Lab- Unit 2 - Solutions

Exercise 1

1a Using Select

```
makes <- select(vehicles, make)
nrow(distinct(makes))    # gives the number of makes of vehicles
```

```
## [1] 128
```

There are 128 different car manufacturers in the dataset

1b filter-arrange-mutate

The following code gives the all cars from 1997 sorted by highway gas mpg and with an extra column which gives the average between city and highway mpg

```
new_vehicles <- vehicles %>% filter(year=="1997") %>%
  arrange(hwy) %>% mutate(average=0.5*(hwy+cty))
```

1c - find the worst mpg of subset

When I performed the filters to find the ID of the car with 2 - Wheel drive (front or rear) that had cty > 20 and the minimum hwy mpg I actually found a that there are a number of cars that have hwy of 22 so the answer is not unique - there's about 20 cars all with hwy $= 22$ which are all equally worst ! the code is below

```
wrse_mt_20 <- filter(vehicles,
                drive == "Rear-Wheel Drive" | drive == "Front-Wheel Drive",cty > 20) %>%
                filter(hwy==min(hwy))
select(wrse_mt_20,id)
```

```
## # A tibble: 20 x 1
##       id
##    <int>
##  1 29781
##  2 25578
##  3 29764
##  4 25122
##  5 25607
##  6 29788
##  7   705
##  8 25582
##  9 29769
## 10 25127
## 11 25625
## 12 29791
## 13 11461
## 14 11462
## 15 13160
## 16 13887
## 17 11487
```

```
## 18 11488
## 19 13180
## 20 13909
```

1d - Pipe Operator

There was a mispelling it should be MPG (not MHG) – sorry! To do this with temporary variables

```
acrs <- filter(vehicles,make=="Acura",year=="2015")
final_result  <- filter(acrs,hwy==min(hwy))
select(final_result,model)[[1]]
```

```
## [1] "RDX 4WD"
```

Note the double brackets to get just the value in the cell... this is a useful method note that you cannot do everything in one filter because you need to do on min on a subset of the data!

Now let's do it with Nested functions

```
select(filter(filter(vehicles,make=="Acura",year=="2015"),hwy==min(hwy)),model)[[1]]
```

```
## [1] "RDX 4WD"
```

gives same result. Finally lets do it with Pipe operator

```
acuras_2015 <- vehicles %>% filter(make=="Acura",year=='2015') %>%
  filter(hwy==min(hwy)) %>% select(model)

acuras_2015[[1]]
```

```
## [1] "RDX 4WD"
```

pipe operator is the best way to do it, especially for more complex queries where it can get confusing...

1d - bonus: measure timings

I wrote three functions, fone, ftwo, fthree for the three approaches above

```
fone <- function(cars) {
acrs <- filter(cars,make=="Acura",year=="2015")
final_result  <- filter(acrs,hwy==min(hwy))
select(final_result,model)[[1]]
}

start.time <- Sys.time()

replicate(1000,fone(vehicles))

end.time <- Sys.time()
time.taken <- end.time - start.time
```

```
time.taken
```

```
## Time difference of 11.59413 secs
```

now for the nested approach

```
ftwo <- function(cars) {
  select(filter(filter(cars,make=="Acura",year=="2015"),hwy==min(hwy)),model)[[1]]
```

```
}

start.time <- Sys.time()

replicate(1000,ftwo(vehicles))

end.time <- Sys.time()
time.taken <- end.time - start.time
```

```
time.taken
```

```
## Time difference of 11.43559 secs
```

finally using piping

```
fthree <- function(cars) {
acuras_2015 <- cars %>% filter(make=="Acura",year=='2015') %>%
  filter(hwy==min(hwy)) %>% select(model)
acuras_2015[[1]]
}

start.time <- Sys.time()

replicate(1000,fthree(vehicles))

end.time <- Sys.time()
time.taken <- end.time - start.time
```

```
time.taken
```

```
## Time difference of 11.60987 secs
```

Conclusion: there doesn't seeem to be any big difference in efficiency for these 3 methods!

1e - challenge

here is the function I wrote below to do this and the result for Honda in 1995

```
mostefficient <- function(make_choice,year_choice) {
sub <- filter(vehicles,make==make_choice,year==year_choice)
hvar <- filter(sub,hwy == max(hwy))
hvar[['model']]
}

mostefficient('Honda','1995')
```

```
## [1] "Civic HB VX"
```

Exercise 2 - Flights

2a-

```
library('nycflights13')
colnames(flights)
```

```
##  [1] "year"           "month"          "day"            "dep_time"
##  [5] "sched_dep_time" "dep_delay"      "arr_time"       "sched_arr_time"
##  [9] "arr_delay"      "carrier"        "flight"         "tailnum"
## [13] "origin"         "dest"           "air_time"       "distance"
## [17] "hour"           "minute"         "time_hour"
```

2b Highest number of delayed departures

```
has_most_delays <- flights %>%
  group_by(carrier) %>%
  filter(dep_delay > 0) %>%
  summarize(num_delay = n()) %>%
  filter(num_delay == max(num_delay)) %>%
  select(carrier)
has_most_delays
```

```
## # A tibble: 1 x 1
##   carrier
##   <chr>
## 1 UA
```

UA is an abbreviation.

2c - left_join

we need to join the dataframe airlines to the dataframe has_most_delays to find the full name of the carrier (as if you didn't know!)

```
most_delayed_name <- has_most_delays %>%
  left_join(airlines,by='carrier') %>%
  select(name)
most_delayed_name
```

```
## # A tibble: 1 x 1
##   name
##   <chr>
## 1 United Air Lines Inc.
```

2d and 2e

```
flights %>% group_by(month) %>%
  summarize(delay = mean(arr_delay,na.rm=TRUE)) %>% # remove NAs
  filter(delay == max(delay)) %>%
  select(month)
```

```
## # A tibble: 1 x 1
##   month
##   <int>
## 1     7
```

the 7 indicates July