MSCS 264: Homework #4

Due Fri, Feb 24 at 11pm

AL Ashir Intisar

This homework focuses on mutate and using pipes to connect multiple data transformations together.

```
library(tidyverse)
library(lubridate)
library(nycflights13)
```

One of the advantages of pipes is that it makes code very readable. For example, this chunk of code

```
mhealth_full %>%
  filter(Group == "By Age") %>%
  rename(date = `Time Period Start Date`) %>%
  select(date, Subgroup, Value)
```

can be read as "Start with mhealth_full dataset, then filter to include only Group"By Age", then rename the Time Period Start Date variable, then select only the variables date, subgroup and value."

1. Consider our diamonds dataset. We usually start with code that looks similar to this. Write a sentence that states what this code does (similar to the style above). (Hint: use ?slice_sample to check out help menu).

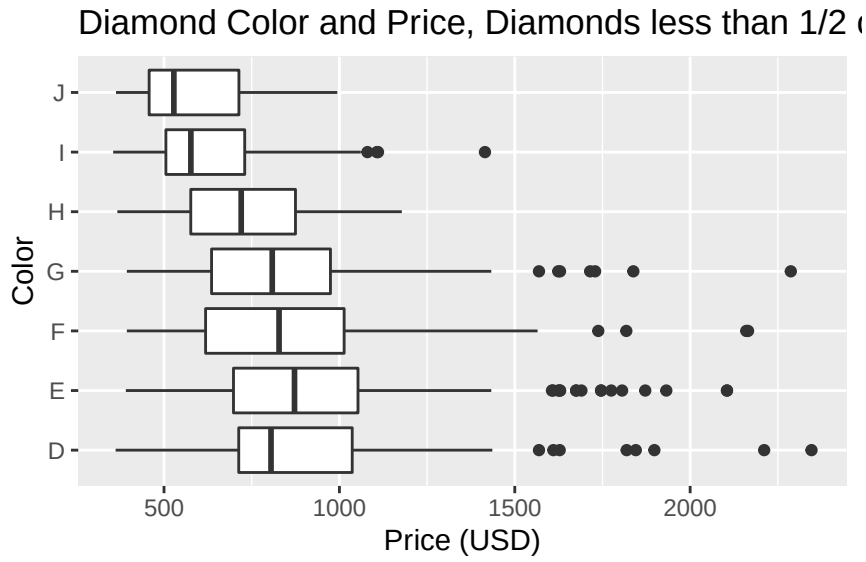
```
smaller <- diamonds %>%
  filter(carat <= 3) %>%
  slice_sample(n = 1000)
```

Ans: slice_sample() function randomly selects rows from the dataset provided where n = number of rows to be selected. The pipeline takles in the dataset diamonds and then filters out the records (rows) with more than 3 carat diamonds and then takes a random 1000 row sample of the filtered data.

2. Your friend wrote the following chunks of code to create a plot. Use pipes to connect from "diamonds" in #1 above, all the way through the plot. Modify the order of steps so that your graph includes the prices for 1000 diamonds that are 1/2 carat or less.

```
smaller_graph <- diamonds |>
  filter(carat <= 0.5) |>
  slice_sample(n = 1000)|>
  ggplot() +
    geom_boxplot(aes(y = color, x = price)) +
    labs(y = "Color", x = "Price (USD)", title = "Diamond Color and Price, Diamonds less than 1/2 carat
```

```
smaller_graph
```



3. Using the diamonds dataset, complete the following steps:

- restrict to diamonds between 1 and 2 carats (include diamonds exactly 1 carat, but not exactly 2 carats) and color D, E, or F.

```
diamonds_3 <- diamonds|>
  filter(carat >=1, carat < 2, color == c("D", "E", "F"))
```

```
head(diamonds_3)
```

```
## # A tibble: 6 x 10
##   carat cut     color clarity depth table price     x     y     z
##   <dbl> <ord>   <ord> <ord>   <dbl> <dbl> <int> <dbl> <dbl> <dbl>
## 1  1.01 Premium F     I1       61.8    60  2781  6.39  6.36  3.94
## 2  1.01 Fair    E     I1       64.5    58  2788  6.29  6.21  4.03
## 3  1.01 Fair    E     SI2      67.4    60  2797  6.19  6.05  4.13
## 4  1    Fair    E     SI2      65.8    58  2948  6.28  6.16  4.09
## 5  1.05 Premium E     I1       61.4    58  2964  6.53  6.46  3.99
## 6  1.01 Fair    D     SI2      64.6    56  3003  6.31  6.24  4.05
```

