Data Manipulation with Pandas

March 29, 2020

0.0.1 Data Manipulation with Pandas

I am creating this notebook to showcase some of my Pandas skills that I have been acquiring. This represents only a chunk of the skills that I have been learning. The notebook contains analysis of three different datasets.

[1]:
```
import pandas as pd
import pickle
```

Starting with data of homeless people in the United States.

[3]:
```
file_path = '/Users/MuhammadBilal/Desktop/Data Camp/Pandas/Data Manipulation␣
↪with pandas/Data/homeless_data.pkl'
```

Opening the pkl file.

[5]:
```
import pickle
```

[6]:
```
import pickle
with open('homeless_data.pkl', 'rb') as f:
    data = pickle.load(f)
```

[7]:
```
homelessness = data
```

[9]:
```
# Printing the head of the homelessness data
print(homelessness.head())
```

```
                region       state  individuals  family_members  state_pop
0  East South Central     Alabama       2570.0           864.0    4887681
1             Pacific      Alaska       1434.0           582.0     735139
2            Mountain     Arizona       7259.0          2606.0    7158024
3  West South Central    Arkansas       2280.0           432.0    3009733
4             Pacific  California     109008.0         20964.0   39461588
```

[10]:
```
# Printing information about homelessness
print(homelessness.info())
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 51 entries, 0 to 50
Data columns (total 5 columns):
```

```
region            51 non-null object
state             51 non-null object
individuals       51 non-null float64
family_members    51 non-null float64
state_pop         51 non-null int64
dtypes: float64(2), int64(1), object(2)
memory usage: 2.4+ KB
None
```

[11]:
```
# Printing the shape of homelessness
print(homelessness.shape)
```

```
(51, 5)
```

[12]:
```
# Printing a description of homelessness
print(homelessness.describe())
```

```
         individuals  family_members     state_pop
count      51.000000       51.000000  5.100000e+01
mean     7225.784314     3504.882353  6.405637e+06
std     15991.025083     7805.411811  7.327258e+06
min       434.000000       75.000000  5.776010e+05
25%      1446.500000      592.000000  1.777414e+06
50%      3082.000000     1482.000000  4.461153e+06
75%      6781.500000     3196.000000  7.340946e+06
max    109008.000000    52070.000000  3.946159e+07
```

We can see that the average number of homeless individuals in each state is about 7226. Let's explore the DataFrame further.

[13]:
```
# Printing the values of homelessness
print(homelessness.values)
```

```
[['East South Central' 'Alabama' 2570.0 864.0 4887681]
 ['Pacific' 'Alaska' 1434.0 582.0 735139]
 ['Mountain' 'Arizona' 7259.0 2606.0 7158024]
 ['West South Central' 'Arkansas' 2280.0 432.0 3009733]
 ['Pacific' 'California' 109008.0 20964.0 39461588]
 ['Mountain' 'Colorado' 7607.0 3250.0 5691287]
 ['New England' 'Connecticut' 2280.0 1696.0 3571520]
 ['South Atlantic' 'Delaware' 708.0 374.0 965479]
 ['South Atlantic' 'District of Columbia' 3770.0 3134.0 701547]
 ['South Atlantic' 'Florida' 21443.0 9587.0 21244317]
 ['South Atlantic' 'Georgia' 6943.0 2556.0 10511131]
 ['Pacific' 'Hawaii' 4131.0 2399.0 1420593]
 ['Mountain' 'Idaho' 1297.0 715.0 1750536]
 ['East North Central' 'Illinois' 6752.0 3891.0 12723071]
 ['East North Central' 'Indiana' 3776.0 1482.0 6695497]
 ['West North Central' 'Iowa' 1711.0 1038.0 3148618]
```

```
 ['West North Central' 'Kansas' 1443.0 773.0 2911359]
 ['East South Central' 'Kentucky' 2735.0 953.0 4461153]
 ['West South Central' 'Louisiana' 2540.0 519.0 4659690]
 ['New England' 'Maine' 1450.0 1066.0 1339057]
 ['South Atlantic' 'Maryland' 4914.0 2230.0 6035802]
 ['New England' 'Massachusetts' 6811.0 13257.0 6882635]
 ['East North Central' 'Michigan' 5209.0 3142.0 9984072]
 ['West North Central' 'Minnesota' 3993.0 3250.0 5606249]
 ['East South Central' 'Mississippi' 1024.0 328.0 2981020]
 ['West North Central' 'Missouri' 3776.0 2107.0 6121623]
 ['Mountain' 'Montana' 983.0 422.0 1060665]
 ['West North Central' 'Nebraska' 1745.0 676.0 1925614]
 ['Mountain' 'Nevada' 7058.0 486.0 3027341]
 ['New England' 'New Hampshire' 835.0 615.0 1353465]
 ['Mid-Atlantic' 'New Jersey' 6048.0 3350.0 8886025]
 ['Mountain' 'New Mexico' 1949.0 602.0 2092741]
 ['Mid-Atlantic' 'New York' 39827.0 52070.0 19530351]
 ['South Atlantic' 'North Carolina' 6451.0 2817.0 10381615]
 ['West North Central' 'North Dakota' 467.0 75.0 758080]
 ['East North Central' 'Ohio' 6929.0 3320.0 11676341]
 ['West South Central' 'Oklahoma' 2823.0 1048.0 3940235]
 ['Pacific' 'Oregon' 11139.0 3337.0 4181886]
 ['Mid-Atlantic' 'Pennsylvania' 8163.0 5349.0 12800922]
 ['New England' 'Rhode Island' 747.0 354.0 1058287]
 ['South Atlantic' 'South Carolina' 3082.0 851.0 5084156]
 ['West North Central' 'South Dakota' 836.0 323.0 878698]
 ['East South Central' 'Tennessee' 6139.0 1744.0 6771631]
 ['West South Central' 'Texas' 19199.0 6111.0 28628666]
 ['Mountain' 'Utah' 1904.0 972.0 3153550]
 ['New England' 'Vermont' 780.0 511.0 624358]
 ['South Atlantic' 'Virginia' 3928.0 2047.0 8501286]
 ['Pacific' 'Washington' 16424.0 5880.0 7523869]
 ['South Atlantic' 'West Virginia' 1021.0 222.0 1804291]
 ['East North Central' 'Wisconsin' 2740.0 2167.0 5807406]
 ['Mountain' 'Wyoming' 434.0 205.0 577601]]
```

[14]:
```
# Printing the column index of homelessness
print(homelessness.columns)
```

```
Index(['region', 'state', 'individuals', 'family_members', 'state_pop'],
dtype='object')
```

[15]:
```
# Printing the row index of homelessness
print(homelessness.index)
```

```
Int64Index([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15, 16,
            17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33,
            34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49,
```

```
        50],
       dtype='int64')
```

Sorting Rows

[17]:
```
# Sorting homelessness by individual
homelessness_ind = homelessness.sort_values('individuals',ascending = True)

# Printing the top few rows
print(homelessness_ind.head())
```

```
                region         state  individuals  family_members  state_pop
50            Mountain       Wyoming        434.0           205.0     577601
34  West North Central  North Dakota        467.0            75.0     758080
7       South Atlantic      Delaware        708.0           374.0     965479
39         New England  Rhode Island        747.0           354.0    1058287
45         New England       Vermont        780.0           511.0     624358
```

[18]:
```
# Sorting homelessness by descending family members
homelessness_fam = homelessness.sort_values('family_members', ascending = False)

# Printing the top few rows
print(homelessness_fam.head())
```

```
                region          state  individuals  family_members  state_pop
32        Mid-Atlantic       New York      39827.0         52070.0   19530351
4              Pacific     California     109008.0         20964.0   39461588
21         New England  Massachusetts       6811.0         13257.0    6882635
9       South Atlantic        Florida      21443.0          9587.0   21244317
43  West South Central          Texas      19199.0          6111.0   28628666
```

[19]:
```
# Sorting homelessness by region, then descending family members
homelessness_reg_fam = homelessness.sort_values(['region', 'family_members'],␣
↪ascending=[True, False] )

# Printing the top few rows
print(homelessness_reg_fam.head())
```

```
                region      state  individuals  family_members  state_pop
13  East North Central   Illinois       6752.0          3891.0   12723071
35  East North Central       Ohio       6929.0          3320.0   11676341
22  East North Central   Michigan       5209.0          3142.0    9984072
49  East North Central  Wisconsin       2740.0          2167.0    5807406
14  East North Central    Indiana       3776.0          1482.0    6695497
```

Subsetting columns

[21]:
```
# Selecting the individuals column
individuals = homelessness['individuals']
```

```
# Printing the head of the result
print(individuals.head())
```

```
0      2570.0
1      1434.0
2      7259.0
3      2280.0
4    109008.0
Name: individuals, dtype: float64
```

[22]:
```
# Selecting the state and family_members columns
state_fam = homelessness[['state', 'family_members']]

# Printing the head of the result
print(state_fam.head())
```

```
        state  family_members
0     Alabama           864.0
1      Alaska           582.0
2     Arizona          2606.0
3    Arkansas           432.0
4  California         20964.0
```

