Arizona Sold Home Prices 2021

By: Matt Cooper

This project uses Python for the whole data process. This dataset describes the sale prices of homes in Arizona during the 2021 year.

The Dataset Basics:

Dataset source: https://www.kaggle.com/antoniong203/arizona-houses-2021
Column Names:

- Price
- Address
- Local_area
- Zipcode
- Beds
- Baths
- Sqft
- URL (direct link to home info)

Number of data rows: 563

The Project:

Upon looking at the data, I decided that I wanted to develop three (3) questions to use this data to answer.

- Q1: What are the top 5 areas with the most home sales?
- Q2: What 'Local Area' has the highest average price?
- Q3: What is the average square footage of the most popular local?

I spent some time browsing the data and performing some exploratory analysis. I found a few aspects of the data that I wanted to handle before diving further into the analysis.

- 'local_area' was renamed 'city'
- 'Price' had skewed distribution, so outliers were removed
- 38 data points were removed
- One data point had non-numerical zip code and was removed
- 'url' and 'address' columns were removed

-------Code Starts-------

```
# -*- coding: utf-8 -*-
# Written by Matt Cooper

import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import scipy as sp
import collections as cl
import seaborn as sns

np.random.seed(1029384756)

#Loading the dataset
az_data =
pd.read_csv('C:/Datasets/Arizona_Home_Data/AZhousingData.csv')

#Printing a short version of the data for easy viewing
print(az_data.head(10))
#The URL and address columns make this printout useless

#Questions to answer with this dataset
# 1. What are the top 5 areas with the most home sales?
# 2. What 'Local Area' has the highest average price?
# 3. What is the average square footage of the most popular local?



#--------Initial Cleanup of dataset--------
#Removes the url and address columns as they are not needed
az_data = az_data.drop(columns=['url', 'address'])

#Printing a short version of the data for easy viewing
print(az_data.head(10))
```

	Price	Local_area	zipcode	beds	baths	sqft
0	229900	Phoenix	85029	2.0	3.0	1498.0
1	294900	Sahuarita	85629	4.0	3.0	1951.0
2	683100	Phoenix	85048	4.0	4.0	3110.0
3	260000	Scottsdale	85257	2.0	1.0	759.0
4	290900	San Tan Valley	85143	2.0	2.0	1052.0
5	377900	Buckeye	85326	5.0	3.0	2267.0
6	344900	Maricopa	85138	3.0	3.0	2200.0
7	269100	Phoenix	85037	2.0	1.0	911.0
8	700900	Gilbert	85296	3.0	3.0	2661.0
9	647900	Chandler	85249	3.0	2.0	2309.0

```
#Prints the details of the data
print(az_data.isna().sum())
```

```
Price         0
Local_area    0
zipcode       0
beds          0
baths         0
sqft          0
dtype: int64
```

```
#Prints the details of the data
print(az_data.info())
```

```
RangeIndex: 563 entries, 0 to 562
Data columns (total 6 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   Price       563 non-null    int64
 1   Local_area  563 non-null    object
 2   zipcode     563 non-null    object
 3   beds        563 non-null    float64
 4   baths       563 non-null    float64
 5   sqft        563 non-null    float64
dtypes: float64(3), int64(1), object(2)
memory usage: 26.5+ KB
None
```

```
#Prints the details of the data
print(az_data.describe())
```

	Price	beds	baths	sqft
count	5.630000e+02	563.000000	563.000000	563.000000
mean	5.455896e+05	3.294849	2.477798	2085.946714
std	7.317538e+05	0.972079	0.861878	936.653497
min	1.775000e+04	1.000000	1.000000	420.000000
25%	3.208775e+05	3.000000	2.000000	1463.000000
50%	4.099000e+05	3.000000	2.000000	1903.000000
75%	5.950000e+05	4.000000	3.000000	2456.500000
max	1.500000e+07	8.000000	8.000000	8625.000000