- create a variable that indicates if the diamond costs more than $10000.

```
diamonds_3 <- diamonds_3|>
  mutate("Cost > $10,000" = price>10000)|>
  arrange(desc(price))
```

```
head(diamonds_3)
```

```
## # A tibble: 6 x 11
##   carat cut       color clarity depth table price     x     y     z Cost > $10~1
##   <dbl> <ord>     <ord> <ord>   <dbl> <dbl> <int> <dbl> <dbl> <dbl> <lgl>
## 1  1.71 Premium   F     VS2      62.3    59 18791  7.57  7.53  4.7  TRUE
## 2  1.51 Ideal     E     VS1      61.5    57 18729  7.34  7.4   4.53 TRUE
## 3  1.42 Ideal     F     VVS1     60.8    56 18682  7.25  7.32  4.43 TRUE
## 4  1.49 Ideal     F     VVS2     61.1    58 18614  7.36  7.38  4.5  TRUE
## 5  1.6  Ideal     F     VS1      60.5    57 18571  7.6   7.63  4.61 TRUE
```

```
## 6  1.72 Very Good E     VS2      63.4    56 18557  7.65  7.55  4.82 TRUE
## # ... with abbreviated variable name 1: 'Cost > $10,000'
```

- create a variable that indicates if a diamond is in one of the top clarity categories (hint: high_clar = ifelse(clarity %in% c("VVS2", "VVS1", "IF"), "high clarity", "low clarity"))

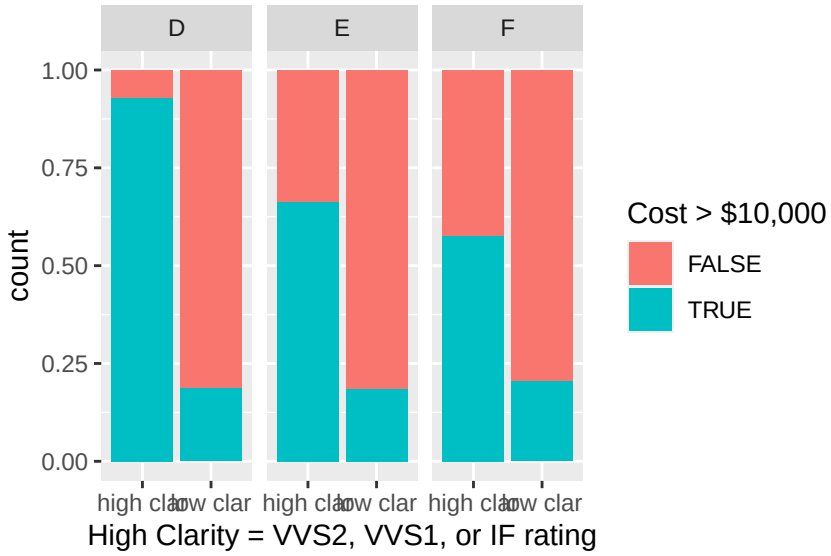
```
diamonds_3 <- diamonds_3|>
  mutate("high clar" = high_clar <- ifelse(diamonds_3$clarity %in% c("VVS2", "VVS1", "IF"), "high clar"
```

```
head(diamonds_3)
```

```
## # A tibble: 6 x 12
##   carat cut    color clarity depth table price     x     y     z Cost ~1 high ~2
##   <dbl> <ord>  <ord> <ord>   <dbl> <dbl> <int> <dbl> <dbl> <dbl> <lgl>   <chr>
## 1  1.71 Premi~ F     VS2      62.3    59 18791  7.57  7.53  4.7  TRUE    low cl~
## 2  1.51 Ideal  E     VS1      61.5    57 18729  7.34  7.4   4.53 TRUE    low cl~
## 3  1.42 Ideal  F     VVS1     60.8    56 18682  7.25  7.32  4.43 TRUE    high c~
## 4  1.49 Ideal  F     VVS2     61.1    58 18614  7.36  7.38  4.5  TRUE    high c~
## 5  1.6  Ideal  F     VS1      60.5    57 18571  7.6   7.63  4.61 TRUE    low cl~
## 6  1.72 Very ~ E     VS2      63.4    56 18557  7.65  7.55  4.82 TRUE    low cl~
## # ... with abbreviated variable names 1: 'Cost > $10,000', 2: 'high clar'
```