[23]:
```
# Selecting only the individuals and state columns, in that order
ind_state = homelessness[['individuals', 'state']]
```

[24]:
```
# Printing the head of the result
print(ind_state.head())
```

```
   individuals       state
0       2570.0     Alabama
1       1434.0      Alaska
2       7259.0     Arizona
3       2280.0    Arkansas
4     109008.0  California
```

Subsetting Rows

[26]:
```
# Filtering for rows where individuals is greater than 10000
ind_gt_10k = homelessness[(homelessness['individuals'] > 10000)]

# Seeing the result
print(ind_gt_10k)
```

```
                region       state  individuals  family_members  state_pop
4              Pacific  California     109008.0         20964.0   39461588
9       South Atlantic     Florida      21443.0          9587.0   21244317
```

```
32        Mid-Atlantic    New York      39827.0          52070.0   19530351
37             Pacific      Oregon      11139.0           3337.0    4181886
43  West South Central       Texas      19199.0           6111.0   28628666
47             Pacific  Washington      16424.0           5880.0    7523869
```

[27]:
```
# Filtering for rows where region is Mountain
mountain_reg = homelessness[homelessness["region"] == "Mountain"]

# Seeing the result
print(mountain_reg)
```

```
      region       state  individuals  family_members  state_pop
2   Mountain     Arizona       7259.0          2606.0    7158024
5   Mountain    Colorado       7607.0          3250.0    5691287
12  Mountain       Idaho       1297.0           715.0    1750536
26  Mountain     Montana        983.0           422.0    1060665
28  Mountain      Nevada       7058.0           486.0    3027341
31  Mountain  New Mexico       1949.0           602.0    2092741
44  Mountain        Utah       1904.0           972.0    3153550
50  Mountain     Wyoming        434.0           205.0     577601
```

[28]:
```
# Filtering for rows where family_members is less than 1000 and region is␣
↪Pacific
fam_lt_1k_pac = homelessness[(homelessness["family_members"] < 1000) &␣
↪(homelessness["region"] == "Pacific")]

# Seeing the result
print(fam_lt_1k_pac)
```

```
    region   state  individuals  family_members  state_pop
1  Pacific  Alaska       1434.0           582.0     735139
```

Subsetting rows by categorical variables

[30]:
```
# Subsetting for rows in South Atlantic or Mid-Atlantic regions
south_mid_atlantic = homelessness[(homelessness['region'] == 'South Atlantic')␣
↪| (homelessness['region'] == 'Mid-Atlantic')]

# Seeing the result
print(south_mid_atlantic)
```

```
            region                 state  individuals  family_members  \
7   South Atlantic              Delaware        708.0           374.0
8   South Atlantic  District of Columbia       3770.0          3134.0
9   South Atlantic               Florida      21443.0          9587.0
10  South Atlantic               Georgia       6943.0          2556.0
20  South Atlantic              Maryland       4914.0          2230.0
30    Mid-Atlantic            New Jersey       6048.0          3350.0
```

```
32    Mid-Atlantic              New York      39827.0         52070.0
33  South Atlantic        North Carolina       6451.0          2817.0
38    Mid-Atlantic          Pennsylvania       8163.0          5349.0
40  South Atlantic        South Carolina       3082.0           851.0
46  South Atlantic              Virginia       3928.0          2047.0
48  South Atlantic         West Virginia       1021.0           222.0

    state_pop
7      965479
8      701547
9    21244317
10   10511131
20    6035802
30    8886025
32   19530351
33   10381615
38   12800922
40    5084156
46    8501286
48    1804291
```

[31]:
```
# The Mojave Desert states
canu = ["California", "Arizona", "Nevada", "Utah"]

# Filtering for rows in the Mojave Desert states
mojave_homelessness = homelessness[homelessness["state"].isin(canu)]

# Seeing the result
print(mojave_homelessness)
```

```
      region       state  individuals  family_members  state_pop
2   Mountain     Arizona       7259.0          2606.0    7158024
4    Pacific  California     109008.0         20964.0   39461588
28  Mountain      Nevada       7058.0           486.0    3027341
44  Mountain        Utah       1904.0           972.0    3153550
```

Subsetting! Using square brackets plus logical conditions is often the most powerful way of identifying interesting rows of data.

Adding new columns

[33]:
```
# Adding total col as sum of individuals and family_members
homelessness["total"] = homelessness["individuals"] +␣
 ↪homelessness["family_members"]

# Adding p_individuals col as proportion of individuals
homelessness["p_individuals"] = homelessness["individuals"] /␣
 ↪homelessness["total"]
```

```
# Seeing the result
print(homelessness)
```

```
                region                 state  individuals  family_members  \
0   East South Central               Alabama       2570.0           864.0
1              Pacific                Alaska       1434.0           582.0
2             Mountain               Arizona       7259.0          2606.0
3   West South Central              Arkansas       2280.0           432.0
4              Pacific            California     109008.0         20964.0
5             Mountain              Colorado       7607.0          3250.0
6          New England           Connecticut       2280.0          1696.0
7       South Atlantic              Delaware        708.0           374.0
8       South Atlantic  District of Columbia       3770.0          3134.0
9       South Atlantic               Florida      21443.0          9587.0
10      South Atlantic               Georgia       6943.0          2556.0
11             Pacific                Hawaii       4131.0          2399.0
12            Mountain                 Idaho       1297.0           715.0
13  East North Central              Illinois       6752.0          3891.0
14  East North Central               Indiana       3776.0          1482.0
15  West North Central                  Iowa       1711.0          1038.0
16  West North Central                Kansas       1443.0           773.0
17  East South Central              Kentucky       2735.0           953.0
18  West South Central             Louisiana       2540.0           519.0
19         New England                 Maine       1450.0          1066.0
20      South Atlantic              Maryland       4914.0          2230.0
21         New England         Massachusetts       6811.0         13257.0
22  East North Central              Michigan       5209.0          3142.0
23  West North Central             Minnesota       3993.0          3250.0
24  East South Central           Mississippi       1024.0           328.0
25  West North Central              Missouri       3776.0          2107.0
26            Mountain               Montana        983.0           422.0
27  West North Central              Nebraska       1745.0           676.0
28            Mountain                Nevada       7058.0           486.0
29         New England         New Hampshire        835.0           615.0
30        Mid-Atlantic            New Jersey       6048.0          3350.0
31            Mountain            New Mexico       1949.0           602.0
32        Mid-Atlantic              New York      39827.0         52070.0
33      South Atlantic        North Carolina       6451.0          2817.0
34  West North Central          North Dakota        467.0            75.0
35  East North Central                  Ohio       6929.0          3320.0
36  West South Central              Oklahoma       2823.0          1048.0
37             Pacific                Oregon      11139.0          3337.0
38        Mid-Atlantic          Pennsylvania       8163.0          5349.0
39         New England          Rhode Island        747.0           354.0
40      South Atlantic        South Carolina       3082.0           851.0
41  West North Central          South Dakota        836.0           323.0
```

42	East South Central	Tennessee	6139.0	1744.0
43	West South Central	Texas	19199.0	6111.0
44	Mountain	Utah	1904.0	972.0
45	New England	Vermont	780.0	511.0
46	South Atlantic	Virginia	3928.0	2047.0
47	Pacific	Washington	16424.0	5880.0
48	South Atlantic	West Virginia	1021.0	222.0
49	East North Central	Wisconsin	2740.0	2167.0
50	Mountain	Wyoming	434.0	205.0

	state_pop	total	p_individuals
0	4887681	3434.0	0.748398
1	735139	2016.0	0.711310
2	7158024	9865.0	0.735834
3	3009733	2712.0	0.840708
4	39461588	129972.0	0.838704
5	5691287	10857.0	0.700654
6	3571520	3976.0	0.573441
7	965479	1082.0	0.654344
8	701547	6904.0	0.546060
9	21244317	31030.0	0.691041
10	10511131	9499.0	0.730919
11	1420593	6530.0	0.632619
12	1750536	2012.0	0.644632
13	12723071	10643.0	0.634408
14	6695497	5258.0	0.718144
15	3148618	2749.0	0.622408
16	2911359	2216.0	0.651173
17	4461153	3688.0	0.741594
18	4659690	3059.0	0.830337
19	1339057	2516.0	0.576312
20	6035802	7144.0	0.687850
21	6882635	20068.0	0.339396
22	9984072	8351.0	0.623758
23	5606249	7243.0	0.551291
24	2981020	1352.0	0.757396
25	6121623	5883.0	0.641849
26	1060665	1405.0	0.699644
27	1925614	2421.0	0.720777
28	3027341	7544.0	0.935578
29	1353465	1450.0	0.575862
30	8886025	9398.0	0.643541
31	2092741	2551.0	0.764014
32	19530351	91897.0	0.433387
33	10381615	9268.0	0.696051
34	758080	542.0	0.861624
35	11676341	10249.0	0.676066
36	3940235	3871.0	0.729269

```
37    4181886   14476.0       0.769481
38   12800922   13512.0       0.604130
39    1058287    1101.0       0.678474
40    5084156    3933.0       0.783626
41     878698    1159.0       0.721311
42    6771631    7883.0       0.778764
43   28628666   25310.0       0.758554
44    3153550    2876.0       0.662031
45     624358    1291.0       0.604183
46    8501286    5975.0       0.657406
47    7523869   22304.0       0.736370
48    1804291    1243.0       0.821400
49    5807406    4907.0       0.558386
50     577601     639.0       0.679186
```