To start off, I wanted to take a quick look at the data to see what I'm working with and to see where any initial errors may be. A few items need to be addressed.

```
#Rename column 'local_area' to 'city'
az_data = az_data.rename(columns={'Local_area' : 'city'})

#Rename column 'Price' to 'price'
az_data = az_data.rename(columns={'Price' : 'price'})

#Re-type 'beds' and 'sqft'
az_data = az_data.astype({'beds': 'int64'})
az_data = az_data.astype({'sqft': 'int64'})
```

```
#Printing a short version of the data for easy viewing
print(az_data.head(10))
```

	price	city	zipcode	beds	baths	sqft
0	229900	Phoenix	85029	2	3.0	1498
1	294900	Sahuarita	85629	4	3.0	1951
2	683100	Phoenix	85048	4	4.0	3110
3	260000	Scottsdale	85257	2	1.0	759
4	290900	San Tan Valley	85143	2	2.0	1052
5	377900	Buckeye	85326	5	3.0	2267
6	344900	Maricopa	85138	3	3.0	2200
7	269100	Phoenix	85037	2	1.0	911
8	700900	Gilbert	85296	3	3.0	2661
9	647900	Chandler	85249	3	2.0	2309

```
#----------Finding why the zipcode column was type object and not
integer
#----------and fixing the problem
#Sort the 'zipcode' data to find error
az_data = az_data.sort_values(by=['zipcode'], ascending= False)

#Printing a short version of the data for easy viewing
print(az_data.head(10))
```

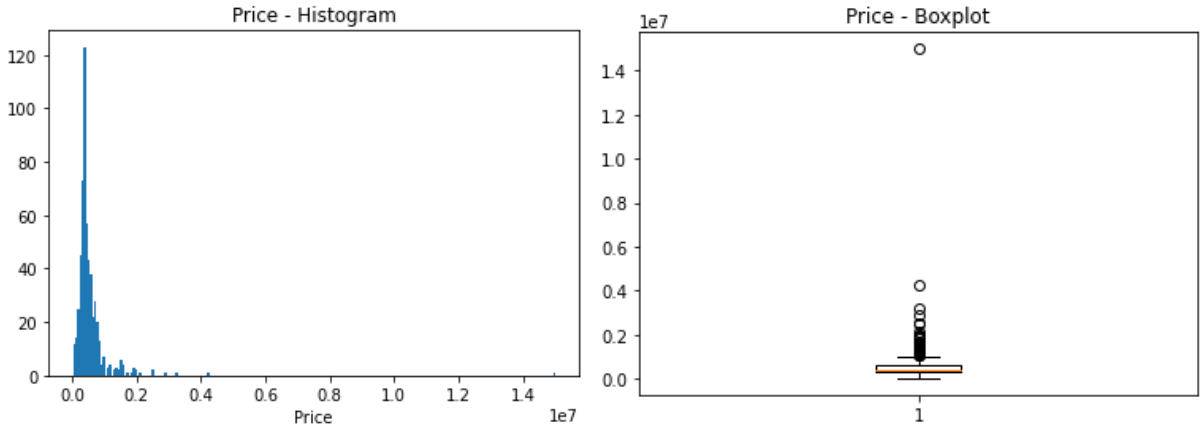
	price	city	zipcode	beds	baths	sqft
549	397500	AVE	Apache County	2	2.0	1440
179	363930	Bullhead City	86442	2	3.0	1715
186	529737	Bullhead City	86442	3	4.0	2800
436	125000	Bullhead City	86442	2	2.0	1456
282	40000	Bullhead City	86442	1	1.0	500
453	560000	Bullhead City	86442	4	3.0	3294
351	394135	Bullhead City	86442	2	3.0	1903
246	449825	Bullhead City	86442	2	3.0	1903
204	50000	Bullhead City	86442	1	1.0	420
273	50000	Bullhead City	86442	2	1.0	720

```
#Removes the string error in the zipcode column
az_data = az_data.iloc[1:]

#Re-type 'zipcode'
az_data = az_data.astype({'zipcode': 'int64'})

#Printing a short version of the data for easy viewing
print(az_data.head(10))
```

