realtor_analysis

October 29, 2021

[1]:
```python
import boto3 # AWS SDK
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
```

[2]:
```python
from matplotlib import pyplot as plt
import numpy as np
import pandas as pd
import seaborn as sns
plt.style.use('fivethirtyeight')
sns.set_style('darkgrid')
sns.set_context('notebook', font_scale=1.5)
sns.set_palette('colorblind')
%matplotlib inline
%pylab inline
```

```
Populating the interactive namespace from numpy and matplotlib
```

1 Data Introduction

[3]:
```python
# bucket = "realtor-data"
# file_name = "realtor_data.csv"

# session = boto3.Session(profile_name='simon')
# s3 = boto3.client('s3')
# obj = s3.get_object(Bucket= bucket, Key= file_name)
housing = pd.read_csv('vancouver_real_estate.csv')
features = ['address','longitude', 'latitude', 'interior_size',␣
↪'building_type', 'bedrooms', 'bathrooms', 'price']
housing.head()
```

[3]:
```
         id                                           address   longitude  \
0  23722871  1107 11967 80 AVENUE|DELTA, British Columbia V… -122.892117
1  23722443  11724 KINGSBRIDGE DRIVE|Richmond, British Colu… -123.095126
2  23722322  3704 1189 MELVILLE STREET|Vancouver, British C… -123.123645
3  23722010  1056 E 14TH AVENUE|Vancouver, British Columbia… -123.082073
4  23721920  205 2119 BELLEVUE AVENUE|West Vancouver, Briti… -123.168890
```

```
    latitude  interior_size    building_type  bedrooms  bathrooms  \
0  49.148822          821.0        Apartment       2.0        2.0
1  49.147349         1074.0  Row / Townhouse       2.0        1.0
2  49.287795         1414.0        Apartment       2.0        3.0
3  49.257664         2210.0            House       4.0        3.0
4  49.329376          801.0        Apartment       2.0        1.0

          agent_name  area_code phone_number    price
0       Calvin Khara      778.0     869-4875   579000
1        Emily Ching      604.0     722-9655   488800
2      Iman Moghadam      604.0     721-6209  1758800
3  Charlie MacKenzie      604.0     787-2188  1848000
4         John Doyle      604.0     726-4516   773500
```

[4]:
```
housing.describe()
```

[4]:
```
                 id   longitude    latitude  interior_size    bedrooms  \
count  6.000000e+02  600.000000  600.000000     552.000000  586.000000
mean   2.371321e+07 -123.023231   49.236684    1558.000000    2.696246
std    4.387682e+03    0.131096    0.055704    1220.329302    1.614005
min    2.370669e+07 -123.277173   49.125920     410.000000    0.000000
25%    2.370999e+07 -123.127794   49.189511     775.750000    2.000000
50%    2.371210e+07 -123.072095   49.238430    1060.000000    2.000000
75%    2.371642e+07 -122.904077   49.276286    2006.250000    3.000000
max    2.372287e+07 -122.731940   49.364633   12413.000000    8.000000

        bathrooms   area_code         price
count  586.000000  597.000000  6.000000e+02
mean     2.264505  654.723618  1.404812e+06
std      1.349898   82.197885  1.786787e+06
min      0.000000  236.000000  2.199000e+05
25%      1.000000  604.000000  6.799750e+05
50%      2.000000  604.000000  9.399000e+05
75%      3.000000  778.000000  1.599000e+06
max      9.000000  866.000000  3.399000e+07
```

[5]:
```
housing.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 600 entries, 0 to 599
Data columns (total 12 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   id             600 non-null    int64
 1   address        600 non-null    object
 2   longitude      600 non-null    float64
 3   latitude       600 non-null    float64
```

```
 4   interior_size  552 non-null    float64
 5   building_type  585 non-null    object
 6   bedrooms       586 non-null    float64
 7   bathrooms      586 non-null    float64
 8   agent_name     600 non-null    object
 9   area_code      597 non-null    float64
 10  phone_number   597 non-null    object
 11  price          600 non-null    int64
dtypes: float64(6), int64(2), object(4)
memory usage: 56.4+ KB
```

[6]:
```
housing.building_type.unique()
```

[6]:
```
array(['Apartment', 'Row / Townhouse', 'House', nan, 'Duplex',
       'Manufactured Home'], dtype=object)
```

[7]:
```
housing.isnull().sum()
```

[7]:
```
id                0
address           0
longitude         0
latitude          0
interior_size    48
building_type    15
bedrooms         14
bathrooms        14
agent_name        0
area_code         3
phone_number      3
price             0
dtype: int64
```

2 Data Processing

I would first like to add another column called 'city' by parsing the address column.