[34]:
```
# Creating indiv_per_10k col as homeless individuals per 10k state pop
homelessness["indiv_per_10k"] = 10000 * homelessness["individuals"] /␣
 ↪homelessness["state_pop"]

# Subsetting rows for indiv_per_10k greater than 20
high_homelessness = homelessness[homelessness["indiv_per_10k"] > 20]

# Sorting high_homelessness by descending indiv_per_10k
high_homelessness_srt = high_homelessness.sort_values("indiv_per_10k",␣
 ↪ascending=False)

# From high_homelessness_srt, selecting the state and indiv_per_10k cols
result = high_homelessness_srt[["state", "indiv_per_10k"]]

# Seeing the result
print(result)
```

```
                   state  indiv_per_10k
8   District of Columbia      53.738381
11                Hawaii      29.079406
4             California      27.623825
37                Oregon      26.636307
28                Nevada      23.314189
47            Washington      21.829195
32              New York      20.392363
```

Washington D.C. has the highest number of homeless individuals - almost 54 per ten thousand people. This is almost double the number of the next-highest state, Hawaii.

0.0.2 Using Walmart sales data to make inferences using Pandas.

[38]:
```
file_path1 = '/Users/MuhammadBilal/Desktop/Data Camp/Data Manipulation with␣
↪pandas/Data/walmart_sales.pkl'
```

[39]:
```
with open('walmart_sales.pkl', 'rb') as f:
    data = pickle.load(f)
sales = data
```

[41]:
```
print(sales.head())
```

```
   store type  department       date  weekly_sales  is_holiday  temperature_c  \
0      1    A           1 2010-02-05      24924.50       False       5.727778
1      1    A           2 2010-02-05      50605.27       False       5.727778
2      1    A           3 2010-02-05      13740.12       False       5.727778
3      1    A           4 2010-02-05      39954.04       False       5.727778
4      1    A           5 2010-02-05      32229.38       False       5.727778

   fuel_price_usd_per_l  unemployment
0              0.679451         8.106
1              0.679451         8.106
2              0.679451         8.106
3              0.679451         8.106
4              0.679451         8.106
```

[42]:
```
# Printing the info about the sales DataFrame
print(sales.info())
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 413119 entries, 0 to 413118
Data columns (total 9 columns):
store                  413119 non-null int64
type                   413119 non-null object
department             413119 non-null int32
date                   413119 non-null datetime64[ns]
weekly_sales           413119 non-null float64
is_holiday             413119 non-null bool
temperature_c          413119 non-null float64
fuel_price_usd_per_l   413119 non-null float64
unemployment           413119 non-null float64
dtypes: bool(1), datetime64[ns](1), float64(4), int32(1), int64(1), object(1)
memory usage: 27.2+ MB
None
```

[43]:
```
# Printing the mean of weekly_sales
print(sales['weekly_sales'].mean())

# Printing the median of weekly_sales
```

```
print(sales['weekly_sales'].median())
```

```
16094.726811185497
7682.47
```

The mean weekly sales is almost double the median weekly sales! This can tell us that there are a few very high sales weeks that are making the mean so much higher than the median.

[44]:
```
# Printing the maximum of the date column
print(sales['date'].max())
```

```
2012-10-26 00:00:00
```

[45]:
```
# Printing the minimum of the date column
print(sales['date'].min())
```

```
2010-02-05 00:00:00
```

The data covers dates from February of 2010 to October of 2012.

Efficient summaries

Using the custom iqr function defined along with .agg() to print the IQR of the temperature_c column of sales.

[47]:
```
# A custom IQR function
def iqr(column):
    return column.quantile(0.75) - column.quantile(0.25)

# Printing IQR of the temperature_c column
print(sales['temperature_c'].agg(iqr))
```

```
15.299999999999994
```

Updating the column selection to use the custom iqr function with .agg() to print the IQR of temperature_c, fuel_price_usd_per_l, and unemployment, in that order. There can be many uses of IQR as they tell us where the data lie, if there are any outliers, they help to draw box plots, and so forth.

[48]:
```
# A custom IQR function
def iqr(column):
    return column.quantile(0.75) - column.quantile(0.25)
```

[49]:
```
# Update to print IQR of temperature_c, fuel_price_usd_per_l, & unemployment
print(sales[["temperature_c", "fuel_price_usd_per_l", "unemployment"]].agg(iqr))
```

```
temperature_c           15.300000
fuel_price_usd_per_l     0.211866
unemployment             1.672000
dtype: float64
```

Updating the aggregation functions called by .agg(): including iqr and np.median in that order.

[50]:
```
# Import NumPy and create custom IQR function
import numpy as np
def iqr(column):
    return column.quantile(0.75) - column.quantile(0.25)
```

[51]:
```
# Update to print IQR and median of temperature_c, fuel_price_usd_per_l, &␣
↪unemployment
print(sales[["temperature_c", "fuel_price_usd_per_l", "unemployment"]].
↪agg([iqr, np.median]))
```

```
        temperature_c  fuel_price_usd_per_l  unemployment
iqr             15.30              0.211866         1.672
median          16.75              0.911922         7.852
```

The .agg() method makes it easy to compute multiple statistics on multiple columns, all in just one line of code.

Cumulative statistics

Cumulative statistics can also be helpful in tracking summary statistics over time. The cumulative sum and cumulative max of a department's weekly sales are calculated, which will allow us to identify what the total sales are as well as what the highest weekly sales are.

[ ]:
```
# A DataFrame called sales_1_1 is created next, which contains the sales data␣
↪for department 1 of store 1.
```

[52]:
```
sales_1_1 = sales[(sales["store"] == 1) & (sales["department"] == 1)]
print(sales_1_1)
```

```
       store type  department       date  weekly_sales  is_holiday  \
0          1    A           1 2010-02-05      24924.50       False
73         1    A           1 2010-02-12      46039.49        True
145        1    A           1 2010-02-19      41595.55       False
218        1    A           1 2010-02-26      19403.54       False
290        1    A           1 2010-03-05      21827.90       False
…      …   …          …         …            …          …
9883       1    A           1 2012-09-28      18947.81       False
9956       1    A           1 2012-10-05      21904.47       False
10028      1    A           1 2012-10-12      22764.01       False
10101      1    A           1 2012-10-19      24185.27       False
10172      1    A           1 2012-10-26      27390.81       False

       temperature_c  fuel_price_usd_per_l  unemployment
0           5.727778              0.679451         8.106
73          3.616667              0.673111         8.106
145         4.405556              0.664129         8.106
218         8.127778              0.676545         8.106
```

```
290          8.055556              0.693452          8.106
…               …                     …              …
9883        24.488889              0.968455          6.908
9956        20.305556              0.955511          6.573
10028       17.216667              0.951284          6.573
10101       19.983333              0.949435          6.573
10172       20.644444              0.926188          6.573

[143 rows x 9 columns]
```

[53]:
```
# Sorting sales_1_1 by date
sales_1_1 = sales_1_1.sort_values("date")

# Getting the cumulative sum of weekly_sales, adding as cum_weekly_sales col
sales_1_1["cum_weekly_sales"] = sales_1_1["weekly_sales"].cumsum()

# Getting the cumulative max of weekly_sales, adding as cum_max_sales col
sales_1_1["cum_max_sales"] = sales_1_1["weekly_sales"].cummax()

# Seeing the calculated columns
print(sales_1_1[["date", "weekly_sales", "cum_weekly_sales", "cum_max_sales"]])
```

```
            date  weekly_sales  cum_weekly_sales  cum_max_sales
0     2010-02-05      24924.50          24924.50       24924.50
73    2010-02-12      46039.49          70963.99       46039.49
145   2010-02-19      41595.55         112559.54       46039.49
218   2010-02-26      19403.54         131963.08       46039.49
290   2010-03-05      21827.90         153790.98       46039.49
…            …             …                 …              …
9883  2012-09-28      18947.81        3123160.62       57592.12
9956  2012-10-05      21904.47        3145065.09       57592.12
10028 2012-10-12      22764.01        3167829.10       57592.12
10101 2012-10-19      24185.27        3192014.37       57592.12
10172 2012-10-26      27390.81        3219405.18       57592.12

[143 rows x 4 columns]
```