	price	city	zipcode	beds	baths	sqft
179	363930	Bullhead City	86442	2	3.0	1715
186	529737	Bullhead City	86442	3	4.0	2800
436	125000	Bullhead City	86442	2	2.0	1456
282	40000	Bullhead City	86442	1	1.0	500
453	560000	Bullhead City	86442	4	3.0	3294
351	394135	Bullhead City	86442	2	3.0	1903
246	449825	Bullhead City	86442	2	3.0	1903
204	50000	Bullhead City	86442	1	1.0	420
273	50000	Bullhead City	86442	2	1.0	720
526	679000	Bullhead City	86442	3	3.0	2241

The initial fixes are complete. Now, to start looking closer at the actual data for errors.

```
#----------Checking price for extreme outliers

#histogram of Price
x = az_data["price"]
plt.title("Price - Histogram")
plt.xlabel("Price")
plt.hist(x, bins='auto')
plt.show()

# Creating boxplot
plt.title("Price - Boxplot")
plt.boxplot(az_data.price)
plt.show()
```



In the 'Price' visualizations, we see several data points as outliers. I wanted to get a numerical determination to assist in the removal of the true outliers.

```
#----------Identifying and counting the outliers in 'price'

# calculating the 1st quartile
q1 = np.quantile(az_data.price, 0.25)

# calculating the 3rd quartile
q3 = np.quantile(az_data.price, 0.75)
med = np.median(az_data.price)

# calculating the iqr region
iqr = q3-q1
```

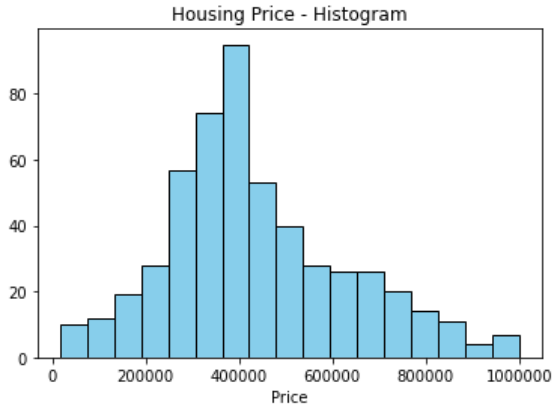
```
# calculating upper and lower whiskers
upper_bound = q3+(1.5*iqr)
lower_bound = q1-(1.5*iqr)
print("IGR: " + str(iqr),"     " , "Upper Bound: " +
str(upper_bound),"     " , "Lower Bound: " + str(lower_bound))
```

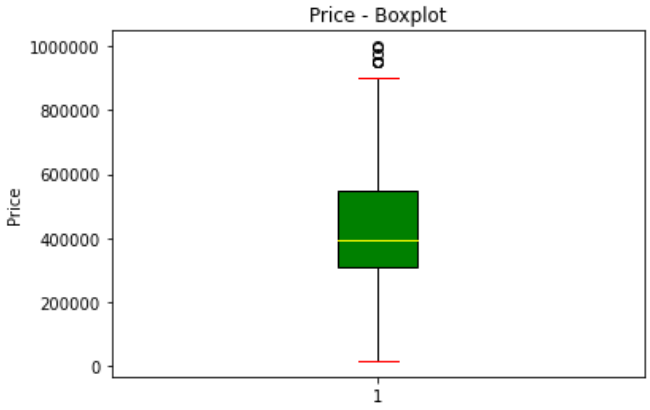
```
IGR: 274561.25       Upper Bound: 1006841.875       Lower Bound: -91403.125
```