[8]:
```
housing.address.sample(10)
```

[8]:
```
303    814 8699 HAZELBRIDGE WAY|Richmond, British Col…
316    706 1768 COOK STREET|Vancouver, British Columb…
81     315 3699 SEXSMITH ROAD|Richmond, British Colum…
385    19 205 LEBLEU STREET|Coquitlam, British Columb…
579    9 3298 E 54TH AVENUE|Vancouver, British Columb…
34     409 E 12TH STREET|North Vancouver, British Col…
66     414 5933 COONEY ROAD|Richmond, British Columbi…
458     6853 123A STREET|SURREY, British Columbia V3W0X4
236    918 WENTWORTH AVENUE|North Vancouver, British …
```

```
232    202 9228 SLOPES MEWS|Burnaby, British Columbia…
Name: address, dtype: object
```

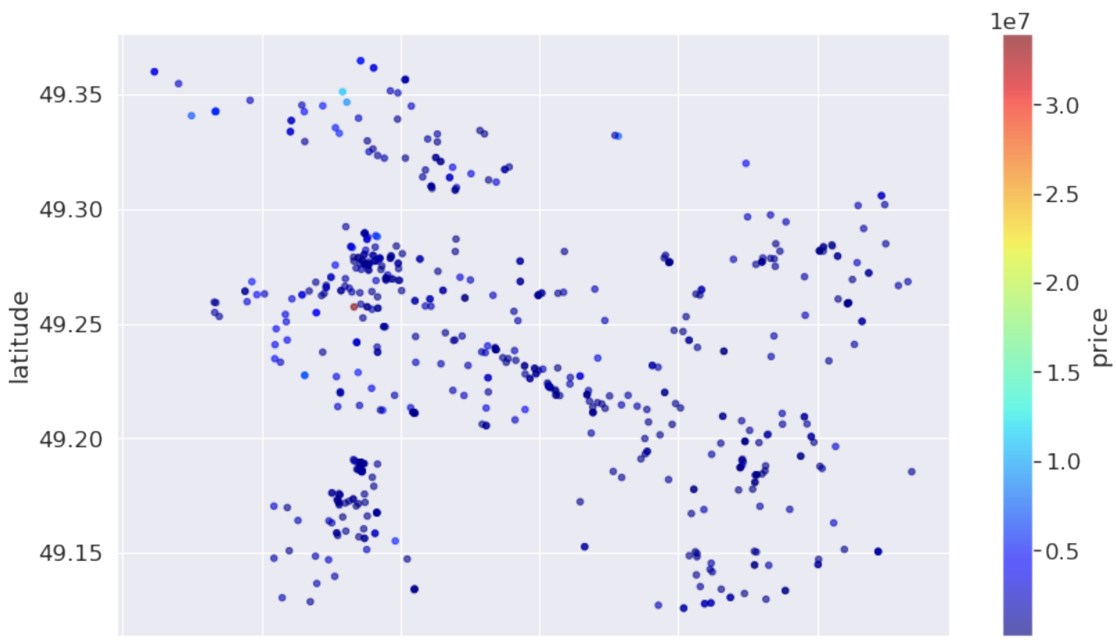
[9]:
```
housing['city'] = housing.address.map(lambda x: x.split('|')[-1].split(',')[0].
↪lower())
housing.city.unique()
```

[9]:
```
array(['delta', 'richmond', 'vancouver', 'west vancouver', 'coquitlam',
       'north vancouver', 'port moody', 'surrey', 'burnaby',
       'new westminster', 'port coquitlam', 'burnslk w'], dtype=object)
```