Dropping duplicates

[54]:
```
# Dropping duplicate store/type combinations
store_types = sales.drop_duplicates(subset=["store", "type"])
print(store_types.head())

# Dropping duplicate store/department combinations
store_depts = sales.drop_duplicates(subset=["store", "department"])
print(store_depts.head())

# Subsetting the rows that are holiday weeks and drop duplicate dates
```

```
holiday_dates = sales[sales["is_holiday"]].drop_duplicates(subset="date")

# Printing date col of holiday_dates
print(holiday_dates["date"])
```

```
       store type  department       date  weekly_sales  is_holiday  \
0          1    A           1 2010-02-05      24924.50       False
10244      2    A           1 2010-02-05      35034.06       False
20482      3    B           1 2010-02-05       6453.58       False
29518      4    A           1 2010-02-05      38724.42       False
39790      5    B           1 2010-02-05       9323.89       False

       temperature_c  fuel_price_usd_per_l  unemployment
0           5.727778              0.679451         8.106
10244       4.550000              0.679451         8.324
20482       7.616667              0.679451         7.368
29518       6.533333              0.686319         8.623
39790       4.277778              0.679451         6.566
   store type  department       date  weekly_sales  is_holiday  temperature_c  \
0      1    A           1 2010-02-05      24924.50       False       5.727778
1      1    A           2 2010-02-05      50605.27       False       5.727778
2      1    A           3 2010-02-05      13740.12       False       5.727778
3      1    A           4 2010-02-05      39954.04       False       5.727778
4      1    A           5 2010-02-05      32229.38       False       5.727778

   fuel_price_usd_per_l  unemployment
0              0.679451         8.106
1              0.679451         8.106
2              0.679451         8.106
3              0.679451         8.106
4              0.679451         8.106
73     2010-02-12
2218   2010-09-10
3014   2010-11-26
3372   2010-12-31
3800   2011-02-11
5940   2011-09-09
6731   2011-11-25
7096   2011-12-30
7527   2012-02-10
9667   2012-09-07
Name: date, dtype: datetime64[ns]
```

The holiday weeks correspond to the Superbowl in February, Labor Day in September, Thanksgiving in November, and Christmas in December. Now that the duplicates are removed. Further, some counting is done.

[55]:
```
# Counting the number of stores of each type
store_counts = store_types["type"].value_counts()
print(store_counts)
```

```
A    22
B    17
C     6
Name: type, dtype: int64
```

[56]:
```
# Getting the proportion of stores of each type
store_props = store_types["type"].value_counts(normalize=True)
print(store_props)
```

```
A    0.488889
B    0.377778
C    0.133333
Name: type, dtype: float64
```

Walmart distinguishes three types of stores: “supercenters”, “discount stores”, and “neighborhood markets”, encoded in this dataset as type “A”, “B”, and “C”.

[57]:
```
# Counting the number of departments of each type and sort
dept_counts_sorted = store_depts["department"].value_counts(sort=True)
print(dept_counts_sorted)
```

```
1     45
9     45
4     45
6     45
8     45
      ..
37    20
50    14
43     5
39     5
65     1
Name: department, Length: 81, dtype: int64
```

[58]:
```
# Getting the proportion of departments of each type and sort
dept_props_sorted = store_depts["department"].value_counts(sort=True,␣
 ↪normalize=True)
print(dept_props_sorted)
```

```
1     0.013778
9     0.013778
4     0.013778
6     0.013778
8     0.013778
```

```
        ...
37    0.006124
50    0.004287
43    0.001531
39    0.001531
65    0.000306
Name: department, Length: 81, dtype: float64
```

[62]:
```
# Calculating total weekly sales
sales_all = sales["weekly_sales"].sum()
print(sales_all)

# Subsetting for type A stores, calc total weekly sales
sales_A = sales[sales["type"] == "A"]["weekly_sales"].sum()
print(sales_A)

# Subsetting for type B stores, calc total weekly sales
sales_B = sales[sales["type"] == "B"]["weekly_sales"].sum()
print(sales_B)

# Subsetting for type C stores, calc total weekly sales
sales_C = sales[sales["type"] == "C"]["weekly_sales"].sum()
print(sales_C)

# Getting proportion for each type
sales_propn_by_type = [sales_A, sales_B, sales_C] / sales_all
print(sales_propn_by_type)
```

```
6649037445.509999
4331014722.749999
1912519195.2199998
405503527.53999996
[0.65137469 0.28763851 0.0609868 ]
```

About 65% of sales occurred in stores of type A', 28% in stores of type B, and 6% in stores of type C. Same calculations can be done using .groupby().

[63]:
```
# Calculations with .groupby()

# Groupping by type; calc total weekly sales
sales_by_type = sales.groupby("type")["weekly_sales"].sum()

# Getting proportion for each type
sales_propn_by_type = sales_by_type / sum(sales_by_type)
print(sales_propn_by_type)
```

```
type
A    0.651375
```

```
B    0.287639
C    0.060987
Name: weekly_sales, dtype: float64
```

[64]:
```
# Groupping sales by "type" and "is_holiday", taking the sum of weekly_sales,␣
 ↪and storing as sales_by_holiday_type.

# From previous step
sales_by_type = sales.groupby("type")["weekly_sales"].sum()

# Groupping by type and is_holiday; calc total weekly sales
sales_by_type_is_holiday = sales.groupby(['type',␣
 ↪'is_holiday'])['weekly_sales'].sum()
print(sales_by_type_is_holiday)
```

```
type  is_holiday
A     False         4.007612e+09
      True          3.234028e+08
B     False         1.765411e+09
      True          1.471081e+08
C     False         3.772478e+08
      True          2.825570e+07
Name: weekly_sales, dtype: float64
```

Same calculations are done with much less code.

Getting the min, max, mean, and median of weekly_sales for each store type using .groupby() and .agg(). Storing this as sales_stats.

Getting the min, max, mean, and median of unemployment and fuel_price_usd_per_l for each store type. Storing this as unemp_fuel_stats.

[65]:
```
# Importing NumPy with the alias np
import numpy as np

# For each store type, aggregate weekly_sales: getting min, max, mean, and␣
 ↪median
sales_stats = sales.groupby("type")["weekly_sales"].agg([np.min, np.max, np.
 ↪mean, np.median])

# Printing sales_stats
print(sales_stats)
```

```
         amin       amax          mean    median
type
A    -4988.94  474330.10  20099.568043  10105.17
B    -3924.00  693099.36  12335.331875   6269.02
C     -379.00  112152.35   9519.532538   1149.67
```

[66]:
```
# For each store type, aggregate unemployment and fuel_price_usd_per_l: getting␣
↪min, max, mean, and median
unemp_fuel_stats = sales.groupby("type")[["unemployment",␣
↪"fuel_price_usd_per_l"]].agg([np.min, np.max, np.mean, np.median])

# Printing unemp_fuel_stats
print(unemp_fuel_stats)
```

```
     unemployment                            fuel_price_usd_per_l            \
             amin    amax      mean median                 amin      amax
type
A           3.879  14.313  7.791595  7.818             0.653034  1.180321
B           4.125  14.313  7.889666  7.806             0.664129  1.180321
C           5.217  14.313  8.934350  8.300             0.664129  1.180321


          mean    median
type
A     0.883391  0.902676
B     0.892997  0.922225
C     0.888848  0.902676
```

Noticing that the minimum weekly_sales is negative because some stores had more returns than sales.

Pivot Tables

Pivoting on one variable

[67]:
```
# Pivoting for mean weekly_sales for each store type
mean_sales_by_type = sales.pivot_table(values='weekly_sales', index='type')

# Printing mean_sales_by_type
print(mean_sales_by_type)
```

```
      weekly_sales
type
A     20099.568043
B     12335.331875
C      9519.532538
```

[68]:
```
# Getting the mean and median (using NumPy functions) of weekly_sales by type␣
↪using .pivot_table() and storing as mean_med_sales_by_type.

# Pivoting for mean and median weekly_sales for each store type
mean_med_sales_by_type = sales.pivot_table(values='weekly_sales', index='type',␣
↪aggfunc=[np.mean, np.median])

# Printing mean_med_sales_by_type
```

```
print(mean_med_sales_by_type)
```

```
              mean       median
      weekly_sales weekly_sales
type
A     20099.568043     10105.17
B     12335.331875      6269.02
C      9519.532538      1149.67
```

[69]:
```
# Getting the mean of weekly_sales by type and is_holiday using .pivot_table()␣
 ↪and store as mean_sales_by_type_holiday.

# Pivoting for mean weekly_sales by store type and holiday
mean_sales_by_type_holiday = sales.pivot_table(values="weekly_sales",␣
 ↪index="type", columns="is_holiday")

# Printing mean_sales_by_type_holiday
print(mean_sales_by_type_holiday)
```

```
is_holiday         False         True
type
A           20008.746759  21297.517824
B           12248.741339  13478.844240
C            9518.528116   9532.963131
```

