Lab 3: More data management and grouping using factors

NAME

DATE

Answer the questions in this lab and submit the compiled HTML or PDF by the deadline. Don't forget to change your name and date in the above header.

This lab uses the `dplyr` and `nycflights13` packages. It is good habit to load all packages in the first code chunk.

```
library(dplyr)
library(nycflights13)
```

Basic verbs

You will use the `flights` data set for the next few exercises. Let's load that into our working environment so we can look at it.

```
flights <- flights
```

- At each step use the assignment operator `<-` to store the results into a new data table and use that data in the next step.
- At each step, print out the resulting data frame so you can see the results.

1. Use `select()` to extract the following variables: `origin`, `distance`, and `air_time`, `dest`.

```
f1 <- flights %>% select(origin, distance, air_time, dest)
f1
```

```
## # A tibble: 336,776 x 4
##    origin distance air_time dest
##    <chr>     <dbl>    <dbl> <chr>
##  1 EWR        1400      227 IAH
##  2 LGA        1416      227 IAH
##  3 JFK        1089      160 MIA
##  4 JFK        1576      183 BQN
##  5 LGA         762      116 ATL
##  6 EWR         719      150 ORD
##  7 EWR        1065      158 FLL
##  8 LGA         229       53 IAD
##  9 JFK         944      140 MCO
## 10 LGA         733      138 ORD
## # ... with 336,766 more rows
```

2. Use `filter()` to select only the flights whose destination (`dest`) is Atlanta (ATL)

```
f2 <- f1 %>% filter(dest == 'ATL')
f2
```

```
## # A tibble: 17,215 x 4
##    origin distance air_time dest
##    <chr>     <dbl>    <dbl> <chr>
##   1 LGA         762      116 ATL
##   2 LGA         762      134 ATL
##   3 JFK         760      128 ATL
##   4 EWR         746      120 ATL
##   5 LGA         762      126 ATL
##   6 LGA         762      126 ATL
##   7 JFK         760      126 ATL
##   8 LGA         762      132 ATL
##   9 LGA         762      123 ATL
##  10 LGA         762      129 ATL
## # ... with 17,205 more rows
```

3. Use `mutate()` to create a new variable `speed` that calculates speed of the plane as 'distance/air_time*60.

```
f3 <- f2 %>% mutate(speed = distance/air_time*60)
f3
```

```
## # A tibble: 17,215 x 5
##    origin distance air_time dest  speed
##    <chr>     <dbl>    <dbl> <chr> <dbl>
##  1 LGA         762      116 ATL    394.
##  2 LGA         762      134 ATL    341.
##  3 JFK         760      128 ATL    356.
##  4 EWR         746      120 ATL    373
##  5 LGA         762      126 ATL    363.
##  6 LGA         762      126 ATL    363.
##  7 JFK         760      126 ATL    362.
##  8 LGA         762      132 ATL    346.
##  9 LGA         762      123 ATL    372.
## 10 LGA         762      129 ATL    354.
## # ... with 17,205 more rows
```

How many passengers can a plane hold before needing another engine?

This question uses the `planes` data set. Let's load that into our working environment so we can look at it.

```
planes <- planes
```

1. Examine the variable `engines` using `table()` and `class`. What is it's data type?

```
table(planes$engines)
```

```
##
##     1    2    3    4
##    27 3288    3    4
```

```
class(planes$engines)
```

```
## [1] "integer"
```

The number of engines is an integer variable with values between 1 and 4.

2. There are too few planes with more than 2 engines. Recode all records with 4 engines to a value of 3. *Hint: Revisit lesson 04.* Create a `table` of this variable again to ensure that all 4's are now 3's.

```
planes$engines[planes$engines==4] <- 3
table(planes$engines)
```

```
##
##    1    2    3
##   27 3288    7
```

3. Create a new factor variable `num_engines` from `engines` with labels "one", "two", "three+".

```
planes$num_engines <- factor(planes$engines, labels=c("one", "two", "three+"))
```

4. Create a two-way `table` of `engines` against `new_engines` to confirm that this new factor variable was created correctly.

```
table(planes$num_engines, planes$engines)
```

```
##
##             1    2    3
##   one      27    0    0
##   two       0 3288    0
##   three+    0    0    7
```

5. Use `dplyr` chaining magic to…

- take the planes data set *and then…*
- `group_by`the `num_engines` *and then…*
- use `summarise` to create three new variables:
  - `ave_seats` as the `mean()` number of `seats`
  - `min_seats` as the `min()` number of `seats`
  - `max_seats` as the `max()` number of `seats`

```
planes %>% group_by(num_engines) %>% summarise(ave_seats = mean(seats),
                                               min_seats = min(seats),
                                               max_seats = max(seats))
```

```
## # A tibble: 3 x 4
##   num_engines ave_seats min_seats max_seats
##   <fct>           <dbl>     <dbl>     <dbl>
## 1 one              3.78         2        16
## 2 two            155.           6       400
## 3 three+         243.           2       450
```