3 Exploaratory Data Analysis

[10]:
```
housing.plot(kind='scatter', x='longitude', y='latitude', alpha=0.6, c='price',␣
↪cmap=plt.get_cmap('jet'), colorbar=True, figsize=(12,8))
```

[10]:
```
<AxesSubplot:xlabel='longitude', ylabel='latitude'>
```



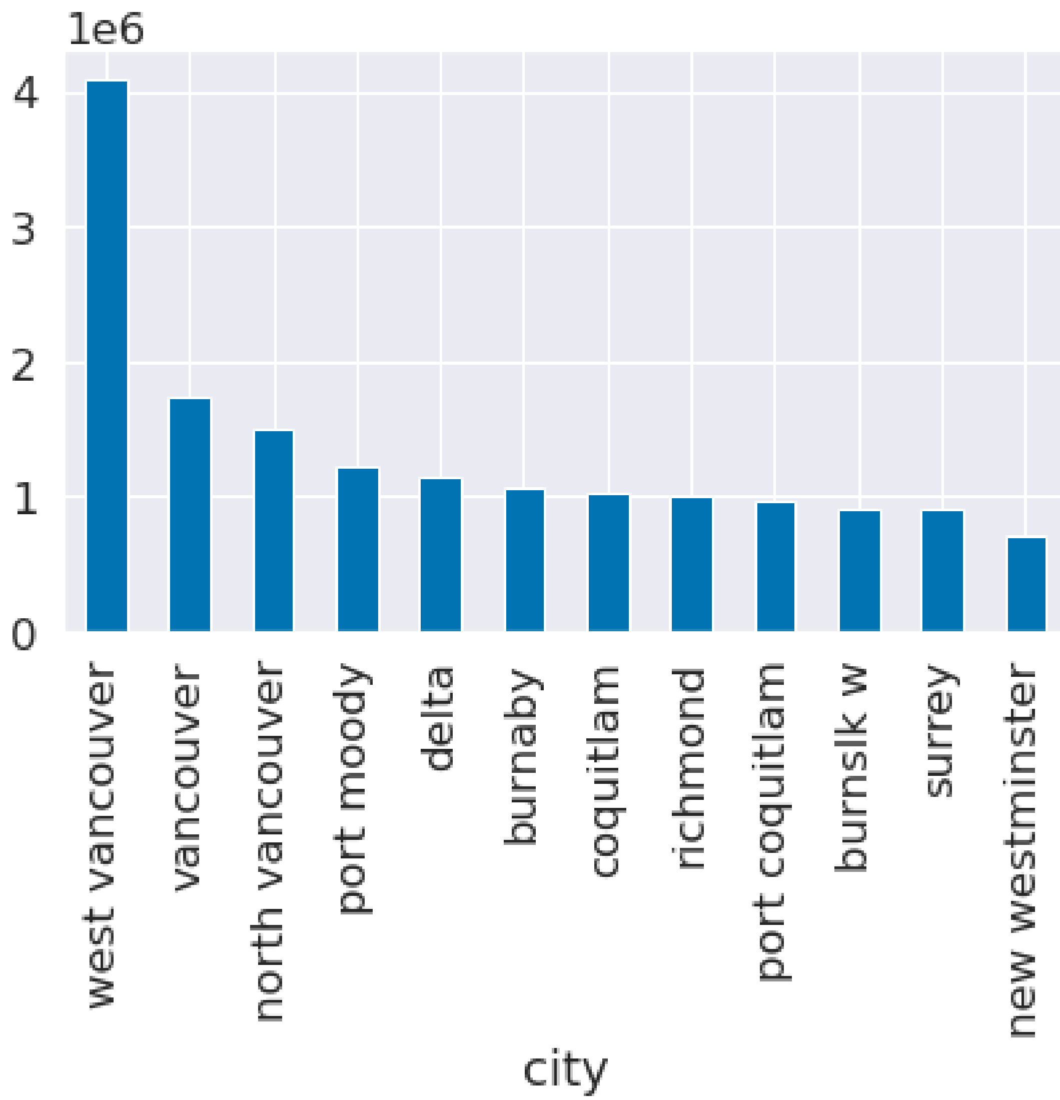
[11]:
```
# from pandas.plotting import scatter_matrix
# attributes = ['price', 'interior_size', 'bedrooms', 'bathrooms']
# scatter_matrix(housing[attributes], figsize=(20,20))
```

3.1 The Most Unaffortable City?

Let us analyze which cities are the most expensive to affort a house. First, we will simply find the average price of a house for each city.