[70]:
```
# Filling in missing values and summing values with pivot tables

# Printing the mean weekly_sales by department and type, filling in any missing␣
 ↪values with 0.

# Printing mean weekly_sales by department and type; fill missing values with 0
print(sales.pivot_table(values='weekly_sales', index='type',␣
 ↪columns='department', fill_value=0))
```

```
department            1             2             3             4   \
type
A           22956.887886  51994.674873  13881.033137  32973.814075
B           17990.876158  43051.996919  12965.414311  21259.895804
C            8951.733462  14424.851713    820.276818  13669.370396

department            5            6             7             8   \
type
A           26803.448045  5585.277707  30786.372028  37091.220995
B           21184.602916  5006.859317  23915.734587  27578.908420
C             767.600774    36.554462    564.668497  12293.092203

department            9            10  …            90            91  \
```

```
type                                    …
A           24025.109521  23757.932155  …  70550.502168  53734.139097
B           22768.012421  17845.169969  …  12368.083287   8338.731355
C             114.774217    335.823648  …  43628.231072  30623.882494

department             92            93            94            95  \
type
A           112156.881662  43296.564971  51067.047111  97094.026043
B            31491.707710   1503.127752   1045.368843  40580.306862
C            60795.759371  23826.284965  31636.174895  50641.564872

department             96            97            98          99
type
A           19900.943552  22093.807101  10979.816195  431.443064
B            4752.674874   3543.243304    299.951644   25.716667
C           15766.025431  13419.542809   5479.758054    8.330952

[3 rows x 81 columns]
```

[71]:
```
# Printing the mean weekly_sales by department and type, filling in any missing␣
 ↪values with 0 and summing all rows and columns.

# Printing the mean weekly_sales by department and type; filling missing values␣
 ↪with 0s; summing all rows and cols
print(sales.pivot_table(values="weekly_sales", index="department",␣
 ↪columns="type", fill_value=0, margins=True))
```

```
type                   A             B             C           All
department
1           22956.887886  17990.876158   8951.733462  19213.485088
2           51994.674873  43051.996919  14424.851713  43607.020113
3           13881.033137  12965.414311    820.276818  11793.698516
4           32973.814075  21259.895804  13669.370396  25974.630238
5           26803.448045  21184.602916    767.600774  21365.583515
…                    …             …             …           …
96          19900.943552   4752.674874  15766.025431  15217.211505
97          22093.807101   3543.243304  13419.542809  14437.120839
98          10979.816195    299.951644   5479.758054   6973.013875
99            431.443064     25.716667      8.330952    415.487065
All         20099.568043  12335.331875   9519.532538  16094.726811

[82 rows x 4 columns]
```

Setting & removing indexes

[73]:
```
# Looking at sales
print(sales.head())
```

```
# Indexing sales by type and department
sales_ind = sales.set_index('type', 'department')

# Looking at sales_ind
print(sales_ind)

# Resetting the index, keeping its contents
print(sales_ind.reset_index())

# Resetting the index, dropping its contents
print(sales_ind.reset_index(drop=True))
```

```
   store type  department       date  weekly_sales  is_holiday  temperature_c  \
0      1    A           1 2010-02-05      24924.50       False       5.727778
1      1    A           2 2010-02-05      50605.27       False       5.727778
2      1    A           3 2010-02-05      13740.12       False       5.727778
3      1    A           4 2010-02-05      39954.04       False       5.727778
4      1    A           5 2010-02-05      32229.38       False       5.727778

   fuel_price_usd_per_l  unemployment
0              0.679451         8.106
1              0.679451         8.106
2              0.679451         8.106
3              0.679451         8.106
4              0.679451         8.106
      store  department       date  weekly_sales  is_holiday  temperature_c  \
type
A         1           1 2010-02-05      24924.50       False       5.727778
A         1           2 2010-02-05      50605.27       False       5.727778
A         1           3 2010-02-05      13740.12       False       5.727778
A         1           4 2010-02-05      39954.04       False       5.727778
A         1           5 2010-02-05      32229.38       False       5.727778
…     …         …         …             …          …             …
B        45           4 2012-10-26      24627.94       False      14.916667
B        45           5 2012-10-26      13256.59       False      14.916667
B        45           6 2012-10-26       1086.31       False      14.916667
B        45           7 2012-10-26      20356.73       False      14.916667
B        45           8 2012-10-26      37857.64       False      14.916667

      fuel_price_usd_per_l  unemployment
type
A                 0.679451         8.106
A                 0.679451         8.106
A                 0.679451         8.106
A                 0.679451         8.106
A                 0.679451         8.106
…                    …            …
```

```
B                 1.025516         8.667
B                 1.025516         8.667
B                 1.025516         8.667
B                 1.025516         8.667
B                 1.025516         8.667

[413119 rows x 8 columns]
       type  store  department       date  weekly_sales  is_holiday  \
0         A      1           1 2010-02-05      24924.50       False
1         A      1           2 2010-02-05      50605.27       False
2         A      1           3 2010-02-05      13740.12       False
3         A      1           4 2010-02-05      39954.04       False
4         A      1           5 2010-02-05      32229.38       False
…     …    …         …         …            …          …
413114    B     45           4 2012-10-26      24627.94       False
413115    B     45           5 2012-10-26      13256.59       False
413116    B     45           6 2012-10-26       1086.31       False
413117    B     45           7 2012-10-26      20356.73       False
413118    B     45           8 2012-10-26      37857.64       False

        temperature_c  fuel_price_usd_per_l  unemployment
0            5.727778              0.679451         8.106
1            5.727778              0.679451         8.106
2            5.727778              0.679451         8.106
3            5.727778              0.679451         8.106
4            5.727778              0.679451         8.106
…               …                    …           …
413114      14.916667              1.025516         8.667
413115      14.916667              1.025516         8.667
413116      14.916667              1.025516         8.667
413117      14.916667              1.025516         8.667
413118      14.916667              1.025516         8.667

[413119 rows x 9 columns]
        store  department       date  weekly_sales  is_holiday  temperature_c  \
0           1           1 2010-02-05      24924.50       False       5.727778
1           1           2 2010-02-05      50605.27       False       5.727778
2           1           3 2010-02-05      13740.12       False       5.727778
3           1           4 2010-02-05      39954.04       False       5.727778
4           1           5 2010-02-05      32229.38       False       5.727778
…       …         …         …            …          …              …
413114     45           4 2012-10-26      24627.94       False      14.916667
413115     45           5 2012-10-26      13256.59       False      14.916667
413116     45           6 2012-10-26       1086.31       False      14.916667
413117     45           7 2012-10-26      20356.73       False      14.916667
413118     45           8 2012-10-26      37857.64       False      14.916667

        fuel_price_usd_per_l  unemployment
```

```
0                   0.679451          8.106
1                   0.679451          8.106
2                   0.679451          8.106
3                   0.679451          8.106
4                   0.679451          8.106
…                      …              …
413114              1.025516          8.667
413115              1.025516          8.667
413116              1.025516          8.667
413117              1.025516          8.667
413118              1.025516          8.667

[413119 rows x 8 columns]
```

Subsetting with .loc[]

[74]:
```
# Making a list of type to subset on
types = ["A", "B", "C"]

# Subsetting temperatures using square brackets
print(sales[sales["type"].isin(types)])

# Subsetting temperatures_ind using .loc[]
print(sales_ind.loc[types])
```

```
        store type  department       date  weekly_sales  is_holiday  \
0           1    A           1 2010-02-05      24924.50       False
1           1    A           2 2010-02-05      50605.27       False
2           1    A           3 2010-02-05      13740.12       False
3           1    A           4 2010-02-05      39954.04       False
4           1    A           5 2010-02-05      32229.38       False
…       …   …          …         …            …          …
413114     45    B           4 2012-10-26      24627.94       False
413115     45    B           5 2012-10-26      13256.59       False
413116     45    B           6 2012-10-26       1086.31       False
413117     45    B           7 2012-10-26      20356.73       False
413118     45    B           8 2012-10-26      37857.64       False

        temperature_c  fuel_price_usd_per_l  unemployment
0            5.727778              0.679451         8.106
1            5.727778              0.679451         8.106
2            5.727778              0.679451         8.106
3            5.727778              0.679451         8.106
4            5.727778              0.679451         8.106
…                 …                     …            …
413114      14.916667              1.025516         8.667
413115      14.916667              1.025516         8.667
413116      14.916667              1.025516         8.667
```

```
413117      14.916667              1.025516         8.667
413118      14.916667              1.025516         8.667

[413119 rows x 9 columns]
      store  department       date  weekly_sales  is_holiday  temperature_c  \
type
A         1           1 2010-02-05      24924.50       False       5.727778
A         1           2 2010-02-05      50605.27       False       5.727778
A         1           3 2010-02-05      13740.12       False       5.727778
A         1           4 2010-02-05      39954.04       False       5.727778
A         1           5 2010-02-05      32229.38       False       5.727778
…     …         …         …             …             …              …
C        44          94 2012-10-26      26641.59       False       8.316667
C        44          95 2012-10-26      32196.45       False       8.316667
C        44          96 2012-10-26       2983.19       False       8.316667
C        44          97 2012-10-26       7054.80       False       8.316667
C        44          98 2012-10-26       4348.96       False       8.316667

      fuel_price_usd_per_l  unemployment
type
A                 0.679451         8.106
A                 0.679451         8.106
A                 0.679451         8.106
A                 0.679451         8.106
A                 0.679451         8.106
…                    …           …
C                 0.991967         5.217
C                 0.991967         5.217
C                 0.991967         5.217
C                 0.991967         5.217
C                 0.991967         5.217

[413119 rows x 8 columns]
```