- create a bar chart that looks like "hw4_clarity_cost_color.png" in the homework images folder.

```
diamonds_3|>
  ggplot()+
  geom_bar(mapping = aes(`high clar`, fill = `Cost > $10,000`), position = "fill")+
  facet_wrap(~color)+
  xlab("High Clarity = VVS2, VVS1, or IF rating")
```



For our next problem, we use the basketball data again.

4. In the previous homework, we found the top scoring centers using the following steps. Rewrite this code using pipes. Also add a line using slice_max to print the top 3 scoring centers.

```
bball3 <- bball|>
  filter(Pos == "C")|>
  select(Player, Tm, PTS)|>
  arrange(desc(PTS))|>
```

```
  rename(Team = Tm, "Points Scored" = PTS)

slice_max(bball3, `Points Scored`, n = 3)
```

```
## # A tibble: 3 x 3
##   Player             Team  `Points Scored`
##   <chr>              <chr>           <dbl>
## 1 Joel Embiid        PHI              2079
## 2 Nikola Jokić       DEN              2004
## 3 Karl-Anthony Towns MIN              1818
```

Map US trends in COVID vaccinations

In this portion, we are going to replicate the plot of covid vaccinations found here: CDC webpage for tracking COVID vaccinations. (Note that our dataset only goes through 2022, while the plot on the website goes through 2023.)

Read the data into R using the chunk below.

```
library(tidyverse)
library(lubridate)
vaccinations <- read_csv("~/Mscs 264 S23/Class/Data/vaccinations.csv")
vaccinations
```

```
## # A tibble: 437 x 17
##    Date ~1 Date  Locat~2 Total~3 Daily~4 Total~5 Peopl~6 7-Day~7 7-Day~8 Total~9
##    <chr>   <chr> <chr>     <dbl>   <dbl>   <dbl>   <dbl>   <dbl>   <dbl> <chr>
##  1 Admin   12/1~ US         4535    4378   32801   28006   14003   16400 N/A
##  2 Admin   12/1~ US        49635   47310   82436   75316   25105   27478 N/A
##  3 Admin   12/1~ US       159673  154288  242109  229604   57401   60527 N/A
##  4 Admin   12/1~ US       272265  265169  514374  494773   98955  102874 N/A
##  5 Admin   12/1~ US       415855  407331  930229  902104  150351  155038 N/A
##  6 Admin   12/1~ US       181658  177753 1111887 1079857  154265  158841 N/A
##  7 Admin   12/2~ US       105049  103148 1216936 1183005  165625  169810 N/A
##  8 Admin   12/2~ US       381433  374954 1598369 1557959  218565  223652 N/A
##  9 Admin   12/2~ US       447197  438765 2045566 1996724  274487  280447 N/A
## 10 Admin   12/2~ US       574485  562090 2620051 2558814  332744  339706 N/A
## # ... with 427 more rows, 7 more variables:
## #   `7-Day Avg Total Doses Administered Daily Change` <chr>,
## #   `Daily Count of People Fully Vaccinated` <dbl>,
## #   `People Fully Vaccinated Cumulative` <dbl>,
## #   `7-Day Avg Daily Count of People Fully Vaccinated` <dbl>,
## #   `Daily Count People Receiving a Booster Dose` <dbl>,
## #   `People Receiving a Booster Dose Cumulative` <dbl>, ...
## # i Use `print(n = ...)` to see more rows, and `colnames()` to see all variable names
```