[12]:
```
housing.groupby('city').price.mean().sort_values(ascending=False).plot.bar()
```

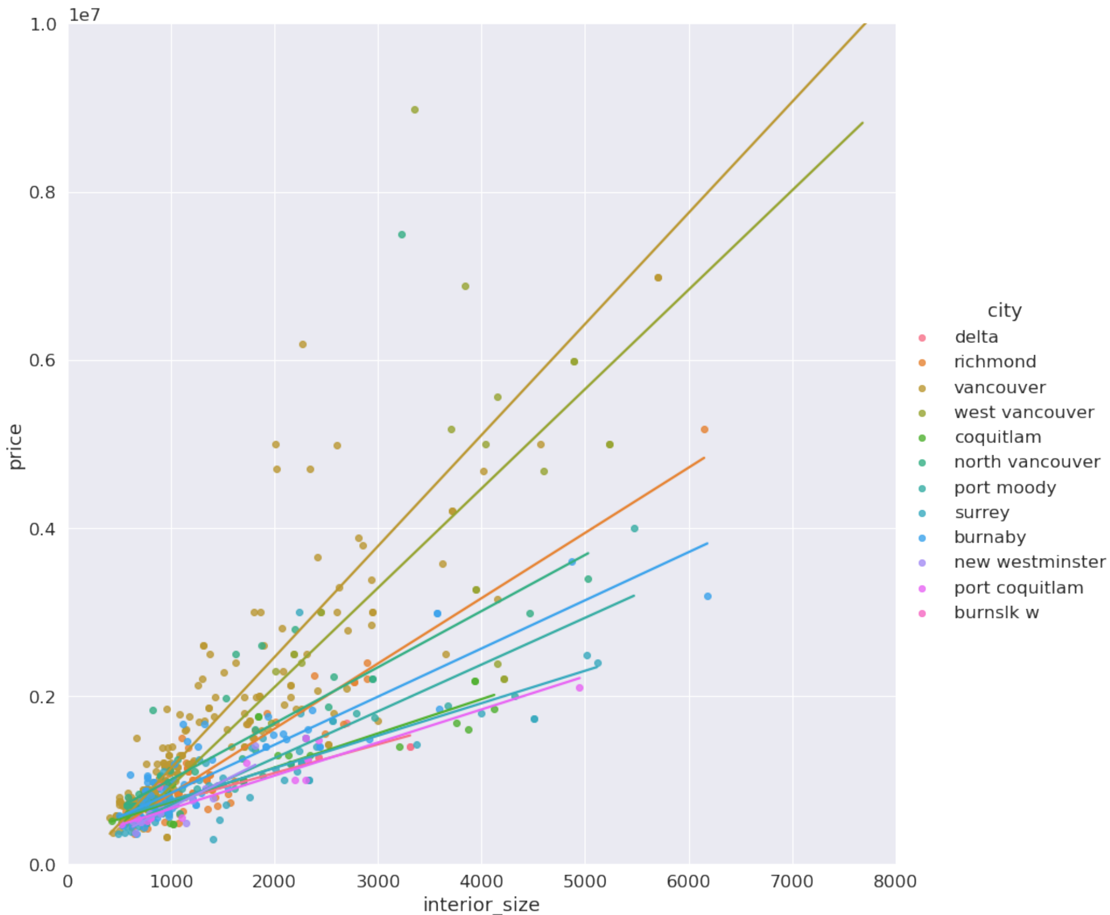
[12]: <AxesSubplot:xlabel='city'>



Well, it seems pretty clear who the winner is. However, is this the most accurate representation of affordability? One of the most important aspects people consider when looking for houses is simply the size of the interior space. Hence, let's fit a linear regression model of interior size vs. price.(p.s. I am ignoring outliers with robust option and setting confidence interval to 0 for faster calculation).