The above code provides me with the upper and lower bounds (cutoffs). The lower bound in this case doesn't mean anything and can be ignored. We are more interested in the upper bound. Any data point above that value is considered an outlier and can be removed. Before I remove the data, I'd like to know how many data points will be removed.

```
#Identifying the outliers and prviding a count
outliers = az_data.price[(az_data.price <= lower_bound) |
(az_data.price >= upper_bound)]
print('The following are the outliers in the
boxplot:{}'.format(outliers))
outliers.count() #the count is 38

#calculating the amount of data lost if outliers are removed
# 562-38 = 532
# 532/562*100 = 94.7% of data would remain
# 100-94.7 = 5.3% of data lost

#----------Remove identified outliers from 'price'

#Sort the 'Price' data to aid in outlier removal
az_data = az_data.sort_values(by=['price'], ascending= False)

#removes to values <= $1 million in the price column
az_data = az_data.iloc[38:]

#Repeat histogram and boxplots
#histogram of Price
x = az_data["price"]
plt.title("Housing Price - Histogram")
plt.xlabel("Price")
plt.hist(x, bins='auto', color = "skyblue", ec="black")
current_valuesx = plt.gca().get_xticks()
plt.gca().set_xticklabels(['{:.0f}'.format(x) for x in
current_valuesx])
```

```
plt.show()

# Creating boxplot
plt.title("Price - Boxplot")
plt.ylabel("Price")
plt.boxplot(az_data.price, patch_artist=True,
boxprops=dict(facecolor='green', color='black'),
capprops=dict(color='red'),
whiskerprops=dict(color='black'),
flierprops=dict(color='blue', markeredgecolor='black'),
medianprops=dict(color='yellow'))
current_values = plt.gca().get_yticks()
plt.gca().set_yticklabels(['{:.0f}'.format(x) for x in
current_values])
plt.show()

#####-----NOTE: the boxplot still shows some outliers due to the new
range
#####-----      of the data no further removal is needed for 'price'
```

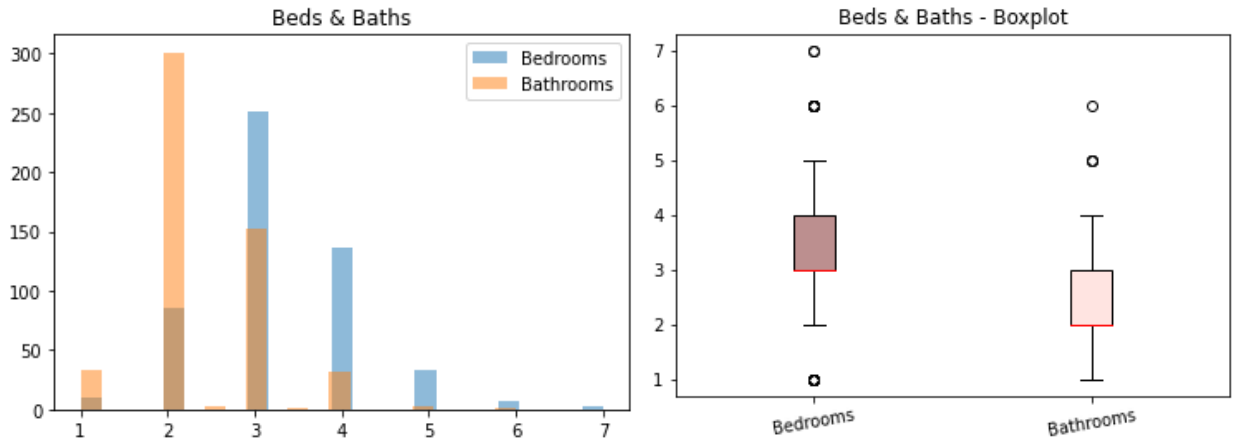




The extreme outliers in the 'Price' column have been removed. While the boxplot continues to show outliers, it is only because the range of the data has changed and therefore the upper bound also gets changed resulting in new data points being labeled as outliers. We do not need to remove them.