[75]:
```
# Setting an index allows more concise code for subsetting rows via .loc[].

# Sorting temperatures_ind by index values
print(sales_ind.sort_index())

# Sorting temperatures_ind by index values at the city level
print(sales_ind.sort_index(level="type"))

# Sorting temperatures_ind by country then descending city
print(sales_ind.sort_index(level=["type", "department"], ascending = [False,␣
 ↪True]))
```

```
      store  department       date  weekly_sales  is_holiday  temperature_c  \
type
```

```
A         1           1 2010-02-05      24924.50       False       5.727778
A        32          40 2011-04-15      59926.33       False       7.761111
A        32          38 2011-04-15      60815.01       False       7.761111
A        32          37 2011-04-15       2930.64       False       7.761111
A        32          36 2011-04-15        949.00       False       7.761111
…     …          …         …           …          …            …
C        37           4 2011-12-02      18770.90       False      11.983333
C        37           3 2011-12-02       1237.25       False      11.983333
C        37           2 2011-12-02      15704.18       False      11.983333
C        37          98 2011-11-25       7436.12        True      19.116667
C        37          98 2010-04-23       5022.08       False      20.016667

      fuel_price_usd_per_l  unemployment
type
A                 0.679451         8.106
A                 0.953926         8.595
A                 0.953926         8.595
A                 0.953926         8.595
A                 0.953926         8.595
…                    …           …
C                 0.837954         7.716
C                 0.837954         7.716
C                 0.837954         7.716
C                 0.854861         7.716
C                 0.738361         8.464

[413119 rows x 8 columns]
      store  department       date  weekly_sales  is_holiday  temperature_c  \
type
A         1           1 2010-02-05      24924.50       False       5.727778
A        32          40 2011-04-15      59926.33       False       7.761111
A        32          38 2011-04-15      60815.01       False       7.761111
A        32          37 2011-04-15       2930.64       False       7.761111
A        32          36 2011-04-15        949.00       False       7.761111
…     …          …         …           …          …            …
C        37           4 2011-12-02      18770.90       False      11.983333
C        37           3 2011-12-02       1237.25       False      11.983333
C        37           2 2011-12-02      15704.18       False      11.983333
C        37          98 2011-11-25       7436.12        True      19.116667
C        37          98 2010-04-23       5022.08       False      20.016667

      fuel_price_usd_per_l  unemployment
type
A                 0.679451         8.106
A                 0.953926         8.595
A                 0.953926         8.595
A                 0.953926         8.595
A                 0.953926         8.595
```

```
…                    …           …
C                 0.837954        7.716
C                 0.837954        7.716
C                 0.837954        7.716
C                 0.854861        7.716
C                 0.738361        8.464

[413119 rows x 8 columns]
      store  department       date  weekly_sales  is_holiday  temperature_c  \
type
A         1           1 2010-02-05      24924.50       False       5.727778
A        32          40 2011-04-15      59926.33       False       7.761111
A        32          38 2011-04-15      60815.01       False       7.761111
A        32          37 2011-04-15       2930.64       False       7.761111
A        32          36 2011-04-15        949.00       False       7.761111
…     …         …         …           …         …           …
C        37           4 2011-12-02      18770.90       False      11.983333
C        37           3 2011-12-02       1237.25       False      11.983333
C        37           2 2011-12-02      15704.18       False      11.983333
C        37          98 2011-11-25       7436.12        True      19.116667
C        37          98 2010-04-23       5022.08       False      20.016667

      fuel_price_usd_per_l  unemployment
type
A                 0.679451        8.106
A                 0.953926        8.595
A                 0.953926        8.595
A                 0.953926        8.595
A                 0.953926        8.595
…                    …           …
C                 0.837954        7.716
C                 0.837954        7.716
C                 0.837954        7.716
C                 0.854861        7.716
C                 0.738361        8.464

[413119 rows x 8 columns]
```

[76]:
```
Sales_srt = sales.set_index(['type', 'department']).sort_index()
print(Sales_srt)
```

```
                 store       date  weekly_sales  is_holiday  temperature_c  \
type department
A    1               1 2010-02-05      24924.50       False       5.727778
     1               1 2010-02-12      46039.49        True       3.616667
     1               1 2010-02-19      41595.55       False       4.405556
     1               1 2010-02-26      19403.54       False       8.127778
     1               1 2010-03-05      21827.90       False       8.055556
```

```
…                 …        …             …          …              …
C    99             43 2011-11-18         50.00       False      14.305556
     99             43 2012-01-06         25.00       False       8.661111
     99             43 2012-06-15          5.00       False      30.972222
     99             44 2010-03-05          0.01       False       4.805556
     99             44 2010-07-30          7.00       False      26.077778

                fuel_price_usd_per_l  unemployment
type department
A    1                      0.679451         8.106
     1                      0.673111         8.106
     1                      0.664129         8.106
     1                      0.676545         8.106
     1                      0.693452         8.106
…                              …             …
C    99                     0.873882        10.148
     99                     0.833992         9.653
     99                     0.896336         9.575
     99                     0.708246         8.119
     99                     0.738890         7.804

[413119 rows x 7 columns]
```

Weekly US avocado data

[77]:
```
import pickle
with open('avoplotto.pkl', 'rb') as f:
    data = pickle.load(f)
```

[78]:
```
avocados = data
print(avocados.head())
```

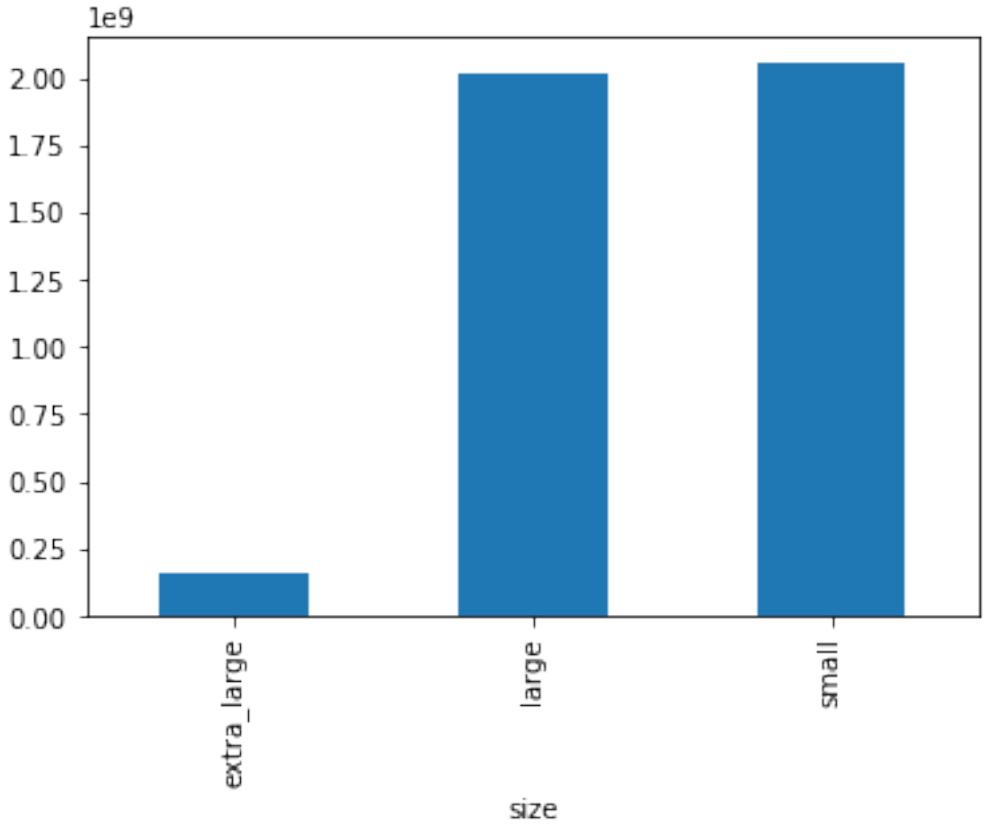
```
         date          type  year  avg_price   size     nb_sold
0  2015-12-27  conventional  2015       0.95  small  9626901.09
1  2015-12-20  conventional  2015       0.98  small  8710021.76
2  2015-12-13  conventional  2015       0.93  small  9855053.66
3  2015-12-06  conventional  2015       0.89  small  9405464.36
4  2015-11-29  conventional  2015       0.99  small  8094803.56
```

Answering the question, which avocado size is most popular?

