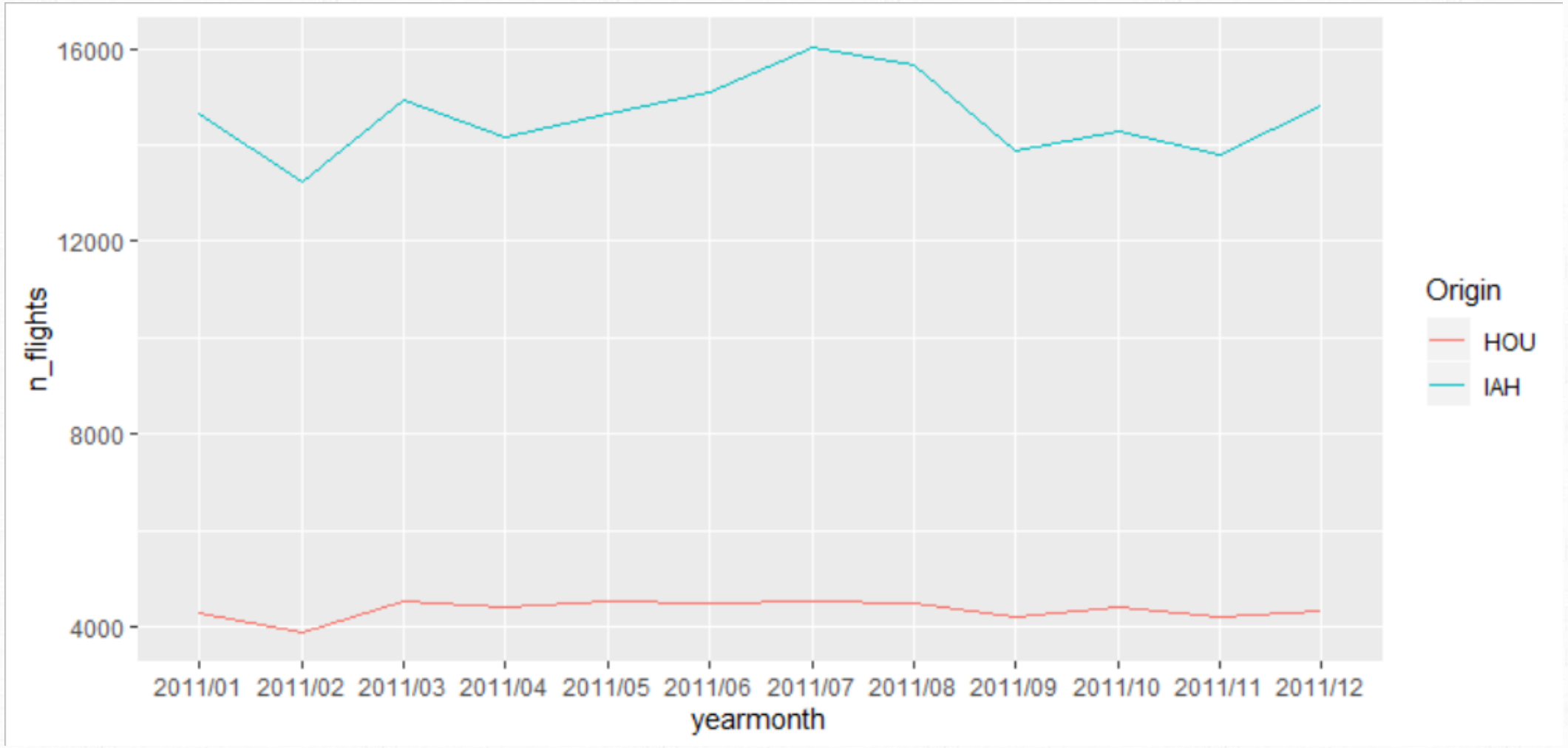


color

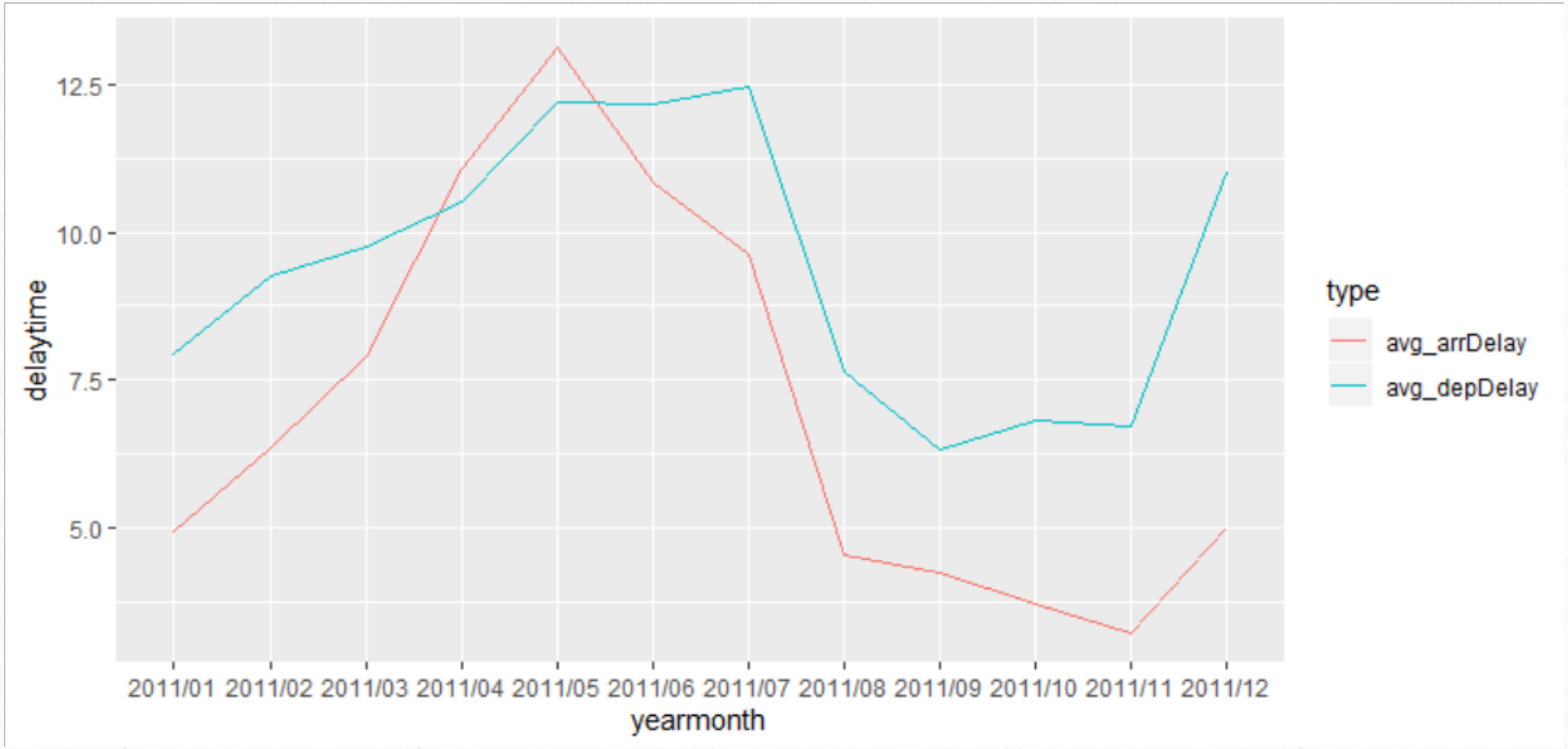
```
> ggplot(fish, aes(x = Year, y = Capture, color = Species)) +
    geom_line()
```



```
hf_data %>% mutate(yearmonth = sprintf("%4d/%02d",Year, Month)) %>%
  group_by(yearmonth, Origin) %>%
  summarize(n_flights = n()) %>%
  ggplot(aes(x = yearmonth, y = n_flights, col = Origin, group = Origin)) +
  geom_line()
```



```
hf_data %>% mutate(yearmonth = sprintf("%4d/%02d",Year, Month)) %>%
  group_by(yearmonth) %>%
  summarize(avg_arrDelay = mean(ArrDelay, na.rm = T),
            avg_depDelay = mean(DepDelay, na.rm = T)) %>%
  gather(type, delaytime, avg_arrDelay, avg_depDelay) %>%
  ggplot(aes(x = yearmonth, y = delaytime, col = type, group = type)) +
  geom_line()
```



```r
hf_data %>% mutate(yearmonth = sprintf("%4d/%02d",Year, Month)) %>%
  group_by(yearmonth) %>%
  summarize(avg_arrDelay = mean(ArrDelay, na.rm = T),
            avg_depDelay = mean(DepDelay, na.rm = T)) %>%
  gather(type, delaytime, avg_arrDelay, avg_depDelay) %>%
  ggplot(aes(x = yearmonth, y = delaytime, col = type, group = type)) +
  geom_line() +
  ggtitle('average delay per month') +
  labs(x = 'Year/Month', y = 'delay in minute', col = 'delay type') +
  scale_color_discrete(labels = c('arrival delay', 'departure delay')) +
  theme_bw()
```