Problem Set 03: Data Wrangling

Your Name

Last modified on August 24, 2024 08:58:16 Eastern Daylight Time

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.

The following code loads the necessary packages for this problem set:

R Code

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

The Data

The following code chunk loads the data set `txhousing` and displays the data using `glimpse`:

R Code

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

```
Rows: 8,602
Columns: 9
$ city      <chr> "Abilene", "Abilene", "Abilene", "Abilene", "Abilene", "Abil~
$ year      <int> 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, ~
$ month     <int> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 1, 2, 3, 4, 5, 6, 7, ~
$ sales     <dbl> 72, 98, 130, 98, 141, 156, 152, 131, 104, 101, 100, 92, 75, ~
$ volume    <dbl> 5380000, 6505000, 9285000, 9730000, 10590000, 13910000, 1263~
$ median    <dbl> 71400, 58700, 58100, 68600, 67300, 66900, 73500, 75000, 6450~
$ listings  <dbl> 701, 746, 784, 785, 794, 780, 742, 765, 771, 764, 721, 658, ~
$ inventory <dbl> 6.3, 6.6, 6.8, 6.9, 6.8, 6.6, 6.2, 6.4, 6.5, 6.6, 6.2, 5.7, ~
$ date      <dbl> 2000.000, 2000.083, 2000.167, 2000.250, 2000.333, 2000.417, ~
```

```
#
#txhousing <- txhousing[sample(1:802),]
```

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.

Problem 1

Examine `txhousing` 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)? Use `R` code to programatically find the last city in `txhousing`.

Problem 1 Answers

```
# Type your code and comments inside the code chunk
```

- Delete this and put your text answer here.

Problem 2

Examine the variable descriptions by typing `?txhousing` in the **console**. What is the `listings` variable in this data set?

Problem 2 Answers

- Delete this and put your text answer here.

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 columns in the data set for the variables `sales` and `volume`.

```r
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 an assignment operator (`<-`) was not used in the code; consequently the selected values are not saved. In the following code, the results are saved in a data frame **ALSO** called `txhousing`. By putting `-` in front of the `date` variable `R` selects all variables **except** the `date` variable.

R Code

```r
txhousing <- txhousing |>
  select(-date)
head(txhousing)
```

```
# A tibble: 6 x 8
  city     year month sales   volume median listings inventory
  <chr>   <int> <int> <dbl>    <dbl>  <dbl>    <dbl>     <dbl>
1 Abilene  2000     1    72  5380000  71400      701       6.3
2 Abilene  2000     2    98  6505000  58700      746       6.6
3 Abilene  2000     3   130  9285000  58100      784       6.8
4 Abilene  2000     4    98  9730000  68600      785       6.9
5 Abilene  2000     5   141 10590000  67300      794       6.8
6 Abilene  2000     6   156 13910000  66900      780       6.6
```

If you examine `txhousing` in the data viewer, the `date` variable is no longer included.

`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**.

Imagine we wanted to reduce the data set to include data for only 2012 in the city of Austin. The code chunk below filters the `txhousing` to only include rows in which the year is 2012 **and** the city is Austin. The results are saved in a new data frame called `austin_12`.

R Code

```r
austin_12 <- txhousing |>
  filter(year == 2012 & city == "Austin")
  #Or filter(year == 2012, city == "Austin")
head(austin_12)
```

```
# A tibble: 6 x 8
  city     year month sales    volume median listings inventory
  <chr>   <int> <int> <dbl>     <dbl>  <dbl>    <dbl>     <dbl>
1 Austin   2012     1  1182 265821275 177400     7432       4.2
2 Austin   2012     2  1415 353527608 191600     7738       4.3
3 Austin   2012     3  2083 533800484 198600     8186       4.5
4 Austin   2012     4  2128 563288160 207400     8239       4.5
5 Austin   2012     5  2611 705383898 210200     8465       4.5
6 Austin   2012     6  2837 791281075 216000     8641       4.5
```

Note

Note that we use `==` to identify the desired criteria.

What if we wanted to restrict our data set to only years before 2004 and the City of Austin? Below we use the `<` symbol to accomplish this. Note we did not **SAVE** these results in a new data frame...so no new data frame showed up in our Environment pane, but the results print out immediately below the code chunk.

R Code

```r
txhousing |>
  filter(year < 2004, city == "Austin") |>
  head()
```

```
# A tibble: 6 x 8
  city     year month sales    volume median listings inventory
  <chr>   <int> <int> <dbl>     <dbl>  <dbl>    <dbl>     <dbl>
1 Austin   2000     1  1025 173053635 133700     3084         2
2 Austin   2000     2  1277 226038438 134000     2989         2
```