5. Using pipes, apply the following transformations to the vaccinations dataset. (Hint: do them one at a time, checking each time to see if it worked, then pipe to the next one.)
   - `rename` the variables "Total Doses Administered Daily" to "total_daily" and "7-Day Avg Total Doses Daily" to "total_7day_avg"
   - `select` only the relevant columns: Date, total_daily, total_7day_avg
   - `mutate` new variaables, total_daily_millions and total_7day_avg_millions by dividing by 1000000.

```
vaccinations <- vaccinations|>
  rename(total_daily = `Total Doses Administered Daily`, total_7day_avg = `7-Day Avg Total Doses Daily`
```

```
  select(Date, total_daily, total_7day_avg)|>
  mutate(total_daily_millions = total_daily/1000000, total_7day_avg_millions = total_7day_avg/1000000)
```

```
head(vaccinations)
```

```
## # A tibble: 6 x 5
##   Date     total_daily total_7day_avg total_daily_millions total_7day_avg_mill~1
##   <chr>          <dbl>          <dbl>                <dbl>                 <dbl>
## 1 12/14/20        4535          16400              0.00454                0.0164
## 2 12/15/20       49635          27478              0.0496                 0.0275
## 3 12/16/20      159673          60527              0.160                  0.0605
## 4 12/17/20      272265         102874              0.272                  0.103
## 5 12/18/20      415855         155038              0.416                  0.155
## 6 12/19/20      181658         158841              0.182                  0.159
## # ... with abbreviated variable name 1: total_7day_avg_millions
```

6. Also create "Date = mdy(Date)" using mutate. What does this do? (Hint: Print the dataset to the console before and after adding this mutate code.)

```
vaccinations <- vaccinations|>
  mutate(Date = mdy(Date))
```

```
head(vaccinations)
```

```
## # A tibble: 6 x 5
##   Date       total_daily total_7day_avg total_daily_millions total_7day_avg_mi~1
##   <date>           <dbl>          <dbl>                <dbl>               <dbl>
## 1 2020-12-14        4535          16400              0.00454              0.0164
## 2 2020-12-15       49635          27478              0.0496               0.0275
## 3 2020-12-16      159673          60527              0.160                0.0605
## 4 2020-12-17      272265         102874              0.272                0.103
## 5 2020-12-18      415855         155038              0.416                0.155
## 6 2020-12-19      181658         158841              0.182                0.159
## # ... with abbreviated variable name 1: total_7day_avg_millions
```

Ans: The function mdy() reverses the order of how the date is represented. NOw it is presented as Year-Month-Date

7. Your resulting dataset should look like vax_tidy (see below). (You can use your dataset from above, or just use vax_tidy if you're not confident about #5-6). I've put code to start your plot here. Add the line for 7-day average to the plot using geom_line. You can see a list of colors available here, and experiment to get the width of the line to look nice.

```
vax_tidy <- read_csv("~/Mscs 264 S23/Class/Data/vax_tidy.csv")
```

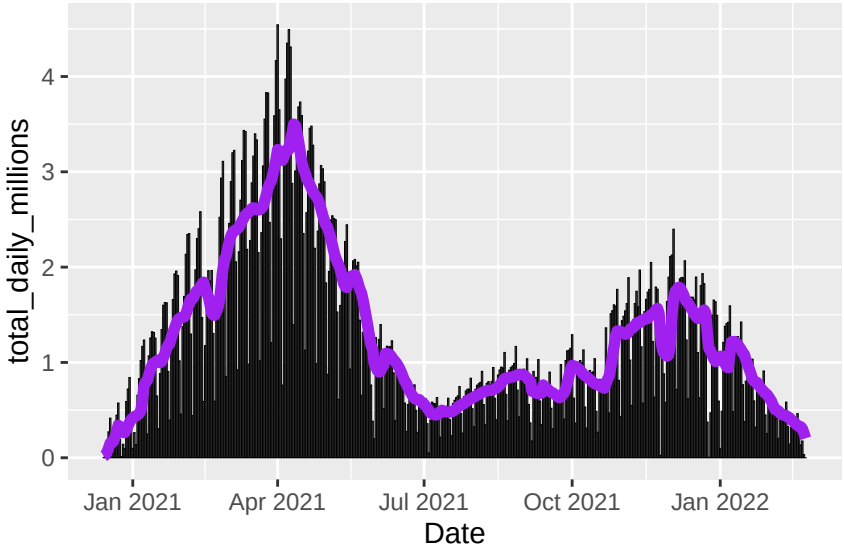
```
## New names:
## Rows: 437 Columns: 6
## -- Column specification
## -------------------------------------------------------- Delimiter: "," dbl
## (5): ...1, total_daily, total_7day_avg, total_daily_millions, total_7da... date
## (1): Date
## i Use `spec()` to retrieve the full column specification for this data. i
## Specify the column types or set `show_col_types = FALSE` to quiet this message.
## * `` -> `...1`
```

```
vax_tidy
```

```
## # A tibble: 437 x 6
```