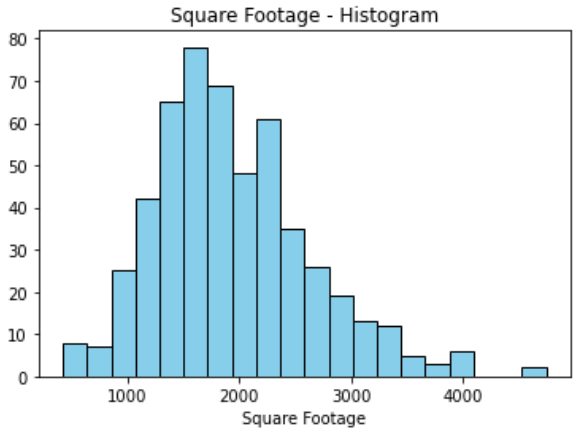
```
#----------Checking 'bedroom' 'bath' and 'sqrt' for outliers

# Bedroom and Bathrooms
#histogram
x = az_data["beds"]
y = az_data["baths"]
plt.title("Beds & Baths")
```

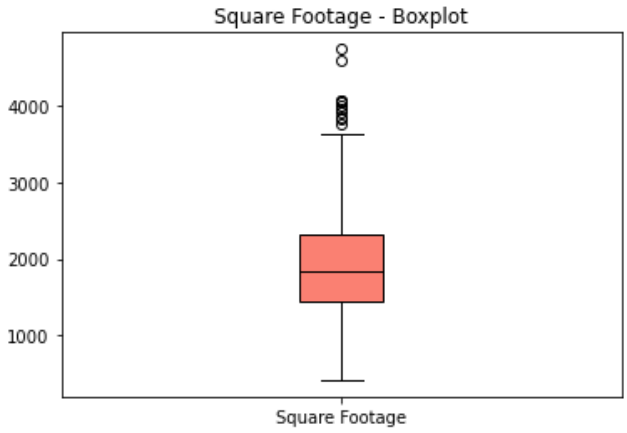
```
plt.hist(x, bins='auto',  alpha=0.5, label='Bedrooms')
plt.hist(y, bins='auto',  alpha=0.5, label='Bathrooms')
plt.legend(loc='upper right')
plt.show()

# Boxplot
azbb = az_data[["beds","baths"]]
plt.title("Beds & Baths - Boxplot")
box = plt.boxplot(azbb, patch_artist=True, meanline=True,
medianprops=dict(color='red'))
plt.xticks([1, 2], ["Bedrooms", "Bathrooms"], rotation=10)

colors = ['rosybrown', 'mistyrose']

for patch, color in zip(box['boxes'], colors):
patch.set_facecolor(color)
plt.show()
```




```
# Square Feet
#histogram
x = az_data["sqft"]
plt.title("Square Footage - Histogram")
plt.xlabel("Square Footage")
plt.hist(x, bins='auto', color = "skyblue", ec="black")
plt.show()

# Creating boxplot
plt.title("Square Footage - Boxplot")
plt.boxplot(az_data.sqft, patch_artist=True,
boxprops=dict(facecolor='salmon', color='black'),
medianprops=dict(color='black'))
plt.xticks([1], ["Square Footage"], rotation=0)
plt.show()
```

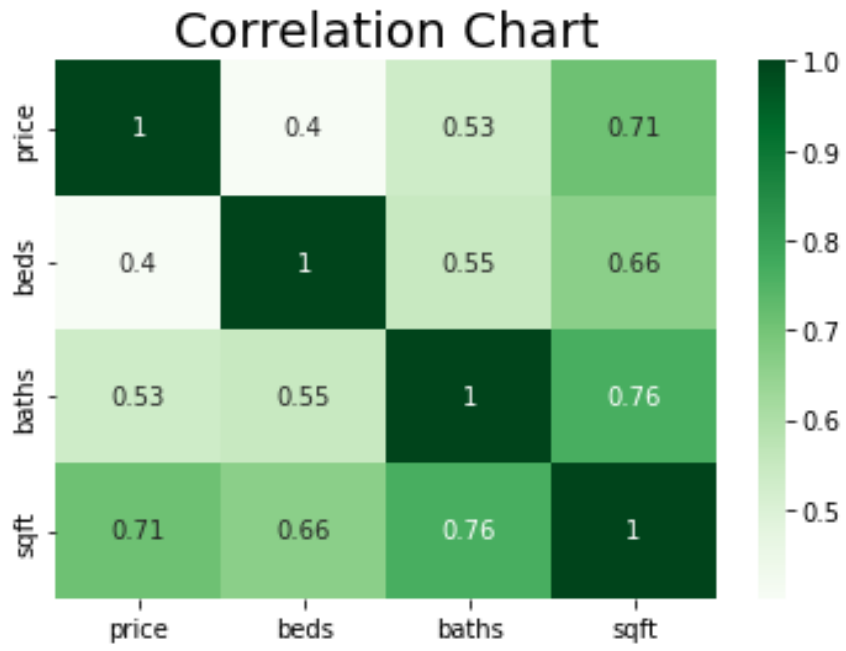
```
#There are outliers, but I can't think of a reason to remove them
```