```
3 Austin   2000     3  1603 298557656 136700     3042       2
4 Austin   2000     4  1556 289197960 136900     3192       2.1
5 Austin   2000     5  1980 393073774 144700     3617       2.3
6 Austin   2000     6  1885 368290072 148800     3799       2.4
```

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.

R Code

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

```
# A tibble: 6 x 8
  city     year month sales   volume median listings inventory
  <chr>   <int> <int> <dbl>    <dbl>  <dbl>    <dbl>     <dbl>
1 Abilene  2000     1    72  5380000  71400      701       6.3
2 Abilene  2000     2    98  6505000  58700      746       6.6
3 Abilene  2000     3   130  9285000  58100      784       6.8
4 Abilene  2000     4    98  9730000  68600      785       6.9
5 Abilene  2000     5   141 10590000  67300      794       6.8
6 Abilene  2000     6   156 13910000  66900      780       6.6
```

```r
tail(aust_ab)
```

```
# A tibble: 6 x 8
  city    year month sales     volume median listings inventory
  <chr>  <int> <int> <dbl>      <dbl>  <dbl>    <dbl>     <dbl>
1 Austin  2015     2  1978  595625521 245300     5733       2.2
2 Austin  2015     3  2677  885779822 253900     5906       2.3
3 Austin  2015     4  2801  931744729 270300     6560       2.5
4 Austin  2015     5  2999 1026501450 271200     7009       2.7
5 Austin  2015     6  3301 1086689918 270200     7419       2.8
6 Austin  2015     7  3466 1150381553 264600     7913       3
```

`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.

R Code

```r
txhousing <- txhousing |>
  mutate(vol_100k = volume/100000)
head(txhousing)
```

```
# A tibble: 6 x 9
  city     year month sales   volume median listings inventory vol_100k
  <chr>   <int> <int> <dbl>    <dbl>  <dbl>    <dbl>     <dbl>    <dbl>
1 Abilene  2000     1    72  5380000  71400      701       6.3     53.8
2 Abilene  2000     2    98  6505000  58700      746       6.6     65.0
3 Abilene  2000     3   130  9285000  58100      784       6.8     92.8
4 Abilene  2000     4    98  9730000  68600      785       6.9     97.3
5 Abilene  2000     5   141 10590000  67300      794       6.8    106.
6 Abilene  2000     6   156 13910000  66900      780       6.6    139.
```

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.

```r
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.

R Code

```
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()) -> ans1
kable(ans1)
```

x_bar_sales	sd_sales	min_vol	max_vol	mdn_list	iqr_list	sample_size
2126.75	500.8361	265821275	791281075	7925	948.75	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?

R Code

```
austin_12 |>
  arrange(desc(volume)) -> ans2
head(ans2, n = 3) |>
  kable()
```

city	year	month	sales	volume	median	listings	inventory
Austin	2012	6	2837	791281075	216000	8641	4.5

Austin	2012	7	2604	718755768	211000	8519	4.3
Austin	2012	8	2647	708540314	205100	8112	4.0

The above code tells `R` to `arrange` the rows in the data set based on the `volume` column and to do so in `desc`ending order. Consequently, the row with the $791,281,075 in sales is shown at the top. We can see that this `volume` occurred in the 6th month (June).

`group_by()`

Sometimes we also want to calculate summary statistics across different levels of another variable. For instance, here we find the average number of monthly sales that occurred in Abilene and Austin across all years in the data set. Note that we **use the `aust_ab` data frame** we created earlier, to restrict our analysis to those two cities.

R Code

```r
aust_ab |>
  group_by(city) |>
  summarize(x_bar_sales = mean(sales)) -> results
results |>
  kable()
```

city	x_bar_sales
Abilene	150.4866
Austin	1996.6898

From the results we can see that there were an average of 150.5 sales per month in Abilene, and 1996.7 sales per month in Austin.

We can give `R` multiple variables to group by. For instance, the following code returns the mean sales for each month in each city averaged across all the years.

R Code

```
aust_ab |> group_by(city, month) |>
  summarize(x_bar_sales = mean(sales)) -> results_ab
results_ab |>
  head() |>
  kable()
```

city	month	x_bar_sales
Abilene	1	96.3125
Abilene	2	121.0000
Abilene	3	151.3750
Abilene	4	159.8750
Abilene	5	177.8750
Abilene	6	190.3125

The mean number of sales for Abilene in January (across all years) was 96.3 homes.

Independent Practice

Basic Syntax

This first set of questions will help you practice basic syntax.

Problem 3

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

Problem 3 Answers

```
# Type your code and comments inside the code chunk
```

Problem 4

Make a data set called `dallas_sub` that includes data only from the city of Dallas in 2012 and 2013. Show the first six rows of `dallas_sub`.

Problem 4 Answers

```
# Type your code and comments inside the code chunk
```

Problem 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 into a data frame called `dallas_sub`. Display the last six rows of `dallas_sub`.