[13]:
```
sns.lmplot(x='interior_size', y='price', hue='city', data=housing, height=12,␣
 ↪robust=True, ci=None)
plt.xlim((0, 8000))
plt.ylim((0,1e7))
```

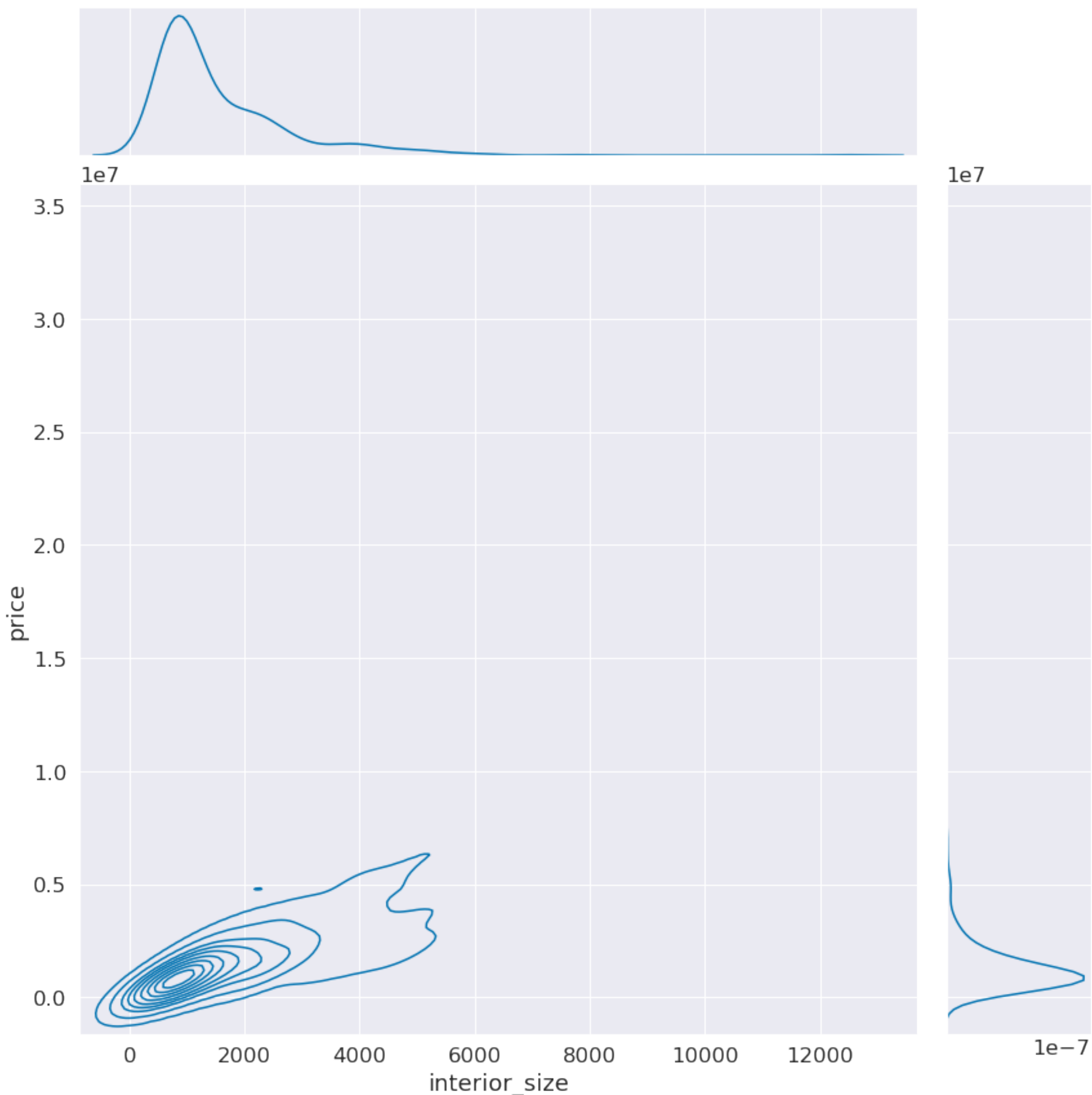
[13]: (0.0, 10000000.0)



Joint plot shows the marginal distribution of the x (interior size) and y (price) axis with histogram or kernel density estimation. The inner most enclosed area indicates the peak of joint density of interior size and price.