We look at the other numerical columns for outliers and while there are some, their presence should not interfere significantly. Next, I'd like to see how the variables correlate (if at all).

```
#Running a correlation on the numerical
az_data2 = az_data[["price","beds", "baths", "sqft"]]
corrAz = az_data2.corr()
corrAz
sns.heatmap(corrAz, annot = True, cmap="Greens")
plt.title("Correlation Chart", fontsize =20)
plt.show()
```



It is interesting to see that square footage and bathrooms have the highest correlation. I expected that the square footage and price would be the highest. In order to answer question #1, I needed to prepare the data to find the result.

```
#Looking at averages
cols = ['price','beds','baths','sqft']
cols2 = ['city', 'price','beds','baths','sqft']
azdata = az_data[cols].mean()
cpmean = az_data[cols2].groupby('city').mean()
cpmean = cpmean.sort_values(by=['price'], ascending= False)

#Setting up to answer Q1
print('Appearances of each city within dataset')
print(az_data['city'].value_counts())

#More setting up to answer Q1
cityCounts = az_data['city'].value_counts()
cityCounts = cityCounts.to_frame()
cityCounts2 = cityCounts['city'].value_counts()
cityCounts2 = cityCounts2.sort_index()

cityCounts = cityCounts.reset_index()
cityCounts = cityCounts.rename(columns={'city' : 'count'})
cityCounts = cityCounts.rename(columns={'index' : 'city'})
```

```
Appearances of each city within dataset
Phoenix                60
Tucson                 33
Mesa                   30
Scottsdale             17
Gilbert                17
                       ..
Young                   1
Kirkland                1
Eloy                    1
White Mountain Lake     1
Holbrook                1
Name: city, Length: 105, dtype: int64
```

```
#-------Answers to questions-------

#Answering Q1
print("1. What are the top 5 areas with the most home sales?")
print(cityCounts.head(5))
```

```
1. What are the top 5 areas with the most home sales?
         city  count
0     Phoenix     60
1      Tucson     33
2        Mesa     30
3  Scottsdale     17
4     Gilbert     17
```

```
#Answering Q2
az_data3=az_data.drop(columns =['beds', 'baths' ,'sqft', 'zipcode' ])
b=az_data3.groupby(['city']).mean()
b=b.sort_values(by =['price'], ascending=False)
b=b.head(1)
b=b.astype({"price": 'int64' })
print()
print()
print("2. What 'Local Area' has the highest average price?")
print(b)
```

```
2. What 'Local Area' has the highest average price?
                  price
city
Mingus Mountain  823999
```

```
#Answering Q3
az_data4=az_data.drop(columns =['beds', 'baths' ,'price', 'zipcode' ])
c=az_data4.groupby(['city']).mean()
c=c.reset_index()
c=c.rename(columns ={'index' : 'city' })
c=c[c.city = ='Phoenix' ]
c=c.astype({"sqft": 'int64' })
#print(c[c.sqft= ='Phoenix' ])
#c=c.sort_values(by =['sqft'], ascending=False)
#c=c.head(1)
#b=b.astype({"price": 'int64' })
print()
print()
print("3. What is the average square footage of the most popular
local?")
print(c)
```

```
3. What is the average square footage of the most popular local?
       city  sqft
60  Phoenix  1837
```