[80]:
```
import pandas as pd

# Importing matplotlib.pyplot with alias plt
import matplotlib.pyplot as plt

# Looking at the first few rows of data
print(avocados.head())
```

```
# Getting the total number of avocados sold of each size
nb_sold_by_size = avocados.groupby('size')['nb_sold'].sum()

# Creating a bar plot of the number of avocados sold by size
nb_sold_by_size.plot(kind='bar')

# Showing the plot
plt.show()
```

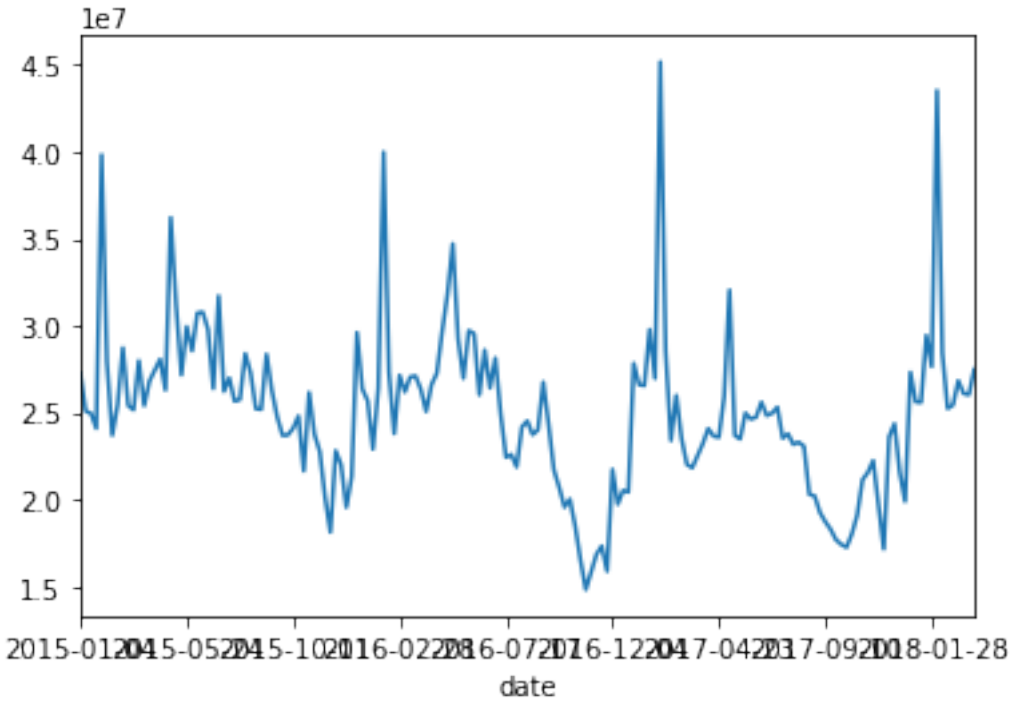
```
         date          type  year  avg_price   size     nb_sold
0  2015-12-27  conventional  2015       0.95  small  9626901.09
1  2015-12-20  conventional  2015       0.98  small  8710021.76
2  2015-12-13  conventional  2015       0.93  small  9855053.66
3  2015-12-06  conventional  2015       0.89  small  9405464.36
4  2015-11-29  conventional  2015       0.99  small  8094803.56
```



It looks like small avocados were the most-purchased size, but large avocados were a close second.

Trying to find changes in sales over time

[81]:
```
# Importing matplotlib.pyplot with alias plt
import matplotlib.pyplot as plt

# Getting the total number of avocados sold on each date
nb_sold_by_date = avocados.groupby('date')['nb_sold'].sum()

# Creating a line plot of the number of avocados sold by date
nb_sold_by_date.plot()

# Showing the plot
plt.show()
```



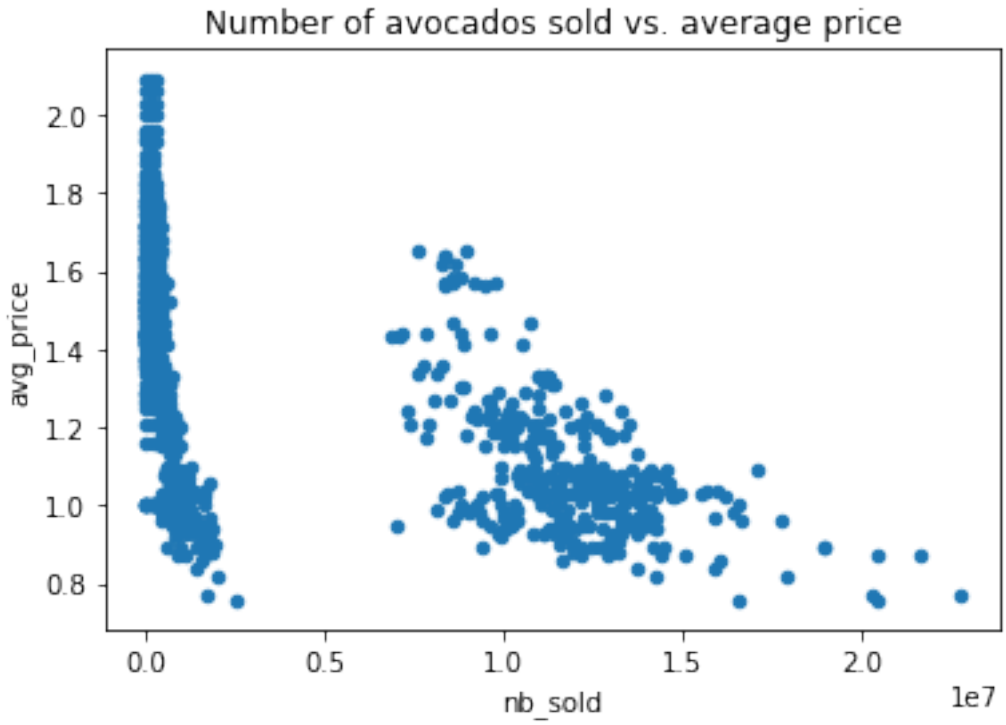
Line plots are great for visualizing something over time. Here, it looks like the number of avocados spikes around the same time each year.

Avocado supply and demand

Scatter plots are ideal for visualizing relationships between numerical variables. In this exercise, we will compare the number of avocados sold to average price and see if they're at all related. If they're related, we may be able to use one number to predict the other.

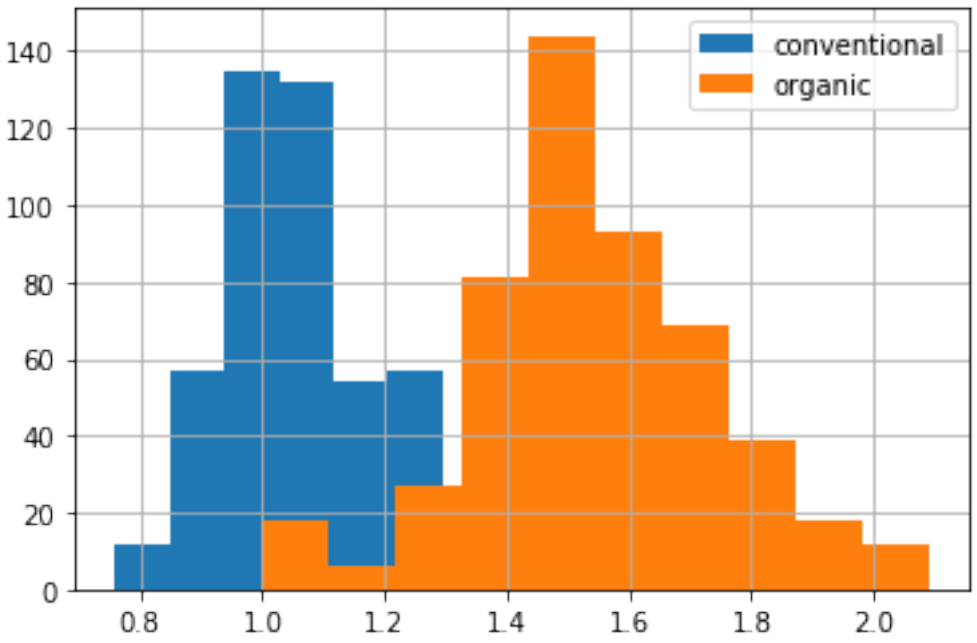
[82]:
```
# Scatter plot of nb_sold vs avg_price with title
avocados.plot(x='nb_sold', y='avg_price', kind='scatter', title='Number of␣
 ↪avocados sold vs. average price')
```

```
# Show the plot
plt.show()
```

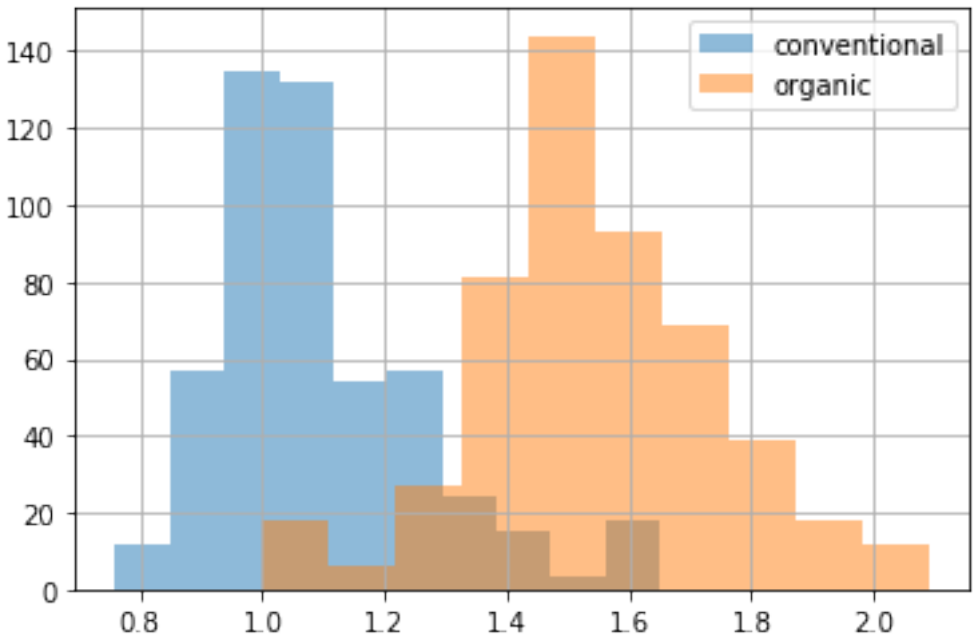


It looks like when more avocados are sold, prices go down. However, this doesn't mean that fewer sales causes higher prices - we can only tell that they're correlated with each other.