[14]:
```
sns.jointplot(x='interior_size', y='price', data=housing, height=12, kind='kde')
```

[14]: <seaborn.axisgrid.JointGrid at 0x7fe5c3805310>



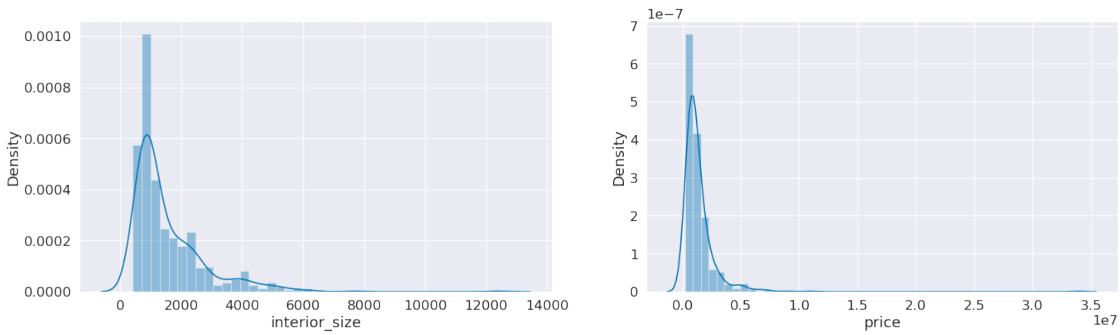
Separating the dimensions, `sns.distplot` can overlay a histogram with kernel density estimation, getting the best of both worlds!

[15]:
```
fig, ax = plt.subplots(1,2, figsize = (20,6))
sns.distplot(housing.interior_size, ax=ax[0])
sns.distplot(housing.price, ax=ax[1])
```

```
/opt/conda/lib/python3.9/site-packages/seaborn/distributions.py:2619:
FutureWarning: `distplot` is a deprecated function and will be removed in a
future version. Please adapt your code to use either `displot` (a figure-level
function with similar flexibility) or `histplot` (an axes-level function for
histograms).
  warnings.warn(msg, FutureWarning)
/opt/conda/lib/python3.9/site-packages/seaborn/distributions.py:2619:
```

```
FutureWarning: `distplot` is a deprecated function and will be removed in a
future version. Please adapt your code to use either `displot` (a figure-level
function with similar flexibility) or `histplot` (an axes-level function for
histograms).
  warnings.warn(msg, FutureWarning)
```