Problem 5 Answers

```
# Type your code and comments inside the code chunk
```

Problem 6

Calculate the **average** percentage of listings that were sold in Dallas **in each month across the years** based on your `dallas_sub` data set. Save the results of the calculation in a data frame called `dallas_summary` with the results stored in `mean_prct_sold`. Display the results of `dallas_summary`.

Problem 6 Answers

```
# Type your code and comments inside the code chunk
```

Problem 7

Arrange the `dallas_summary` in `desc`ending order based on the average percentage of listings that were sold in Dallas, 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.

Problem 7 Answers

```
# Type your code and comments inside the code chunk
```

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.

Problem 8

Run the following code chunk. Study the code, and the output. Explain in your own words what this code chunk calculated. Specifically, compare this code to the code in Problems 4 through 7.

Problem 8 Answers

```
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))
```

```
# A tibble: 12 x 2
   month mean_prct_sold
   <int>          <dbl>
 1     8           38.5
 2     5           38.2
 3     6           37.2
 4     7           37.1
 5    12           35.5
 6     4           34.5
 7     3           32.2
 8    10           32.1
 9     9           31.8
10    11           30.6
11     2           23.5
12     1           20.5
```

- Delete this and put your text answer here.

Problem 9

In January of 2015, what city had the fewest houses listed for sale? Report the city and the number of houses said city had listed for sale using inline `R` code.

Problem 9 Answers

```
# Type your code and comments inside the code chunk
```

- Delete this and put your text answer here.

Problem 10

In 2012, in which month were the most houses sold in Texas? Report the month were the most houses sold in Texas and the number of houses sold for that month using inline `R` code.

Problem 10 Answers

```
# Type your code and comments inside the code chunk
```

- Delete this and put your text answer here.

Problem 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.

Problem 11 Answers

```
# Type your code and comments inside the code chunk
```

Turning in Your Work

You will need to make sure you commit and push all of your changes to the github education repository where you obtained the lab.

Tip

- Make sure you **render a final copy with all your changes** and work.
- Look at your final html file to make sure it contains the work you expect and is formatted properly.

Logging out of the Server

There are many statistics classes and students using the Server. To keep the server running as fast as possible, it is best to sign out when you are done. To do so, follow all the same steps for closing Quarto document:

Tip

- Save all your work.
- Click on the orange button in the far right corner of the screen to quit `R`
- Choose **don't save** for the **Workspace image**
- When the browser refreshes, you can click on the sign out next to your name in the top right.
- You are signed out.

```r
sessionInfo()
```

```
R version 4.4.1 (2024-06-14)
Platform: x86_64-redhat-linux-gnu
Running under: Red Hat Enterprise Linux 9.4 (Plow)

Matrix products: default
BLAS/LAPACK: FlexiBLAS OPENBLAS-OPENMP;  LAPACK version 3.9.0

locale:
 [1] LC_CTYPE=en_US.UTF-8       LC_NUMERIC=C
 [3] LC_TIME=en_US.UTF-8        LC_COLLATE=en_US.UTF-8
 [5] LC_MONETARY=en_US.UTF-8    LC_MESSAGES=en_US.UTF-8
 [7] LC_PAPER=en_US.UTF-8       LC_NAME=C
 [9] LC_ADDRESS=C               LC_TELEPHONE=C
[11] LC_MEASUREMENT=en_US.UTF-8 LC_IDENTIFICATION=C

time zone: America/New_York
tzcode source: system (glibc)

attached base packages:
[1] stats     graphics  grDevices utils     datasets  methods   base

other attached packages:
 [1] scales_1.3.0    lubridate_1.9.3 forcats_1.0.0   stringr_1.5.1
 [5] dplyr_1.1.4     purrr_1.0.2     readr_2.1.5     tidyr_1.3.1
 [9] tibble_3.2.1    ggplot2_3.5.1   tidyverse_2.0.0 knitr_1.48

loaded via a namespace (and not attached):
 [1] gtable_0.3.5      jsonlite_1.8.8    compiler_4.4.1    tidyselect_1.2.1
 [5] yaml_2.3.10       fastmap_1.2.0     R6_2.5.1          generics_0.1.3
 [9] munsell_0.5.1     pillar_1.9.0      tzdb_0.4.0        rlang_1.1.4
[13] utf8_1.2.4        stringi_1.8.4     xfun_0.47         timechange_0.3.0
[17] cli_3.6.3         withr_3.0.1       magrittr_2.0.3    digest_0.6.36
[21] grid_4.4.1        rstudioapi_0.16.0 hms_1.1.3         lifecycle_1.0.4
[25] vctrs_0.6.5       evaluate_0.24.0   glue_1.7.0        fansi_1.0.6
[29] colorspace_2.1-1  rmarkdown_2.28    tools_4.4.1       pkgconfig_2.0.3
[33] htmltools_0.5.8.1
```