Homework 0

Rudy Martinez

September 2020

This homework uses `txhousing` dataset from `ggplot2` package, which is a part of the `tidyverse`. Additionally, it uses Consumer Price Index (CPI) dataset, which I downloaded from the BLS website (https://data.bls.gov/timeseries/CUUR0000SA0) and cleaned. I will provide the code for cleaning CPI data in the solution to this homework.

`txhousing` consists of the following variables:

```
names(txhousing)
```

```
## [1] "city"      "year"      "month"     "sales"     "volume"    "median"
## [7] "listings"  "inventory" "date"
```

Before you start working on this homework, study the variables in `txhousing` as well as the structure of the dataset by typing this command in your console.

```
help(txhousing)
```

Take a peek at the data by typing:

```
head(txhousing)
```

```
## # A tibble: 6 x 9
##   city     year month sales   volume median listings inventory  date
##   <chr>   <int> <int> <dbl>    <dbl>  <dbl>    <dbl>     <dbl> <dbl>
## 1 Abilene  2000     1    72  5380000  71400      701       6.3 2000
## 2 Abilene  2000     2    98  6505000  58700      746       6.6 2000.
## 3 Abilene  2000     3   130  9285000  58100      784       6.8 2000.
## 4 Abilene  2000     4    98  9730000  68600      785       6.9 2000.
## 5 Abilene  2000     5   141 10590000  67300      794       6.8 2000.
## 6 Abilene  2000     6   156 13910000  66900      780       6.6 2000.
```

Save the CPI dataset in your project folder where you have also saved this homework file. Read the CPI dataset into this session. It consists of the following columns:

```
cpi = read_rds('cpi.rds') #read your rds file here
names(cpi)
```

```
## [1] "year"       "month_name" "cpi"
```

Take a look at the first few observations by using `head()` function.

```
head(cpi)
```

```
## # A tibble: 6 x 3
##    year month_name   cpi
##   <dbl> <chr>      <dbl>
## 1  1999 jan         164.
## 2  1999 feb         164.
## 3  1999 mar         165
## 4  1999 apr         166.
## 5  1999 may         166.
## 6  1999 jun         166.
```

This homework consists of 10 questions and each carries one point. Your objective is to reproduce the output shown in the HTML file for Q1 through Q9. For Q10 just print the name of the city and the value.

Dollar values over a long time series make comparison difficult due to inflation. $100 in January 2000 is worth $154 in July 2020. In `txhousing`, there are two variables—`volume` and `median`—which are specified in unadjusted USD. Questions 1 to 4 are designed to inflation-adjust these two variables to July 2020 dollars.

Q1

Create a new data frame `month_map` with 12 rows and 2 columns titled `month_name` and `month`. `month_name` in `month_map` should have only the unique values from `month_name` column in CPI data frame. `month` column should contain the month numbers from 1 to 12.

Try to not type out this data frame manually and instead try to do it algorithmically. This is how it will look:

```
month_map = data.frame(month_name = unique(cpi$month_name),
                month = c(1:12))

head(month_map, 4)
```

```
##   month_name month
## 1        jan     1
## 2        feb     2
## 3        mar     3
## 4        apr     4
```

Q2

Merge `month_map` to the CPI data frame. Explicitly identify the common key variable on which you will perform the merge. Store the resulting merged data frame as `cpi_merge`. **Print the first six rows.**

Here are the first six rows of `cpi_merge`:

```
cpi_merge = cpi %>%
  inner_join(month_map, by = "month_name")

head(cpi_merge)
```

```
## # A tibble: 6 x 4
##    year month_name   cpi month
##   <dbl> <chr>      <dbl> <int>
## 1  1999 jan         164.     1
## 2  1999 feb         164.     2
## 3  1999 mar         165      3
## 4  1999 apr         166.     4
## 5  1999 may         166.     5
## 6  1999 jun         166.     6
```

Q3

We want to add a column to the `txhousing` data frame that holds the information on the CPI. Rather than altering `txhousing`, create a new data frame `housing` by merging `txhousing` and `cpi_merge`. The common keys for merging are `year` and `month`. Note that the resulting data frame is essentially `txhousing` with *just one more column* of CPI.

Here are the top six rows of `housing`:

```
housing = txhousing %>%
  inner_join(cpi_merge, by = c("year", "month"))

housing %>% select(-c("month_name")) %>% head()
```

```
## # A tibble: 6 x 10
##   city     year month sales   volume median listings inventory  date   cpi
##   <chr>   <dbl> <int> <dbl>    <dbl>  <dbl>    <dbl>     <dbl> <dbl> <dbl>
## 1 Abilene  2000     1    72  5380000  71400      701       6.3 2000   169.
## 2 Abilene  2000     2    98  6505000  58700      746       6.6 2000.  170.
## 3 Abilene  2000     3   130  9285000  58100      784       6.8 2000.  171.
## 4 Abilene  2000     4    98  9730000  68600      785       6.9 2000.  171.
## 5 Abilene  2000     5   141 10590000  67300      794       6.8 2000.  172.
## 6 Abilene  2000     6   156 13910000  66900      780       6.6 2000.  172.
```

Q4

Modify `housing` by adding these new columns:

1. `cpi_latest` - Contains the CPI of July 2020. This entire column will have the same value appearing in each cell.
2. `multiplier` - Ratio of `cpi_latest` and `cpi`
3. `volume_adj` - Adjusted volume as a product of `volume` and `multiplier`
4. `median_adj` - Adjusted median sale price as a product of `median` and `multiplier`

Here I show the top six rows with only a select few columns to help you ensure your output resembles this:

```
jul_2020 = filter(cpi, year == 2020, month_name == 'jul') # creates a unique data frame with July 2020

housing = mutate(housing,
                 cpi_latest = jul_2020$cpi,
```

```
                multiplier = cpi_latest / cpi,
                volume_adj = volume * multiplier,
                median_adj = median * multiplier)

housing %>% select(c("city", "year", "month", "volume", "median", "cpi_latest", "multiplier", "volume_a
```

```
## # A tibble: 6 x 9
##   city     year month  volume median cpi_latest multiplier volume_adj median_adj
##   <chr>   <dbl> <int>   <dbl>  <dbl>      <dbl>      <dbl>      <dbl>      <dbl>
## 1 Abilene  2000     1  5.38e6  71400       259.       1.53   8258077.    109596.
## 2 Abilene  2000     2  6.50e6  58700       259.       1.53   9926101.     89571.
## 3 Abilene  2000     3  9.28e6  58100       259.       1.51  14052294.     87931.
## 4 Abilene  2000     4  9.73e6  68600       259.       1.51  14717179.    103761.
## 5 Abilene  2000     5  1.06e7  67300       259.       1.51  15999298.    101676.
## 6 Abilene  2000     6  1.39e7  66900       259.       1.50  20905423.    100544.
```

Q5

Using `housing` from Q4, create a new data frame `housing_sum1` with this information for each `city` *across all years and months*:

1. Maximum and minimum `volume_adj`
2. Maximum and minimum `median_adj` sale price

Hint: If you group by `city`*, you will get the summary across all the years and months.*

Merge `housing_sum1` into `housing` by `city` and save it as a new dataset `housing_1`.

Here I show *first* six rows of `housing_1` and only a select columns:

```
housing_sum1 = housing %>%
  group_by(city) %>%
    summarize(volume_adj_max = max(volume_adj, na.rm = TRUE),
              volume_adj_min = min(volume_adj, na.rm = TRUE),
              median_adj_max = max(median_adj, na.rm = TRUE),
              median_adj_min = min(median_adj, na.rm = TRUE),
              .groups = "drop")

housing_1 = housing %>%
  inner_join(housing_sum1, by = "city")

housing_1 %>% select(c("city", "year", "month", "volume_adj_max", "volume_adj_min", "median_adj_max", "
```

```
## # A tibble: 6 x 7
##   city    year month volume_adj_max volume_adj_min median_adj_max median_adj_min
##   <chr>  <dbl> <int>          <dbl>          <dbl>          <dbl>          <dbl>
## 1 Abile~  2000     1      49773624.       7678915.        161440.         84890.
## 2 Abile~  2000     2      49773624.       7678915.        161440.         84890.
## 3 Abile~  2000     3      49773624.       7678915.        161440.         84890.
## 4 Abile~  2000     4      49773624.       7678915.        161440.         84890.
## 5 Abile~  2000     5      49773624.       7678915.        161440.         84890.
## 6 Abile~  2000     6      49773624.       7678915.        161440.         84890.
```

Here I show *last* six rows of `housing_1` and only a select columns:

```
housing_1 %>% select(c("city", "year", "month", "volume_adj_max", "volume_adj_min", "median_adj_max", "
```

```
## # A tibble: 6 x 7
##   city    year month volume_adj_max volume_adj_min median_adj_max median_adj_min
##   <chr>  <dbl> <int>          <dbl>          <dbl>          <dbl>          <dbl>
## 1 Wichi~  2015     2      33832205.       8337061.        147084.         84452.
## 2 Wichi~  2015     3      33832205.       8337061.        147084.         84452.
## 3 Wichi~  2015     4      33832205.       8337061.        147084.         84452.
## 4 Wichi~  2015     5      33832205.       8337061.        147084.         84452.
## 5 Wichi~  2015     6      33832205.       8337061.        147084.         84452.
## 6 Wichi~  2015     7      33832205.       8337061.        147084.         84452.
```

Q6

1. Using `housing_1` from Q5, create a new data frame `housing_min` which will retain only the rows of `housing_1` where `volume_adj` of a `city` was equal to the minimum adjusted volume.

Here I show first six rows of `housing_min` and only a select columns:

```
housing_min = filter(housing_1, volume_adj == volume_adj_min)

housing_min %>% select(c("city", "year", "month", "volume_adj", "volume_adj_min", "volume_adj_max")) %>%
```

```
## # A tibble: 6 x 6
##   city       year month volume_adj volume_adj_min volume_adj_max
##   <chr>     <dbl> <int>      <dbl>          <dbl>          <dbl>
## 1 Abilene    2003     1   7678915.       7678915.      49773624.
## 2 Amarillo   2005    10   9397614.       9397614.      67596289.
## 3 Arlington  2011     1  30008080.      30008080.     137867140.
## 4 Austin     2009     1 252191554.     252191554.    1248942028.
## 5 Bay Area   2000     1  45009066.      45009066.     214278169.
## 6 Beaumont   2001     1  13117820.      13117820.      57413784.
```

2. Using `housing_1` from Q5, create a new data frame `housing_max` which will retain only the rows of `housing_1` where `median_adj` of a `city` was equal to the maximum adjusted median sale price.

Here I show first six rows of `housing_max` and only a select columns:

```
housing_max = filter(housing_1, median_adj == median_adj_max)

housing_max %>% select(c("city", "year", "month", "median_adj", "median_adj_max", "median_adj_min")) %>%
```

```
## # A tibble: 6 x 6
##   city       year month median_adj median_adj_max median_adj_min
##   <chr>     <dbl> <int>      <dbl>          <dbl>          <dbl>
## 1 Abilene    2015     7    161440.        161440.         84890.
## 2 Amarillo   2015     5    172040.        172040.        113162.
## 3 Arlington  2015     6    195435.        195435.        131800.
## 4 Austin     2015     4    296007.        296007.        200091.
## 5 Bay Area   2015     7    218004.        218004.        154570.
## 6 Beaumont   2010     1    195862.        195862.        108773.
```

Q7

Using `housing_1` from Q5, create a new data frame `housing_sum2` with this information for each `year` and `month` pair *across all cities*:

1. Median listings
2. Median sales

Hint: If even a single value for `listings` or `sales` of a `city` is `NA`, the median of that variable will be `NA`. In order to avoid this, use `na.rm = TRUE` argument in `median()`.

Here I show first six rows of `housing_sum2` and all the columns:

```
housing_sum2 = housing_1 %>%
  group_by(year, month) %>%
    summarize(listings_med = median(listings, na.rm = TRUE),
              sales_med = median(sales, na.rm = TRUE),
              .groups = "drop")

housing_sum2 %>% select(c("year", "month", "listings_med", "sales_med")) %>% head()
```

```
## # A tibble: 6 x 4
##    year month listings_med sales_med
##   <dbl> <int>        <dbl>     <dbl>
## 1  2000     1         972         99
## 2  2000     2         916.       134
## 3  2000     3         946.       167
## 4  2000     4         985        153
## 5  2000     5         978.       165
## 6  2000     6         864.       188
```

Merge `housing_sum2` into `housing_1` and save a new data frame `housing_2`.

Here I show first six rows of `housing_2` and some of the columns:

```
housing_2 = housing_1 %>%
  inner_join(housing_sum2, by = c("year", "month"))

housing_2 %>% select(c("city", "year", "month", "listings", "sales", "listings_med", "sales_med")) %>% h
```

```
## # A tibble: 6 x 7
##   city     year month listings sales listings_med sales_med
##   <chr>   <dbl> <int>    <dbl> <dbl>        <dbl>     <dbl>
## 1 Abilene  2000     1      701    72         972         99
## 2 Abilene  2000     2      746    98         916.       134
## 3 Abilene  2000     3      784   130         946.       167
## 4 Abilene  2000     4      785    98         985        153
## 5 Abilene  2000     5      794   141         978.       165
## 6 Abilene  2000     6      780   156         864.       188
```

Q8

Modify `housing_2` from Q7 to add these indicator variables (also called dummy variables):

1. `listings_ind` - If a city's `listings` is less than or equal to the median listings for that `year` and `month` across all the cities, the value should be `0` else it should be `1`.
2. `sales_ind` - If a city's `sales` is less than or equal to the median sales for that `year` and `month` across all the cities, the value should be `0` else it should be `1`.

Hint: This can be achieved using `ifelse()` function from R along with `mutate()` from `dplyr`

Here I show first six rows of `housing_2` and some of the columns:

```
housing_2 = mutate(housing_2,
                   listings_ind = ifelse(housing_2$listings <= housing_2$listings_med, 0, 1),
                   sales_ind = ifelse(housing_2$sales <= housing_2$sales_med, 0, 1))

housing_2 %>% select(c("city", "year", "month", "listings", "listings_med", "listings_ind", "sales", "sa
```

```
## # A tibble: 6 x 9
##   city   year month listings listings_med listings_ind sales sales_med sales_ind
##   <chr> <dbl> <int>    <dbl>        <dbl>        <dbl> <dbl>     <dbl>     <dbl>
## 1 Abil~  2000     1      701          972            0    72        99         0
## 2 Abil~  2000     2      746          916.           0    98       134         0
## 3 Abil~  2000     3      784          946.           0   130       167         0
## 4 Abil~  2000     4      785          985            0    98       153         0
## 5 Abil~  2000     5      794          978.           0   141       165         0
## 6 Abil~  2000     6      780          864.           0   156       188         0
```

Q9

Using `housing_2` from Q8, add a new variable `market_hotness` as follows:

listings_ind	sales_ind	market_hotness
0	0	Low
0	1	High
1	0	Very Low
1	1	Average

Here I show first six rows of `housing_2` and some of the columns:

```
housing_2 = mutate(housing_2,
                   market_hotness = ifelse((housing_2$listings_ind == 0) & (housing_2$sales_ind == 0),
                   ifelse((housing_2$listings_ind == 0) & (housing_2$sales_ind == 1), print("High"),
                   ifelse((housing_2$listings_ind == 1) & (housing_2$sales_ind == 0), print("Very Low")
                   ifelse((housing_2$listings_ind == 1) & (housing_2$sales_ind == 1), print("Average"),
                          )))))

housing_2 %>% select(c("city", "year", "month", "listings", "sales", "listings_ind", "sales_ind", "mark
```

```
## # A tibble: 6 x 8
##   city     year month listings sales listings_ind sales_ind market_hotness
##   <chr>   <dbl> <int>    <dbl> <dbl>        <dbl>     <dbl> <chr>
## 1 Abilene  2000     1      701    72            0         0 Low
## 2 Abilene  2000     2      746    98            0         0 Low
## 3 Abilene  2000     3      784   130            0         0 Low
## 4 Abilene  2000     4      785    98            0         0 Low
## 5 Abilene  2000     5      794   141            0         0 Low
## 6 Abilene  2000     6      780   156            0         0 Low
```

Q10

Which city has the highest *average* `median_adj` sale price and what is that price?

```
avg_median_adj_city = housing %>%
  group_by(city) %>%
    summarize(avg_median_adj = mean(median_adj, na.rm = TRUE),
              .groups = "drop")

avg_median_adj_city %>% arrange(desc(avg_median_adj)) %>% head(.,1)
```

```
## # A tibble: 1 x 2
##   city          avg_median_adj
##   <chr>                  <dbl>
## 1 Collin County        252325.
```