[15]: <AxesSubplot:xlabel='price', ylabel='Density'>



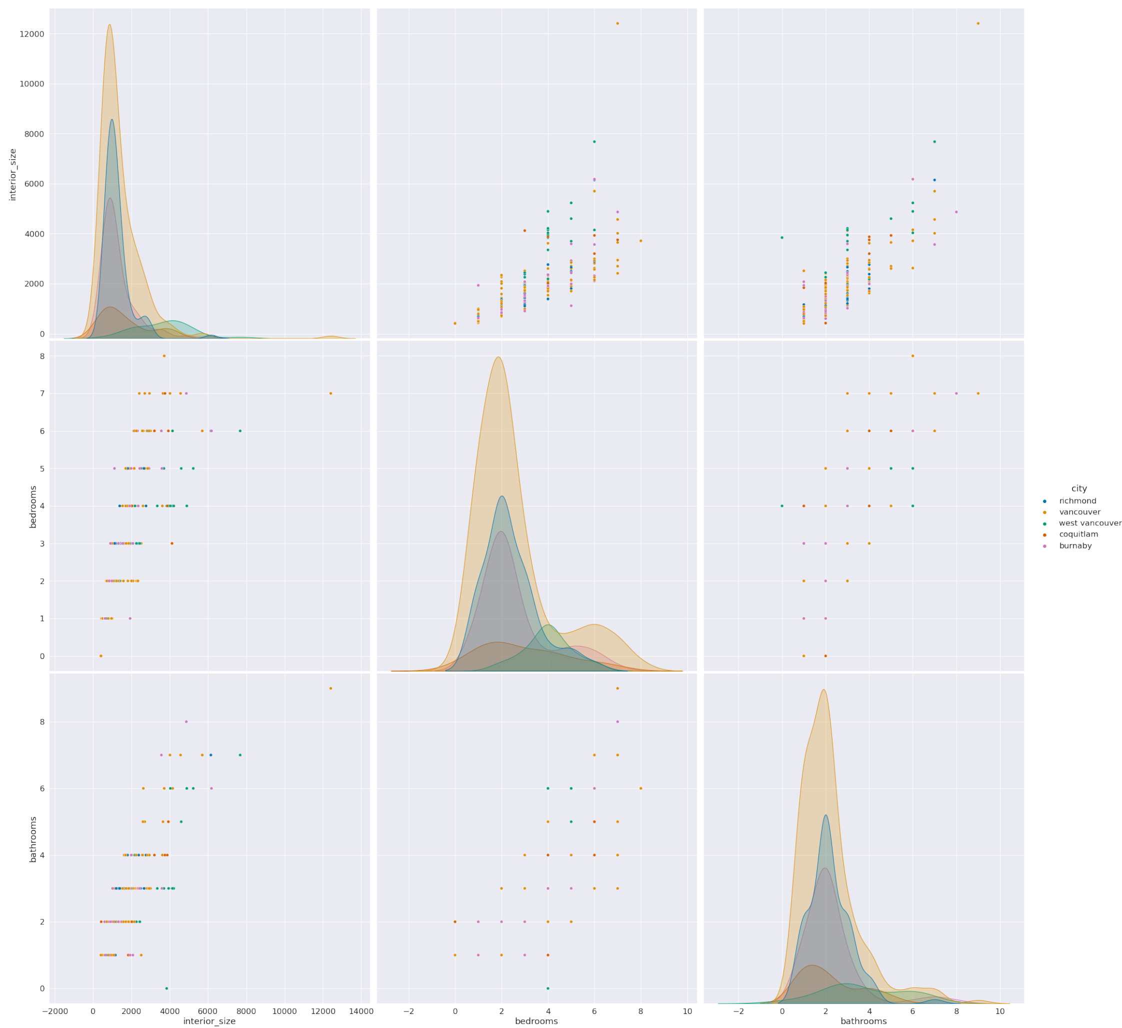
Although West Vancouver has the highest average price, when comparing the price per square foot it seems like Vancouver is actually slightly more unaffordable than West Vancouver. Let's dive deeper into some of the most expensive cities of the metropolitan area.

[16]:
```
exp_cities = ['vancouver', 'west vancouver', 'richmond', 'coquitlam', 'burnaby']
exp_housing = housing[housing.city.isin(exp_cities)]
```

Pair plot extends the same functionality of joint plot by plotting each two dimensional pairs selected.

[17]:
```
features = ['interior_size', 'bedrooms', 'bathrooms', 'city']
sns.pairplot(exp_housing[features], height=10, hue='city')
```

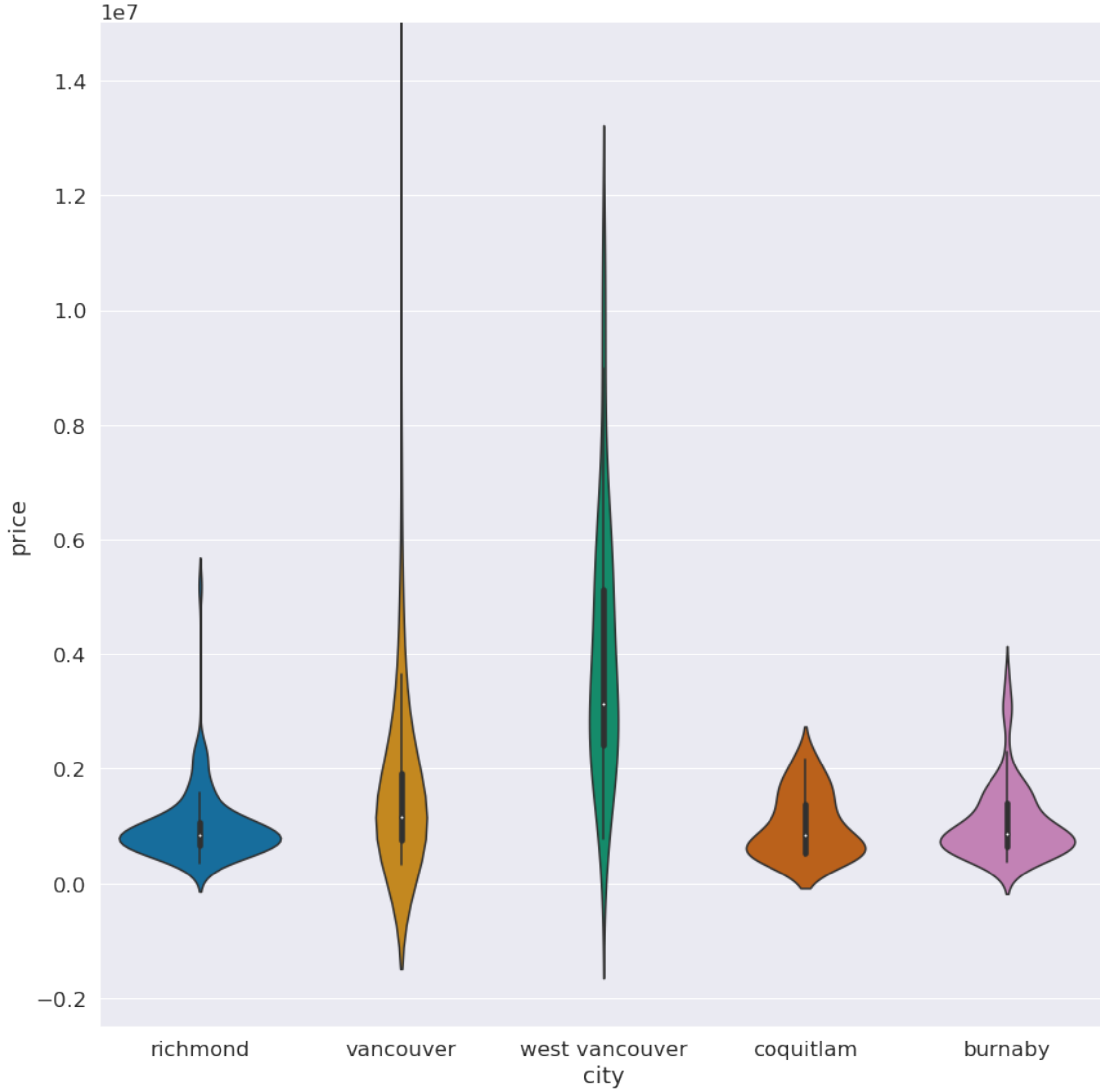
[17]: <seaborn.axisgrid.PairGrid at 0x7fe5c1aa6dc0>



For categorical plotting, **sns.catplot** is a great way to create boxplots and violin plots.

[18]:
```
sns.catplot(y='price', x='city', kind='violin', data=exp_housing, height=12)
plt.ylim((-0.25e7, 1.5e7))
```

[18]:
```
(-2500000.0, 15000000.0)
```



[19]:
```
import plotly.express as px
fig = px.density_mapbox(housing, lat='latitude', lon='longitude',␣
↪z='price',radius=5,
                        center=dict(lat=49.29225, lon=-123.14),zoom=10,
                        mapbox_style='stamen-terrain')
fig.show()
```


Burnaby
1
Map tiles by Stamen Design under CC BY 3.0
Data by OpenStreetMap contributors under ODbL
price
30M
20M
10M

```
[ ]:
```