Photo credit: Pixabay

Practical Statistics & Visualization With Python & Plotly

How to use Python and Plotly for statistical visualization, inference, and modeling

Susan Li Follow
May 14, 2019 · 9 min read

One day last week, I was googling "***statistics with Python***", the results were somewhat unfruitful. Most literature, tutorials and articles focus on statistics with ***R***, because ***R*** is a language dedicated to statistics and has more statistical analysis features than Python.

In two excellent statistics books, "***Practical Statistics for Data Scientists***" and "***An Introduction to Statistical Learning***", the statistical concepts were all implemented in ***R***.

Data science is a fusion of multiple disciplines, including statistics, computer science, information technology, and domain-specific fields. And we use powerful, open-source

Python tools daily to manipulate, analyze, and visualize datasets.

And I would certainly recommend anyone interested in becoming a Data Scientist or Machine Learning Engineer to develop a deep understanding and practice constantly on statistical learning theories.

This prompts me to write a post for the subject. And I will use one dataset to review as many statistics concepts as I can and lets get started!

The Data

The data is the house prices data set that can be found here.

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from plotly.offline import init_notebook_mode, iplot
import plotly.figure_factory as ff
import cufflinks
cufflinks.go_offline()
cufflinks.set_config_file(world_readable=True, theme='pearl')
import plotly.graph_objs as go
import plotly.plotly as py
import plotly
from plotly import tools
plotly.tools.set_credentials_file(username='XXX', api_key='XXX')
init_notebook_mode(connected=True)
pd.set_option('display.max_columns', 100)

df = pd.read_csv('house_train.csv')
df.drop('Id', axis=1, inplace=True)
df.head()
```

	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	LandContour	Utilities	LotConfig	LandSlope	Neighborhood	Condition1	Condition2
0	60	RL	65.0	8450	Pave	NaN	Reg	Lvl	AllPub	Inside	Gtl	CollgCr	Norm	Norm
1	20	RL	80.0	9600	Pave	NaN	Reg	Lvl	AllPub	FR2	Gtl	Veenker	Feedr	Norm
2	60	RL	68.0	11250	Pave	NaN	IR1	Lvl	AllPub	Inside	Gtl	CollgCr	Norm	Norm
3	70	RL	60.0	9550	Pave	NaN	IR1	Lvl	AllPub	Corner	Gtl	Crawfor	Norm	Norm
4	60	RL	84.0	14260	Pave	NaN	IR1	Lvl	AllPub	FR2	Gtl	NoRidge	Norm	Norm

Table 1

Univariate Data Analysis

Univariate analysis is perhaps the simplest form of statistical analysis, and the key fact is that only one variable is involved.

Describing Data

Statistical summary for numeric data include things like the mean, min, and max of the data, can be useful to get a feel for how large some of the variables are and what variables may be the most important.

```
df.describe().T
```

	count	mean	std	min	25%	50%	75%	max
MSSubClass	1460.0	56.897260	42.300571	20.0	20.00	50.0	70.00	190.0
LotFrontage	1201.0	70.049958	24.284752	21.0	59.00	69.0	80.00	313.0
LotArea	1460.0	10516.828082	9981.264932	1300.0	7553.50	9478.5	11601.50	215245.0
OverallQual	1460.0	6.099315	1.382997	1.0	5.00	6.0	7.00	10.0
OverallCond	1460.0	5.575342	1.112799	1.0	5.00	5.0	6.00	9.0
YearBuilt	1460.0	1971.267808	30.202904	1872.0	1954.00	1973.0	2000.00	2010.0
YearRemodAdd	1460.0	1984.865753	20.645407	1950.0	1967.00	1994.0	2004.00	2010.0
MasVnrArea	1452.0	103.685262	181.066207	0.0	0.00	0.0	166.00	1600.0
BsmtFinSF1	1460.0	443.639726	456.098091	0.0	0.00	383.5	712.25	5644.0
BsmtFinSF2	1460.0	46.549315	161.319273	0.0	0.00	0.0	0.00	1474.0
BsmtUnfSF	1460.0	567.240411	441.866955	0.0	223.00	477.5	808.00	2336.0
TotalBsmtSF	1460.0	1057.429452	438.705324	0.0	795.75	991.5	1298.25	6110.0
1stFlrSF	1460.0	1162.626712	386.587738	334.0	882.00	1087.0	1391.25	4692.0
2ndFlrSF	1460.0	346.992466	436.528436	0.0	0.00	0.0	728.00	2065.0
LowQualFinSF	1460.0	5.844521	48.623081	0.0	0.00	0.0	0.00	572.0
GrLivArea	1460.0	1515.463699	525.480383	334.0	1129.50	1464.0	1776.75	5642.0
BsmtFullBath	1460.0	0.425342	0.518911	0.0	0.00	0.0	1.00	3.0
BsmtHalfBath	1460.0	0.057534	0.238753	0.0	0.00	0.0	0.00	2.0
FullBath	1460.0	1.565068	0.550916	0.0	1.00	2.0	2.00	3.0
HalfBath	1460.0	0.382877	0.502885	0.0	0.00	0.0	1.00	2.0
BedroomAbvGr	1460.0	2.866438	0.815778	0.0	2.00	3.0	3.00	8.0
KitchenAbvGr	1460.0	1.046575	0.220338	0.0	1.00	1.0	1.00	3.0
TotRmsAbvGrd	1460.0	6.517808	1.625393	2.0	5.00	6.0	7.00	14.0
Fireplaces	1460.0	0.613014	0.644666	0.0	0.00	1.0	1.00	3.0
GarageYrBlt	1379.0	1978.506164	24.689725	1900.0	1961.00	1980.0	2002.00	2010.0
GarageCars	1460.0	1.767123	0.747315	0.0	1.00	2.0	2.00	4.0
GarageArea	1460.0	472.980137	213.804841	0.0	334.50	480.0	576.00	1418.0
WoodDeckSF	1460.0	94.244521	125.338794	0.0	0.00	0.0	168.00	857.0
OpenPorchSF	1460.0	46.660274	66.256028	0.0	0.00	25.0	68.00	547.0
EnclosedPorch	1460.0	21.954110	61.119149	0.0	0.00	0.0	0.00	552.0
3SsnPorch	1460.0	3.409589	29.317331	0.0	0.00	0.0	0.00	508.0
ScreenPorch	1460.0	15.060959	55.757415	0.0	0.00	0.0	0.00	480.0
PoolArea	1460.0	2.758904	40.177307	0.0	0.00	0.0	0.00	738.0
MiscVal	1460.0	43.489041	496.123024	0.0	0.00	0.0	0.00	15500.0

MoSold	1460.0	6.321918	2.703626	1.0	5.00	6.0	8.00	12.0
YrSold	1460.0	2007.815753	1.328095	2006.0	2007.00	2008.0	2009.00	2010.0
SalePrice	1460.0	180921.195890	79442.502883	34900.0	129975.00	163000.0	214000.00	755000.0
HouseAge	1460.0	47.732192	30.202904	9.0	19.00	48.0	65.00	147.0

Table 2

Statistical summary for categorical or string variables will show “count”, “unique”, “top”, and “freq”.

```
table_cat = ff.create_table(df.describe(include=['O']).T, index=True,
index_title='Categorical columns')
iplot(table_cat)
```

Categorical columns	count	unique	top	freq
MSZoning	1460	5	RL	1151
Street	1460	2	Pave	1454
Alley	91	2	Grvl	50
LotShape	1460	4	Reg	925
LandContour	1460	4	Lvl	1311
Utilities	1460	2	AllPub	1459
LotConfig	1460	5	Inside	1052
LandSlope	1460	3	Gtl	1382
Neighborhood	1460	25	NAmes	225
Condition1	1460	9	Norm	1260
Condition2	1460	8	Norm	1445
BldgType	1460	5	1Fam	1220
HouseStyle	1460	8	1Story	726
RoofStyle	1460	6	Gable	1141
RoofMatl	1460	8	CompShg	1434
Exterior1st	1460	15	VinylSd	515
Exterior2nd	1460	16	VinylSd	504
MasVnrType	1452	4	None	864
ExterQual	1460	4	TA	906
ExterCond	1460	5	TA	1282
Foundation	1460	6	PConc	647
BsmtQual	1423	4	TA	649
BsmtCond	1423	4	TA	1311
BsmtExposure	1422	4	No	953
BsmtFinType1	1423	6	Unf	430
BsmtFinType2	1422	6	Unf	1256
Heating	1460	6	GasA	1428
HeatingQC	1460	5	Ex	741
CentralAir	1460	2	Y	1365
Electrical	1459	5	SBrkr	1334
KitchenQual	1460	4	TA	735
Functional	1460	7	Typ	1360
FireplaceQu	770	5	Gd	380
GarageType	1379	6	Attchd	870
GarageFinish	1379	3	Unf	605
GarageQual	1379	5	TA	1311
GarageCond	1379	5	TA	1326
PavedDrive	1460	3	Y	1340
PoolQC	7	3	Gd	3
Fence	281	4	MnPrv	157
MiscFeature	54	4	Shed	49
SaleType	1460	9	WD	1267
SaleCondition	1460	6	Normal	1198

Table 3

Histogram

Plot a histogram of SalePrice of all the houses in the data.

```
df['SalePrice'].iplot(
    kind='hist',
    bins=100,
    xTitle='price',
    linecolor='black',
    yTitle='count',
    title='Histogram of Sale Price')
```

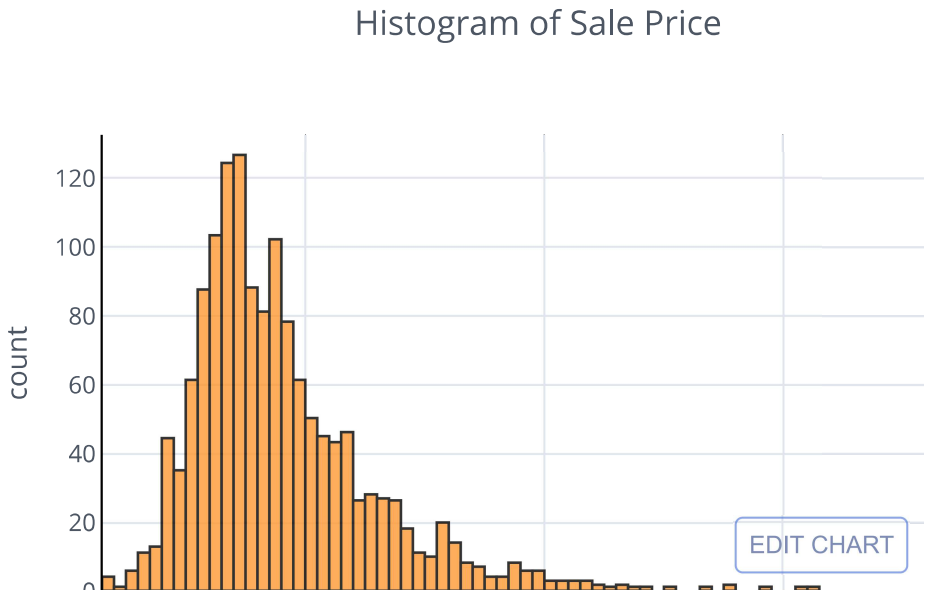


Figure 1

Boxplot

Plot a boxplot of SalePrice of all the houses in the data. Boxplots do not show the shape of the distribution, but they can give us a better idea about the center and spread of the distribution as well as any potential outliers that may exist. Boxplots and Histograms often complement each other and help us understand more about the data.

```
df['SalePrice'].iplot(kind='box', title='Box plot of SalePrice')
```

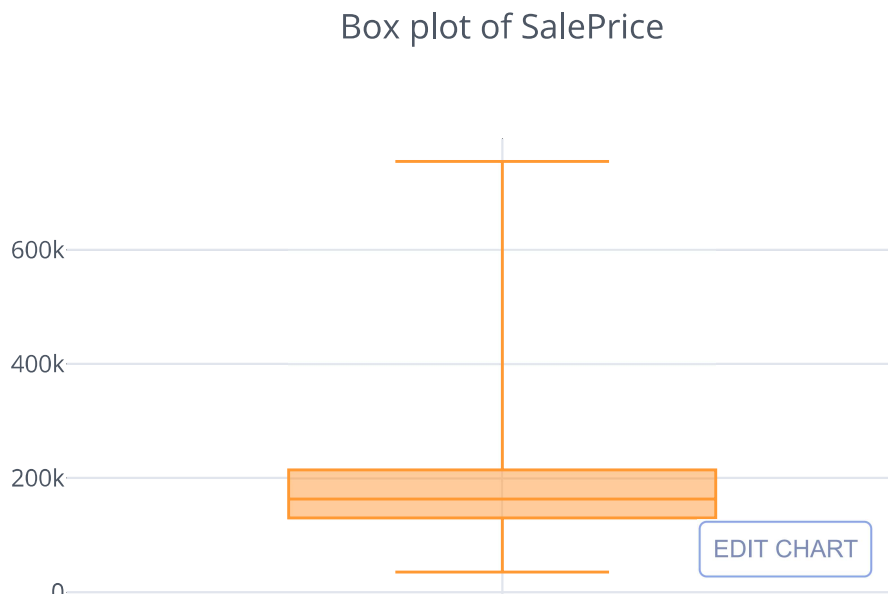


Figure 2

Histograms and Boxplots by Groups

Plotting by groups, we can see how a variable changes in response to another. For example, if there is a difference between house SalePrice with or with no central air conditioning. Or if house SalePrice varies according to the size of the garage, and so on.

Boxplot and histogram of house sale price grouped by with or with no air conditioning

```
trace0 = go.Box(
    y=df.loc[df['CentralAir'] == 'Y']['SalePrice'],
    name = 'With air conditioning',
    marker = dict(
        color = 'rgb(214, 12, 140)',
    )
)
trace1 = go.Box(
    y=df.loc[df['CentralAir'] == 'N']['SalePrice'],
    name = 'no air conditioning',
    marker = dict(
```

```
        color = 'rgb(0, 128, 128)',
    )
)
data = [trace0, trace1]
layout = go.Layout(
    title = "Boxplot of Sale Price by air conditioning"
)

fig = go.Figure(data=data,layout=layout)
py.iplot(fig)
```

boxplot_aircon.py hosted with ♡ by GitHub view raw

boxplot.aircon.py

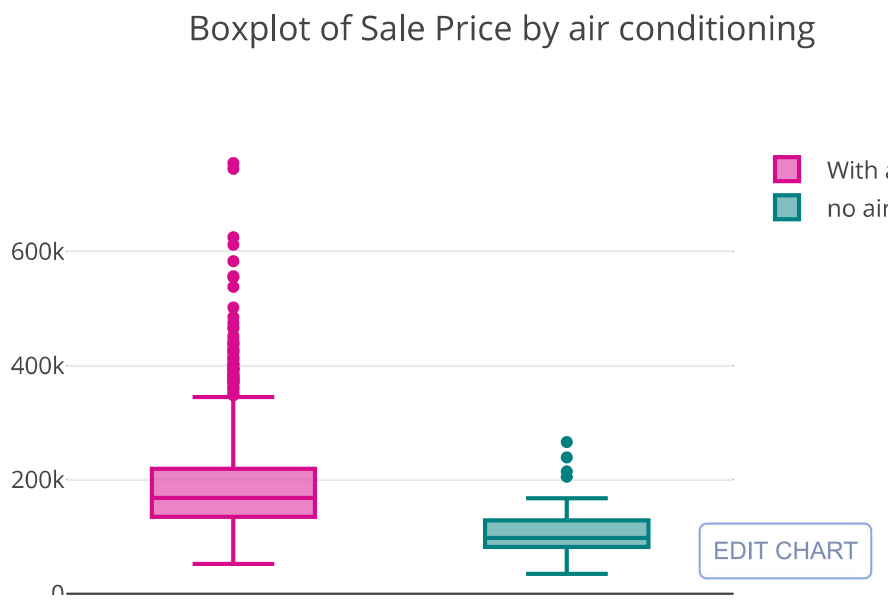


Figure 3

```
trace0 = go.Histogram(
    x=df.loc[df['CentralAir'] == 'Y']['SalePrice'], name='With Central air conditioning',
    opacity=0.75
)
trace1 = go.Histogram(
```

```
    x=df.loc[df['CentralAir'] == 'N']['SalePrice'], name='No Central air conditioning',
    opacity=0.75
)

data = [trace0, trace1]
layout = go.Layout(barmode='overlay', title='Histogram of House Sale Price for both with and wit
fig = go.Figure(data=data, layout=layout)

py.iplot(fig)
```

histogram_aircon.py hosted with ♡ by GitHub view raw

histogram_aircon.py

stogram of House Sale Price for both with and with no Central

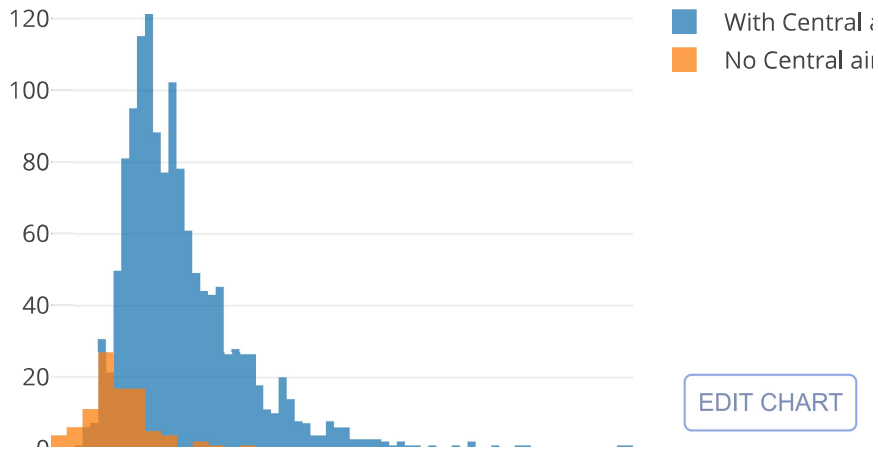


Figure 4

```
df.groupby('CentralAir')['SalePrice'].describe()
```

	count	mean	std	min	25%	50%	75%	max
CentralAir								

CentralAir

N	95.0	105264.073684	40671.273961	34900.0	82000.0	98000.0	128500.0	265979.0
Y	1365.0	186186.709890	78805.206820	52000.0	134800.0	168000.0	219210.0	755000.0

Table 4

It is obviously that the mean and median sale price for houses with no air conditioning are much lower than the houses with air conditioning.

Boxplot and histogram of house sale price grouped by garage size

```
trace0 = go.Box(
    y=df.loc[df['GarageCars'] == 0]['SalePrice'],
    name = 'no garage',
    marker = dict(
        color = 'rgb(214, 12, 140)',
    )
)
trace1 = go.Box(
    y=df.loc[df['GarageCars'] == 1]['SalePrice'],
    name = '1-car garage',
    marker = dict(
        color = 'rgb(0, 128, 128)',
    )
)
trace2 = go.Box(
    y=df.loc[df['GarageCars'] == 2]['SalePrice'],
    name = '2-cars garage',
    marker = dict(
        color = 'rgb(12, 102, 14)',
    )
)
trace3 = go.Box(
    y=df.loc[df['GarageCars'] == 3]['SalePrice'],
    name = '3-cars garage',
    marker = dict(
        color = 'rgb(10, 0, 100)',
    )
)
trace4 = go.Box(
    y=df.loc[df['GarageCars'] == 4]['SalePrice'],
    name = '4-cars garage',
    marker = dict(
```

```
        color = 'rgb(100, 0, 10)',
    )
)
data = [trace0, trace1, trace2, trace3, trace4]
layout = go.Layout(
    title = "Boxplot of Sale Price by garage size"
)

fig = go.Figure(data=data,layout=layout)
py.iplot(fig)
```

boxplot_garage.py hosted with ♡ by GitHub view raw

boxplot_garage.py

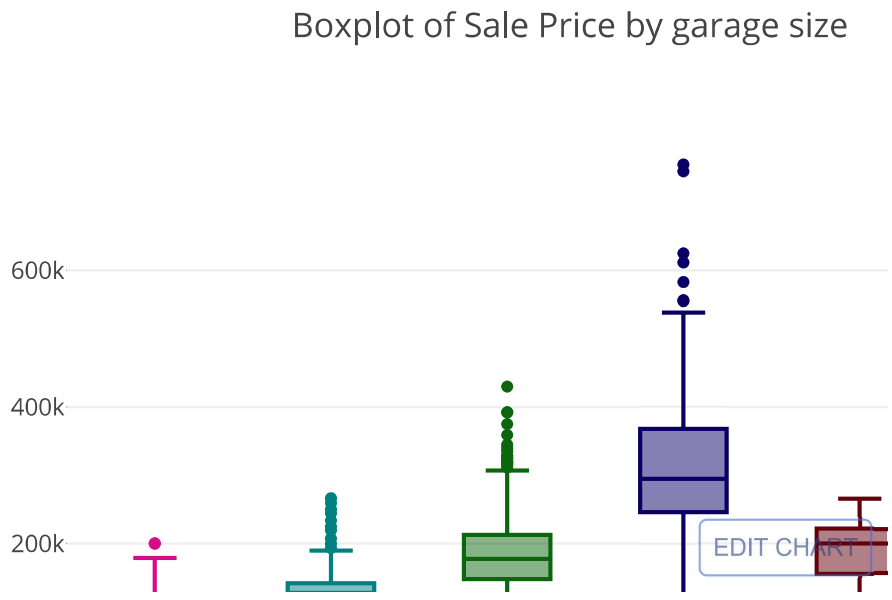


Figure 5

The larger the garage, the higher house median price, this works until we reach 3-cars garage. Apparently, the houses with 3-cars garages have the highest median price, even higher than the houses with 4-cars garage.

Histogram of house sale price with no garage

```
df.loc[df['GarageCars'] == 0]['SalePrice'].iplot(
    kind='hist',
    bins=50,
    xTitle='price',
    linecolor='black',
    yTitle='count',
    title='Histogram of Sale Price of houses with no garage')
```

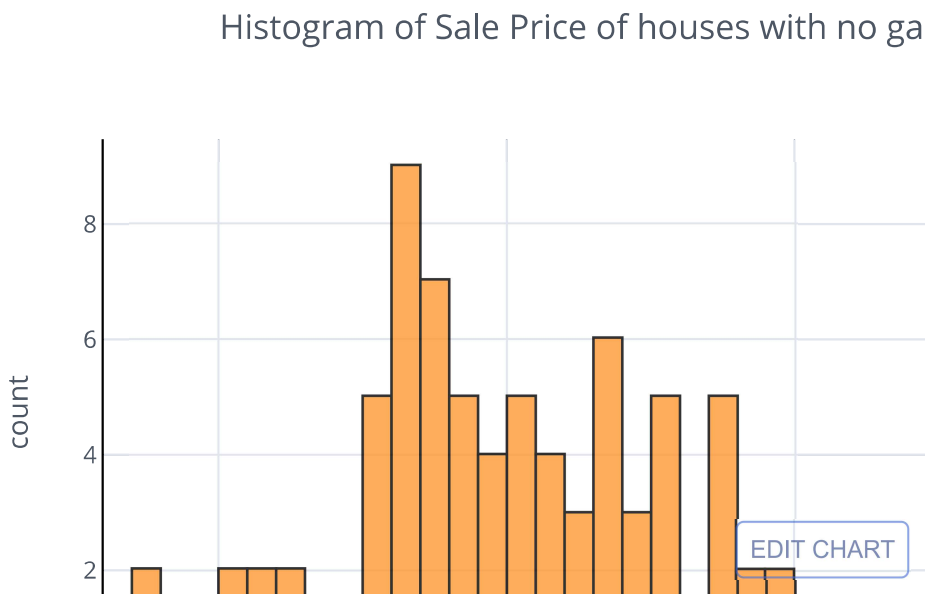


Figure 6

Histogram of house sale price with 1-car garage

```
df.loc[df['GarageCars'] == 1]['SalePrice'].iplot(
    kind='hist',
    bins=50,
    xTitle='price',
    linecolor='black',
```

```
    yTitle='count',
    title='Histogram of Sale Price of houses with 1-car garage')
```

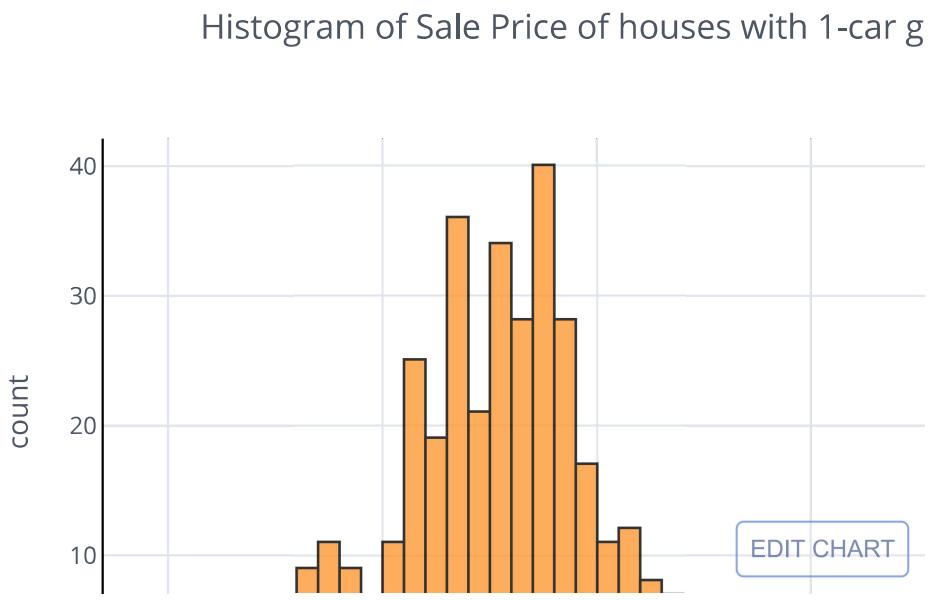


Figure 7

Histogram of house sale price with 2-car garage

```
df.loc[df['GarageCars'] == 2]['SalePrice'].iplot(
    kind='hist',
    bins=100,
    xTitle='price',
    linecolor='black',
    yTitle='count',
    title='Histogram of Sale Price of houses with 2-car garage')
```

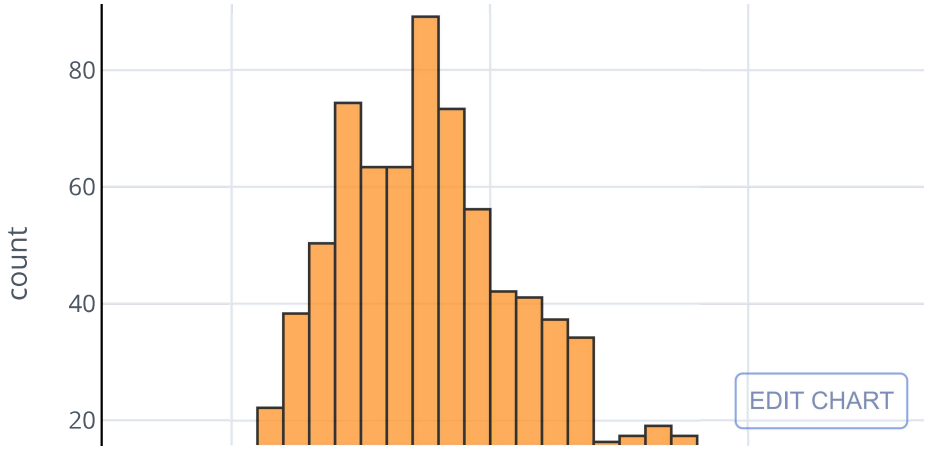


Figure 8

Histogram of house sale price with 3-car garage

```
df.loc[df['GarageCars'] == 3]['SalePrice'].iplot(
    kind='hist',
    bins=50,
    xTitle='price',
    linecolor='black',
    yTitle='count',
    title='Histogram of Sale Price of houses with 3-car garage')
```

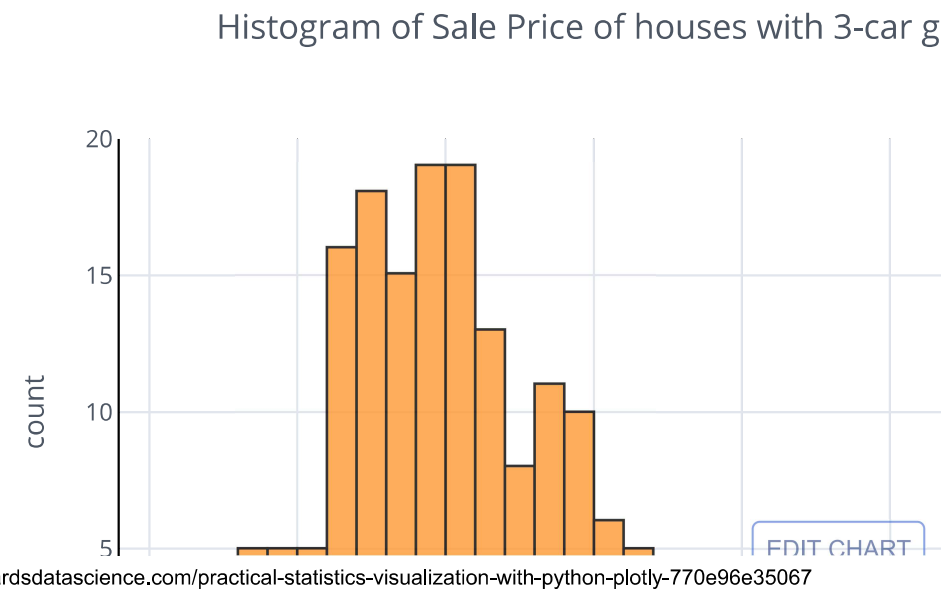


Figure 9

Histogram of house sale price with 4-car garage

```
df.loc[df['GarageCars'] == 4]['SalePrice'].iplot(
    kind='hist',
    bins=10,
    xTitle='price',
    linecolor='black',
    yTitle='count',
    title='Histogram of Sale Price of houses with 4-car garage')
```

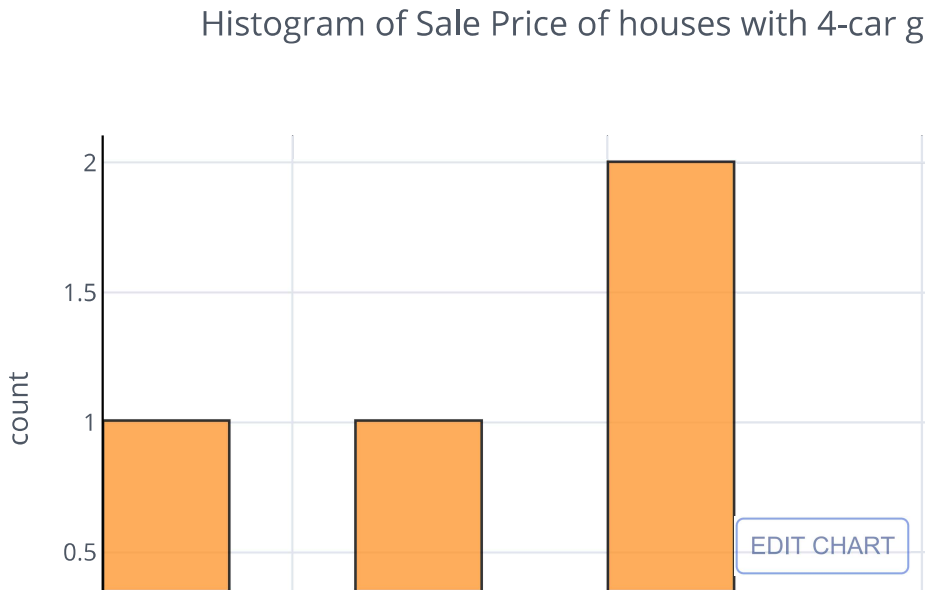


Figure 10

Frequency Table

Frequency tells us how often something happened. Frequency tables give us a snapshot of the data to allow us to find patterns.

Overall quality frequency table

```
x = df.OverallQual.value_counts()
x/x.sum()
```

```
5     0.271918
6     0.256164
7     0.218493
8     0.115068
4     0.079452
9     0.029452
3     0.013699
10    0.012329
2     0.002055
1     0.001370
Name: OverallQual, dtype: float64
```

Table 5

Garage size frequency table

```
x = df.GarageCars.value_counts()
x/x.sum()
```

```
2    0.564384
1    0.252740
3    0.123973
0    0.055479
4    0.003425
Name: GarageCars, dtype: float64
```

Table 6

Central air conditioning frequency table

```
x = df.CentralAir.value_counts()
x/x.sum()
```

```
Y    0.934932
N    0.065068
Name: CentralAir, dtype: float64
```

Table 7

Numerical Summaries

A quick way to get a set of numerical summaries for a quantitative variable is to use the describe method.

```
df.SalePrice.describe()
```

```
count      1460.000000
mean     180921.195890
std       79442.502883
min       34900.000000
25%      129975.000000
50%      163000.000000
75%      214000.000000
max      755000.000000
Name: SalePrice, dtype: float64
```

Table 8

We can also calculate individual summary statistics of SalePrice.

```
print("The mean of sale price, - Pandas method: ",
df.SalePrice.mean())
print("The mean of sale price, - Numpy function: ",
np.mean(df.SalePrice))
print("The median sale price: ", df.SalePrice.median())
print("50th percentile, same as the median: ",
np.percentile(df.SalePrice, 50))
print("75th percentile: ", np.percentile(df.SalePrice, 75))
print("Pandas method for quantiles, equivalent to 75th percentile: ",
df.SalePrice.quantile(0.75))
```

```
The mean of sale price, - Pandas method:  180921.19589041095
The mean of sale price, - Numpy function:  180921.19589041095
The median sale price:  163000.0
50th percentile, same as the median:  163000.0
75th percentile:  214000.0
Pandas method for quantiles, equivalent to 75th percentile:  214000.0
```

Calculate the proportion of the houses with sale price between 25th percentile (129975) and 75th percentile (214000).

```
print('The proportion of the houses with prices between 25th
percentile and 75th percentile: ', np.mean((df.SalePrice >= 129975) &
(df.SalePrice <= 214000)))
```

```
The proportion of the houses with prices between 25th percentile and 75th percentile:  0.5020547945205479
```

Calculate the proportion of the houses with total square feet of basement area between 25th percentile (795.75) and 75th percentile (1298.25).

```
print('The proportion of house with total square feet of basement
area between 25th percentile and 75th percentile: ',
np.mean((df.TotalBsmtSF >= 795.75) & (df.TotalBsmtSF <= 1298.25)))
```

```
The proportion of house with total square feet of basement area between 25th percentile and 75th percentile:  0.5
```

Lastly, we calculate the proportion of the houses based on either conditions. Since some houses are under both criteria, the proportion below is less than the sum of the two proportions calculated above.

```
a = (df.SalePrice >= 129975) & (df.SalePrice <= 214000)
b = (df.TotalBsmtSF >= 795.75) & (df.TotalBsmtSF <= 1298.25)
print(np.mean(a | b))
```

```
0.7143835616438357
```

Calculate sale price IQR for houses with no air conditioning.

```
q75, q25 = np.percentile(df.loc[df['CentralAir']=='N']['SalePrice'],
[75,25])
iqr = q75 - q25
print('Sale price IQR for houses with no air conditioning: ', iqr)
```

```
Sale price IQR for houses with no air conditioning:  46500.0
```

Calculate sale price IQR for houses with air conditioning.

```
q75, q25 = np.percentile(df.loc[df['CentralAir']=='Y']['SalePrice'],
[75,25])
iqr = q75 - q25
print('Sale price IQR for houses with air conditioning: ', iqr)
```

```
Sale price IQR for houses with air conditioning:  84410.0
```

Stratification

Another way to get more information out of a dataset is to divide it into smaller, more uniform subsets, and analyze each of these "strata" on its own. We will create a new HouseAge column, then partition the data into HouseAge strata, and construct side-by-side boxplots of the sale price within each stratum.

```
df['HouseAge'] = 2019 - df['YearBuilt']
df["AgeGrp"] = pd.cut(df.HouseAge, [9, 20, 40, 60, 80, 100, 147]) #
Create age strata based on these cut points
plt.figure(figsize=(12, 5))
sns.boxplot(x="AgeGrp", y="SalePrice", data=df);
```

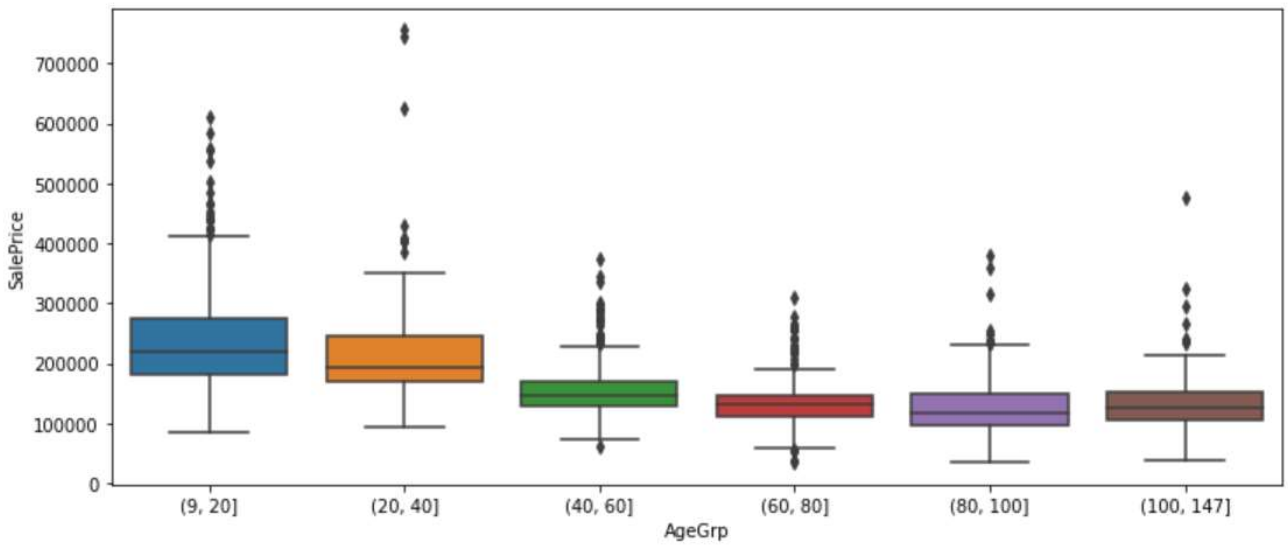


Figure 11

The older the house, the lower the median price, that is, house price tends to decrease with age, until it reaches 100 years old. The median price of over 100 year old houses is higher than the median price of houses age between 80 and 100 years.

```
plt.figure(figsize=(12, 5))
sns.boxplot(x="AgeGrp", y="SalePrice", hue="CentralAir", data=df)
plt.show();
```

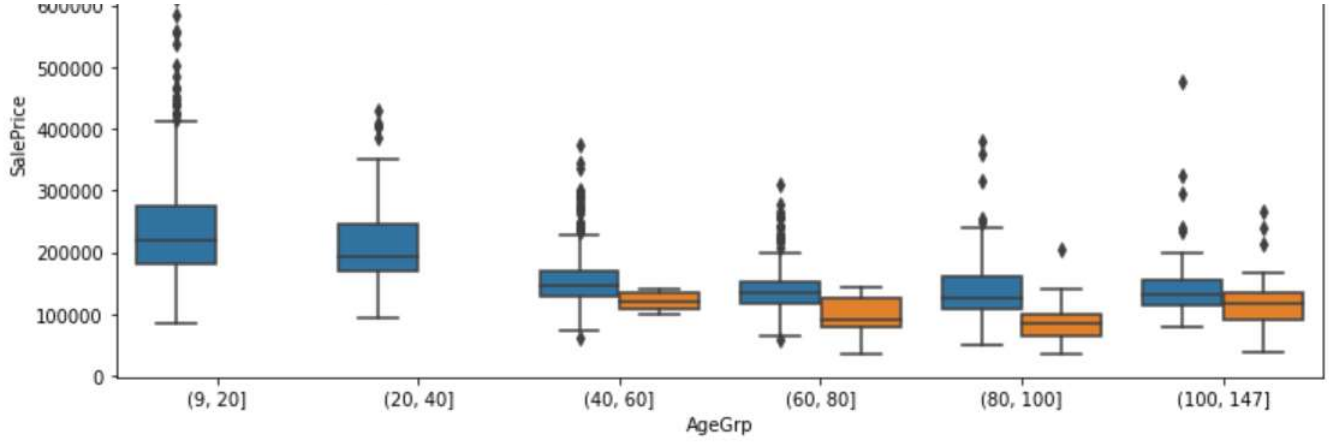


Figure 12

We have learned earlier that house price tends to differ between with and with no air conditioning. From above graph, we also find out that recent houses (9–40 years old) are all equipped with air conditioning.

```
plt.figure(figsize=(12, 5))
sns.boxplot(x="CentralAir", y="SalePrice", hue="AgeGrp", data=df)
plt.show();
```

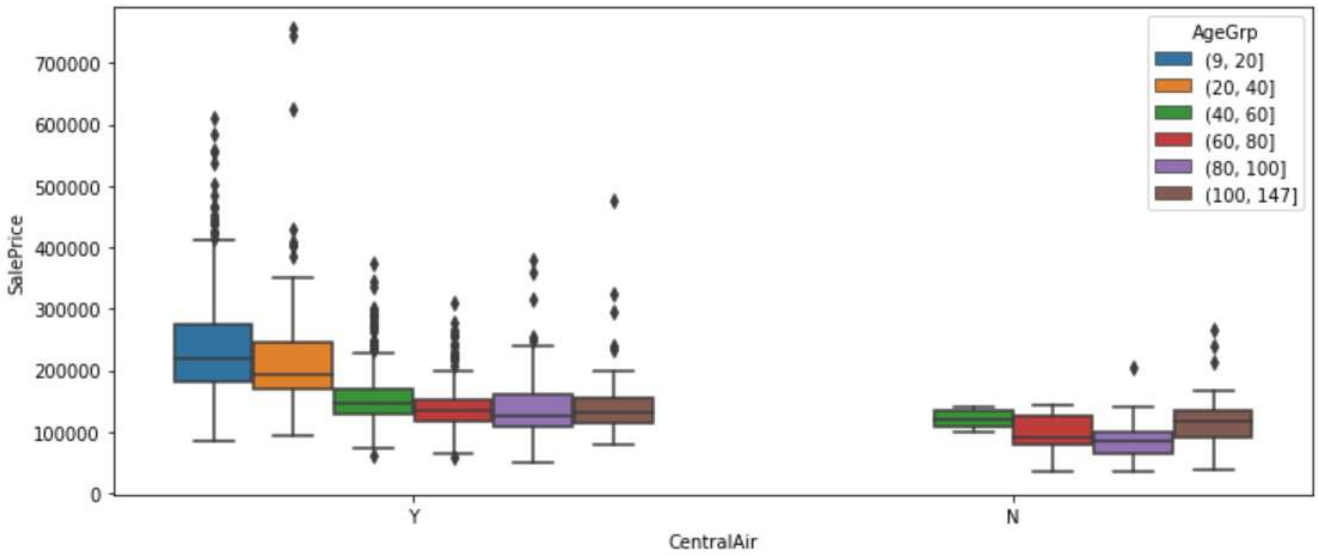


Figure 13

We now group first by air conditioning, and then within air conditioning group by age bands. Each approach highlights a different aspect of the data.

We can also stratify jointly by House age and air conditioning to explore how building type varies by both of these factors simultaneously.

