Exercises

Import data

3.2.5: 1,4,5

1

```r
# Combining all conditions into single pipe for fun
a <- flights |>
  filter(arr_delay>=2) |>
  filter(dest == 'IAH' | dest == 'HOU') |>
  filter(carrier == 'AA' | carrier == 'UA' | carrier == 'DL') |>
  filter(month == 7 | month == 8 | month == 9) |>
  filter(dep_delay<=0)

# Have to do this pipe separate cause it contradicts if in other one
b <- flights |>
  filter(dep_delay>=1 & arr_delay<30)
```

4

```r
un <- flights |>
  distinct(month,day)

dim(un)
```

```
[1] 365   2
```

There are 365 unique month, day pairs, so yes there was a flight everyday of 2013.

5

```
long <- flights |>
  arrange(desc(distance)) |>
  head(1)

short <- flights |>
  arrange(distance) |>
  head(1)

long
```

```
# 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
# i 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>
```

```
short
```

```
# 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
# i 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>
```

JFK to HNL is the longest and JFK to LGA is the shortest

3.3.5: 1,4

1

```
flights |>
  mutate(delay = dep_time-sched_dep_time) |>
  mutate(diff = delay-dep_delay) |>
  filter(diff != 0 | !is.na(diff))
```

```
# A tibble: 328,521 x 21
    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
# i 328,511 more rows
# i 13 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>, delay <int>, diff <dbl>
```

I would expect dep_deal = dep_time - sched_dep_time, but we can see they are actually storing time as hmm. So 558 is 5:58 and 600 is 6:00, so the intuitive approach doesn't work.

4

any_of() selects any variable in a vector. It helps when you want variables removed because calling it twice doesn't error.

3.5.7: 1,2,4,6

1

```r
flights |>
  group_by(carrier) |>
  summarise(avg_dep_delay = mean(dep_delay, na.rm = TRUE)) |>
  arrange(desc(avg_dep_delay))
```

```
# A tibble: 16 x 2
   carrier avg_dep_delay
   <chr>           <dbl>
 1 F9              20.2
 2 EV                20.0 
 3 YV                19.0 
 4 FL                18.7 
 5 WN                17.7 
 6 9E                16.7 
 7 B6                13.0 
 8 VX                12.9 
 9 OO                12.6 
10 UA                12.1 
11 MQ                10.6 
12 DL                 9.26
13 AA                 8.59
14 AS                 5.80
15 HA                 4.90
16 US                 3.78
```

F9 is the worst.

2

```
flights |>
  group_by(dest) |>
  slice_max(dep_delay, n = 1, with_ties = FALSE) |>
  ungroup()
```

```
# A tibble: 105 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    12    14     2223           2001       142      133           2304
 2  2013     7    23     1139            800       219     1250            909
 3  2013     1    25      123           2000       323      229           2101
 4  2013     8    17     1740           1625        75     2042           2003
 5  2013     7    22     2257            759       898      121           1026
 6  2013     7    10     2056           1505       351     2347           1758
 7  2013     6    14     1158            816       222     1335           1007
 8  2013     2    21     1728           1316       252     1839           1413
 9  2013    12     1     1504           1056       248     1628           1230
10  2013     4    10       25           1900       325      136           2045
# i 95 more rows
# i 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>
```

4

It excludes the n min/max rows and returns the rest.

6

```
df <- tibble(
  x = 1:5,
  y = c("a", "b", "a", "a", "b"),
  z = c("K", "K", "L", "L", "K")
)
```

group_by() will set the metadata group values for the tibble for the columns passed.

```
df |>
  group_by(y)
```

```
# A tibble: 5 x 3
# Groups:   y [2]
      x y     z
  <int> <chr> <chr>
1     1 a     K
2     2 b     K
3     3 a     L
4     4 a     L
5     5 b     K
```

arrange(y) will order y alphabetically

```
df |>
  arrange(y)
```

```
# A tibble: 5 x 3
      x y     z
  <int> <chr> <chr>
```

```
1     1 a     K
2     3 a     L
3     4 a     L
4     2 b     K
5     5 b     K
```

This will get the mean x value for each y value:

```r
df |>
  group_by(y) |>
  summarize(mean_x = mean(x))
```

```
# A tibble: 2 x 2
  y     mean_x
  <chr>  <dbl>
1 a       2.67
2 b       3.5
```

This will do the same but for each unique y,z pair (and drop the group metadata):

```r
df |>
  group_by(y, z) |>
  summarize(mean_x = mean(x), .groups = "drop")
```

```
# A tibble: 3 x 3
  y     z     mean_x
  <chr> <chr>  <dbl>
1 a     K        1
2 a     L        3.5
3 b     K        3.5
```

These will get the mean_x for each unique y,z pair. The second one adds a column to the original tibble and the first just lists all uniqe values.

```r
df |>
  group_by(y, z) |>
  summarize(mean_x = mean(x))
```

```
`summarise()` has grouped output by 'y'. You can override using the `.groups`
argument.
```

```
# A tibble: 3 x 3
# Groups:   y [2]
  y     z     mean_x
  <chr> <chr>  <dbl>
1 a     K        1
2 a     L        3.5
3 b     K        3.5
```

```
df |>
  group_by(y, z) |>
  mutate(mean_x = mean(x))
```

```
# A tibble: 5 x 4
# Groups:   y, z [3]
      x y     z     mean_x
  <int> <chr> <chr>  <dbl>
1     1 a     K        1
2     2 b     K        3.5
3     3 a     L        3.5
4     4 a     L        3.5
5     5 b     K        3.5
```