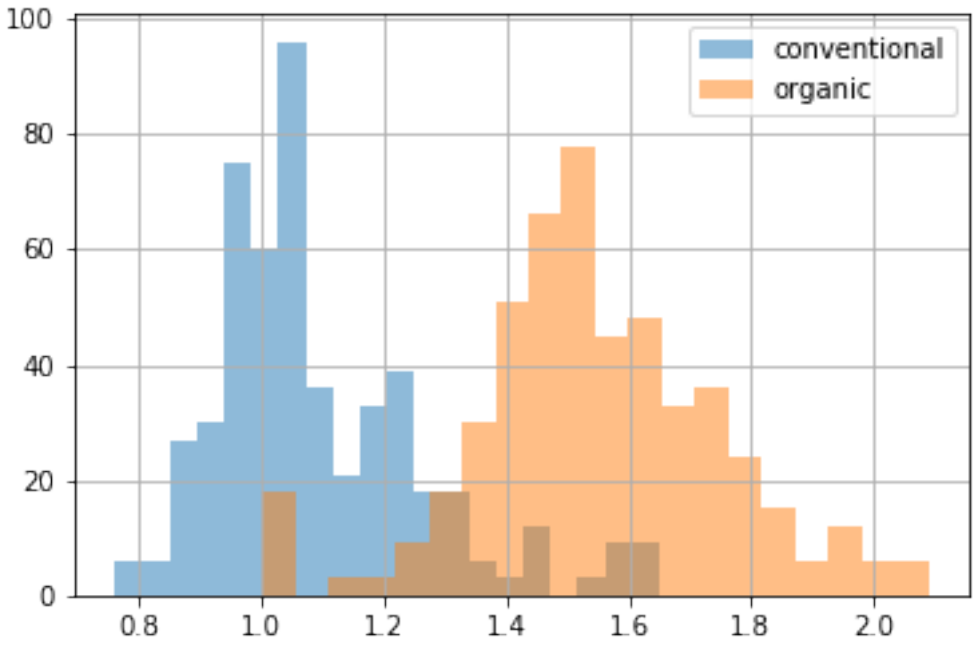
[83]:
```
# Histogram of conventional avg_price
avocados[avocados['type']=='conventional']['avg_price'].hist()

# Histogram of organic avg_price
avocados[avocados['type']=='organic']['avg_price'].hist()

# Adding a legend
plt.legend(['conventional','organic'])

# Showing the plot
plt.show()
```



[84]:
```
# Modifying histogram transparency to 0.5
avocados[avocados["type"] == "conventional"]["avg_price"].hist(alpha=0.5)

# Modifying histogram transparency to 0.5
avocados[avocados["type"] == "organic"]["avg_price"].hist(alpha=0.5)

# Adding a legend
plt.legend(["conventional", "organic"])

# Showing the plot
plt.show()
```



[85]:
```
# Modifying bins to 20
avocados[avocados["type"] == "conventional"]["avg_price"].hist(alpha=0.5,␣
↪bins=20)

# Modifying bins to 20
avocados[avocados["type"] == "organic"]["avg_price"].hist(alpha=0.5, bins=20)

# Adding a legend
plt.legend(["conventional", "organic"])

# Showing the plot
plt.show()
```



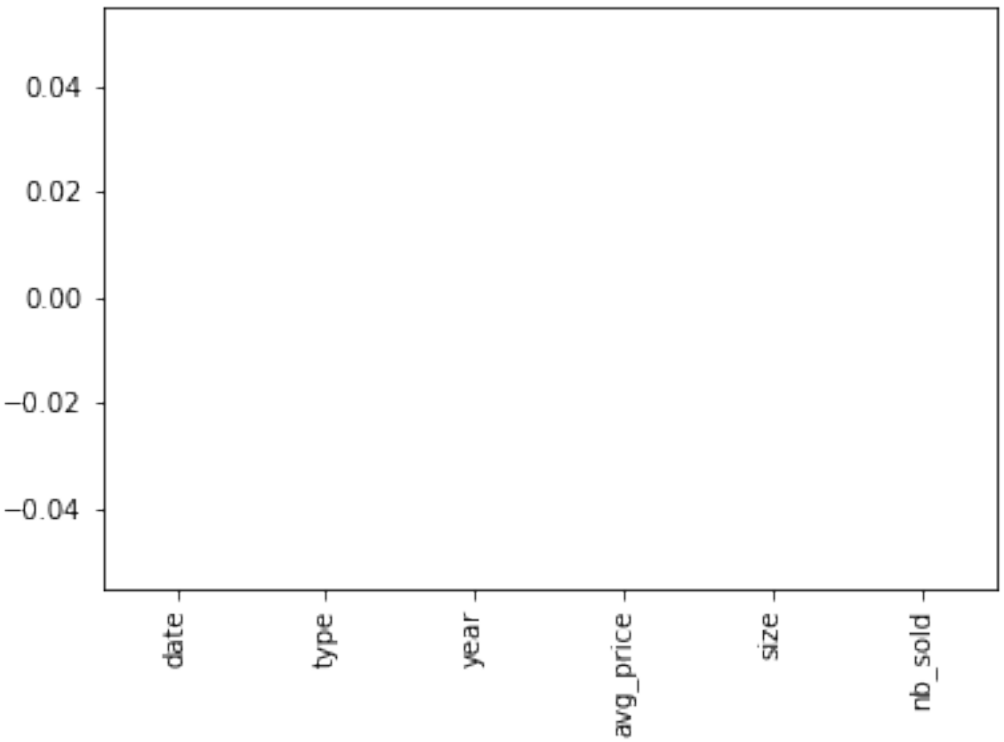
We can see that on average, organic avocados are more expensive than conventional ones, but their price distributions have some overlap.

Finding missing values.

[86]:
```
# subsetting data for analysis
avocados_2016 = avocados[avocados["year"] == 2016]

# Importing matplotlib.pyplot with alias plt
import matplotlib.pyplot as plt

# Checking individual values for missing values
print(avocados_2016.isna())

# Checking each column for missing values
print(avocados_2016.isna().any())

# Bar plot of missing values by variable
avocados_2016.isna().sum().plot(kind='bar')

# Showing the plot
plt.show()
```

```
      date   type   year  avg_price   size  nb_sold
52   False  False  False      False  False    False
53   False  False  False      False  False    False
54   False  False  False      False  False    False
```

```
55   False  False  False      False  False    False
56   False  False  False      False  False    False
..     …    …    …        …    …      …
944  False  False  False      False  False    False
945  False  False  False      False  False    False
946  False  False  False      False  False    False
947  False  False  False      False  False    False
948  False  False  False      False  False    False

[312 rows x 6 columns]
date         False
type         False
year         False
avg_price    False
size         False
nb_sold      False
dtype: bool
```



Above results indicate there is no missing value.

List of dictionaries. Below dictionaries can be used to subset and analyze data.

[87]:
```
# Creating a list of dictionaries with new data
avocados_list = [
```

```
    {"date": "2019-11-03", "small_sold": 10376832, "large_sold": 7835071},
    {"date": "2019-11-10", "small_sold": 10717154, "large_sold": 8561348},
]
```

[88]:
```
# Converting list into DataFrame
avocados_2019 = pd.DataFrame(avocados_list)

# Printing the new DataFrame
print(avocados_2019)
```

```
         date  small_sold  large_sold
0  2019-11-03    10376832     7835071
1  2019-11-10    10717154     8561348
```

[89]:
```
# Dictionary of lists
# Creating a dictionary of lists with new data
avocados_dict = {
  "date": ["2019-11-17", "2019-12-01"],
  "small_sold": [10859987, 9291631],
  "large_sold": [7674135, 6238096]
}
```

[90]:
```
# Converting dictionary into DataFrame
avocados_2019 = pd.DataFrame(avocados_dict)

# Printing the new DataFrame
print(avocados_2019)
```

```
         date  small_sold  large_sold
0  2019-11-17    10859987     7674135
1  2019-12-01     9291631     6238096
```

The list-of-dictionaries method creates DataFrames column-by-column.

Thanks a lot for your attention.

[ ]: