Problem Set 03: Data Wrangling

Contents

In this problem set we will practice some of the key data manipulation tasks for describing, summarizing, and working with data. We will specifically review the following functions from the `dplyr` package:

- `select`
- `mutate`
- `summarize`
- `arrange`
- `filter`
- `group_by`
- `%>%`

In addition we will review how to save objects using the `<-` assignment operator.

Set up

For this problem set, you are being given a R Markdown template file to fill your answers into! You can find this template file on the course Moodle page, it is titled **PS03_firstname_lastname.Rmd**

When you download this file, we recommend you edit the file name to have *your* first name and last name.

Fill in your answers below the corresponding question headers. Be sure to include text *and* code where necessary.

R Packages

For this problem set, you'll need to load the `ggplot2` and `dplyr` libraries.

```
library(ggplot2)
library(dplyr)
```

The data

After loading the two necessary libraries, run the following code to load and take a `glimpse` of the data:

```
data(txhousing)
glimpse(txhousing)
```

Make sure all the above code is included in your R markdown document, or code later on in your document may not run correctly when you knit to PDF

These data are about housing in Texas. Each row is monthly data for a given city in Texas in a given year. There are multiple years of data for each city.

Question 1

After running all the code above in the console, take a look at the data in the data viewer. You can accomplish this two different ways:

a) click on the name of the data in the Environment pane, or
b) type `View(txhousing)` in the **console**.

What is the last city listed in the data set (in row 8602)?

Question 2

Take a look at the variable descriptions by typing `?txhousing` into the **console**. What information does the `listings` variable contain?

Data Wrangling Review

select

Sometimes we want to pull out or extract just one or two columns of data. The following code will extract only the column in the data set for the variables `sales` and `volume`.

```
txhousing %>% select(sales, volume)
```

The `%>%` symbol is called the **piping** operator. Here, it takes the `txhousing` **data frame** and "pipes" or feeds it into the `select` function. You can think of the `%>%` symbol as the word "then".

Note that we did not use an assignment operator `<-` so we did not save these extracted, selected values. In the following code, we save the results of our selection by overwriting the original `txhousing` data frame.

By putting a `-` in front of the `date` variable, we tell R to **not** select the `date` variable. In other words, select all variables **except** the `date` variable.

```
txhousing <- txhousing %>% select(-date)
```

If you run the above code, and look at the `txhousing` data frame in the data viewer, you will see that the `date` variable has been removed.

filter

The filter function allows you to pull out just the **rows** (cases or observations) you want, based on some criteria in **one of the columns**. Think of the `filter` command as literally like a filter - it only lets through the rows that match your criteria and "fit" through the filter.

Imagine for instance that we wanted to reduce the size of the data set, and only include data for the city of Austin in 2012. This code below takes the `txhousing data`, then filters it to only include rows in which the `year` variable is 2012, **and** the `city` variable is Austin. The results are saved in a new data frame called `austin_12` that shows up in the **workspace**.

```
austin_12 <- txhousing %>% filter(year == 2012, city == "Austin")
```

Note that we use `==` to say that we only want values that **exactly match** 2012 and "Austin"

What if we wanted to restrict our data set to only years before 2004 and the City of Austin? Below we use the `<` symbol (meaning "less than") to accomplish this.

```
txhousing %>% filter(year < 2004, city == "Austin")
```

Note we did not save these results in a new data frame using the assignment operator, so no new data frame showed up in our Environment pane. Instead, the results print out immediately below the code chunk.

What if we wanted to use multiple cities? Below we use the `|` symbol to indicate that the city could be Austin **OR** Abilene. In this case, we **saved** these results as a new data frame called `aust_ab` that appears in your Environment pane.

```
aust_ab <- txhousing %>% filter(city == "Austin" | city == "Abilene")
```

mutate

The mutate function can add new columns (variables) to a data frame. For instance, the following will add a new column to the data called `vol_100k` that expresses volume in units of $100,000.

```
txhousing <- txhousing %>%
  mutate(vol_100k = volume/100000)
```

Note that we **SAVED** these results in new data frame called `txhousing`. This therefore **overwrote** the old `txhousing` data frame with a new version that contains this column. You can open the `txhousing` data frame in the viewer to confirm that it now contains this new column.

summarize

One of the first tasks in data analysis is often to get descriptive statistics that help to understand the central tendency and variability in the data. The `summarize()` command can take a column of data, and reduce it to a summary statistic.

For instance, the code below uses the `austin_12` data set made earlier to calculate the mean monthly number of `sales` in Austin in 2012.

```
austin_12 %>% summarize(x_bar_sales = mean(sales))
```

This code tells R to calculate the `mean` of the variable `sales`, and to save the results in a variable called `x_bar_sales`.

You can also calculate multiple summary statistics at once, and even for multiple variables. Below we also calculate a standard deviation `sd()` of `sales`, a minimum `min()` of the `volume` variable, a maximum `max()`

of the `volume` variable, etc. The `n()` calculates sample size... or the number of rows/ cases in the data frame.

```
austin_12 %>% summarize(x_bar_sales = mean(sales),
                        sd_sales = sd(sales),
                        min_vol = min(volume),
                        max_vol = max(volume),
                        mdn_list = median(listings),
                        iqr_list = IQR(listings),
                        sample_size = n())
```

```
## # A tibble: 1 x 7
##   x_bar_sales sd_sales   min_vol   max_vol mdn_list iqr_list sample_size
##         <dbl>    <dbl>     <dbl>     <dbl>    <dbl>    <dbl>       <int>
## 1       2127.     501. 265821275 791281075     7925     949.          12
```

Note that the names of the elements you calculate are user defined, like `xbar_sales`, `min_vol`, and `mdn_list`. You could customize these names as you like (but don't use spaces in your names).

arrange

You just determined that the maximum volume of monthly sales in Austin in 2012 was a total of $791,281,075 ....but what if you wanted to know *what month* that occurred in? Copy paste, and run the following into a new code chunk:

```
austin_12 %>%
  arrange(desc(volume))
```

This tells R to `arrange` the rows in the data set based on the `volume` column, and to do so in `desc`ending order. So the row with the $791,281,075 in sales is shown at the top! We can see that this `volume` occurred in the sixth month (June).

group_by

Sometimes we also want to calculate summary statistics for individual sub-groups in our data, instead of summarizing the *entire* data set. We can tell other functions in `dplyr` to perform their computations on a per-group basis by applying the `group_by` function.

The `group_by` function won't change how your data frame appears in the data viewer, or prints out in the console, but it will add some internal information which other `dplyr` functions will see and recognize, so they know to treat each group separately.

To define your sub-groups, just give the `group_by` function a comma-separated list of variable names which together define your sub-groups. For example, here we find the average number of monthly sales that occurred in Abilene and Austin, across all years in the data set, by grouping by the `city` variable. Note that we **use the `aust_ab` data frame** we created earlier, to restrict our analysis to those two cities.

```
aust_ab %>% group_by(city) %>%
  summarize(x_bar_sales = mean(sales))
```

```
## # A tibble: 2 x 2
##   city    x_bar_sales
##   <chr>         <dbl>
## 1 Abilene        150.
## 2 Austin        1997.
```

From the results we can see that there were an average of 150 sales per month in Abilene, and 1996 in Austin.

We can give R multiple variables to group by. For instance, we can find the mean sales for each month in each city, averaged across all the years, by grouping by the `city` **and** the `month` variables. R will create a

sub-group for each combination of values in the `city` variable and the `month` variable. So, you should get 24 mean values in this case.

```
aust_ab %>% group_by(city, month) %>%
  summarize(x_bar_sales = mean(sales))
```

If we run the above code, we should see that the mean number of sales in January, in Abilene, was 96 homes.

Independent practice

Basic syntax

This first set of questions will help you practice basic syntax. All you need to include is a Question header, and code for each.

Question 3

Write a code chunk to remove the inventory variable. Save the results in a data frame called `txhousing`. Confirm in the data viewer that the variable has been removed.

Question 4

Make a data set called `dallas_sub` that includes data only from the city of Dallas in 2012 & 2013.

Question 5

Add a column **to the `dallas_sub` data set** called `prct_sold` that calculates the percentage of `listings` that were `sold` (`sales/listings * 100`). Be sure to **save** the results also as a data frame called `dallas_sub`.

Question 6

Calculate the **average** percentage of listings that were sold in Dallas **in each month of the year** based on your `dallas_sub` data set. Save the results of the calculation in an data frame called `dallas_summary`.

Question 7

Arrange the `dallas_summary` in `desc`ending order based on the average percentage of listings that were sold in Dallas in 2012, so you can see **which month** had the greatest percentage of houses sold in Dallas on average from 2012-2013. You do not need to save the results.

More advanced wrangling

Please answer the following questions with text and/or code where appropriate. You may have to use multiple `dplyr` functions to answer each question. Think through the steps of how to get to the answer you are trying to find.

Question 8

Run the following code chunk. Study the code, and the output. Explain in your own words what this code chunk calculated.

```r
txhousing %>%
  filter(year == 2012 | year == 2013, city == "Dallas") %>%
  mutate(prct_sold = sales/listings *100) %>%
  group_by(month) %>%
  summarize(mean_prct_sold = mean(prct_sold)) %>%
  arrange(desc(mean_prct_sold))
```

Question 9

In January of 2015, what city had the fewest houses listed for sale? (show code and text please)

Question 10

In 2012, in which month were the most houses sold in Texas? (show code and text please)

Question 11

Generate a single table that shows the total number of houses sold in **Austin** in **2000 and 2001** (total over the entire period), & the total number of houses sold in **Dallas** in **2000 and 2001** (total over the entire period). This calculation requires a number of steps, so it might help you to first write out on paper the different steps you will need to take. That will help you set out a "blueprint" for tackling the problem.

Hint: recall the `sum()` function can add values.