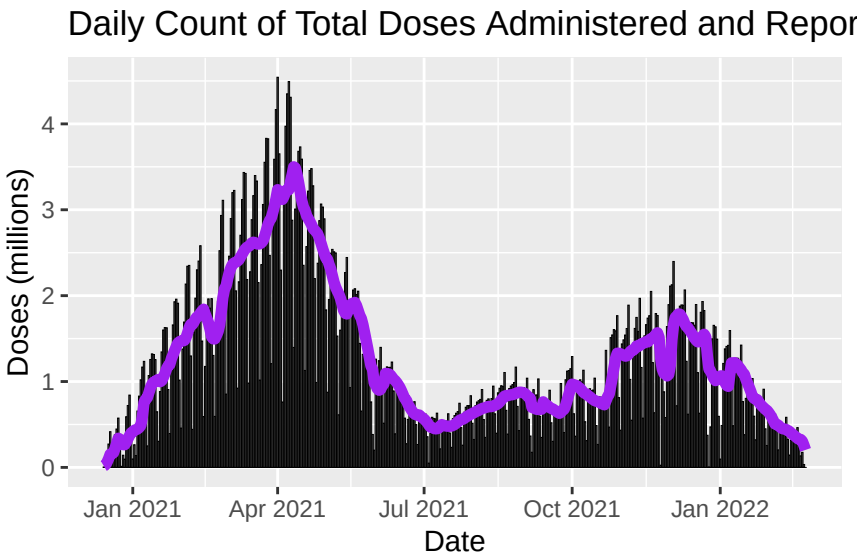
```
##     ...1 Date       total_daily total_7day_avg total_daily_millions total_7day~1
##    <dbl> <date>           <dbl>          <dbl>                <dbl>        <dbl>
##  1     1 2020-12-14        4535          16400              0.00454       0.0164
##  2     2 2020-12-15       49635          27478              0.0496        0.0275
##  3     3 2020-12-16      159673          60527              0.160         0.0605
##  4     4 2020-12-17      272265         102874              0.272         0.103
##  5     5 2020-12-18      415855         155038              0.416         0.155
##  6     6 2020-12-19      181658         158841              0.182         0.159
##  7     7 2020-12-20      105049         169810              0.105         0.170
##  8     8 2020-12-21      381433         223652              0.381         0.224
##  9     9 2020-12-22      447197         280447              0.447         0.280
## 10    10 2020-12-23      574485         339706              0.574         0.340
## # ... with 427 more rows, and abbreviated variable name
## #   1: total_7day_avg_millions
## # i Use 'print(n = ...)' to see more rows
```

```
ggplot(vaccinations) +
  geom_col(aes(Date, total_daily_millions), color = "black", size = .05)+
  geom_line(aes(Date, total_7day_avg_millions), color = "purple",  size = 2)
```



8. Add labs to your plot for the y axis and a title. See "hw4_vax_plot.png" in the Homework > images folder as an example of what your resulting plot should look like.

```
ggplot(vaccinations) +
  geom_col(aes(Date, total_daily_millions), color = "black", size = .05)+
  geom_line(aes(Date, total_7day_avg_millions), color = "purple",  size = 2)+
  labs(title = "Daily Count of Total Doses Administered and Reported")+
  xlab("Date")+
  ylab("Doses (millions)")
```


Daily Count of Total Doses Administered and Repor
Doses (millions)
0
1
2
3
4
Jan 2021
Apr 2021
Jul 2021
Oct 2021
Jan 2022
Date