```
df1 = df.groupby(["AgeGrp", "CentralAir"])["BldgType"]
df1 = df1.value_counts()
df1 = df1.unstack()
df1 = df1.apply(lambda x: x/x.sum(), axis=1)
print(df1.to_string(float_format="%.3f"))
```

BldgType		1Fam	2fmCon	Duplex	Twnhs	TwnhsE
AgeGrp	CentralAir					
(9, 20]	Y	0.782	NaN	NaN	0.046	0.172
(20, 40]	Y	0.827	NaN	0.053	0.010	0.111
(40, 60]	N	0.167	0.167	0.667	NaN	NaN
	Y	0.813	0.008	0.067	0.059	0.053
(60, 80]	N	0.556	0.148	0.296	NaN	NaN
	Y	0.955	0.030	0.015	NaN	NaN
(80, 100]	N	0.926	0.074	NaN	NaN	NaN
	Y	0.992	0.008	NaN	NaN	NaN
(100, 147]	N	0.771	0.229	NaN	NaN	NaN
	Y	0.860	0.120	0.020	NaN	NaN

Table 9

For all house age groups, vast majority type of dwelling in the data is 1Fam. The older the house, the more likely to have no air conditioning. However, for a 1Fam house over 100 years old, it is a little more likely to have air conditioning than not. There were neither very new nor very old duplex house types. For a 40–60 year old duplex house, it is more likely to have no air conditioning.

Multivariate Analysis

Multivariate analysis is based on the statistical principle of multivariate statistics, which involves observation and analysis of more than one statistical outcome variable at a time.

Scatter plot

A scatter plot is a very common and easily-understood visualization of quantitative bivariate data. Below we make a scatter plot of Sale Price against Above ground living area square feet. it is apparently a linear relationship.

```
df.iplot(
    x='GrLivArea',
    y='SalePrice',
    xTitle='Above ground living area square feet',
    yTitle='Sale price',
    mode='markers',
    title='Sale Price vs Above ground living area square feet')
```

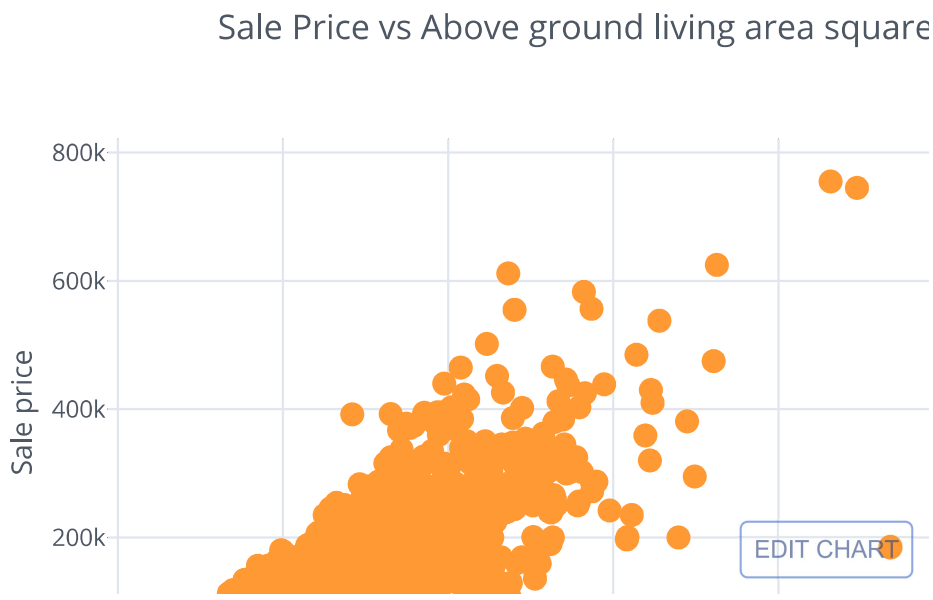


Figure 14

2D Density Joint plot

The following two plot margins show the densities for the Sale Price and Above ground living area separately, while the plot in the center shows their density jointly.

```
1  trace1 = go.Scatter(
```

```
    x=df['GrLivArea'], y=df['SalePrice'], mode='markers', name='points',
    marker=dict(color='rgb(102,0,0)', size=2, opacity=0.4)
)
trace2 = go.Histogram2dContour(
    x=df['GrLivArea'], y=df['SalePrice'], name='density', ncontours=20,
    colorscale='Hot', reversescale=True, showscale=False
)
trace3 = go.Histogram(
    x=df['GrLivArea'], name='Ground Living area density',
    marker=dict(color='rgb(102,0,0)'),
    yaxis='y2'
)
trace4 = go.Histogram(
    y=df['SalePrice'], name='Sale Price density', marker=dict(color='rgb(102,0,0)'),
    xaxis='x2'
)
data = [trace1, trace2, trace3, trace4]

layout = go.Layout(
    showlegend=False,
    autosize=False,
    width=600,
    height=550,
    xaxis=dict(
        domain=[0, 0.85],
        showgrid=False,
        zeroline=False
    ),
    yaxis=dict(
        domain=[0, 0.85],
        showgrid=False,
        zeroline=False
    ),
    margin=dict(
        t=50
    ),
    hovermode='closest',
    bargap=0,
    xaxis2=dict(
        domain=[0.85, 1],
        showgrid=False,
        zeroline=False
    ),
    yaxis2=dict(
        domain=[0.85, 1],
```

```
        showgrid=False,
        zeroline=False
    )
)

fig = go.Figure(data=data, layout=layout)
py.iplot(fig)
```

price_GrLivArea.py hosted with ♡ by GitHub view raw

price_GrLivArea.py

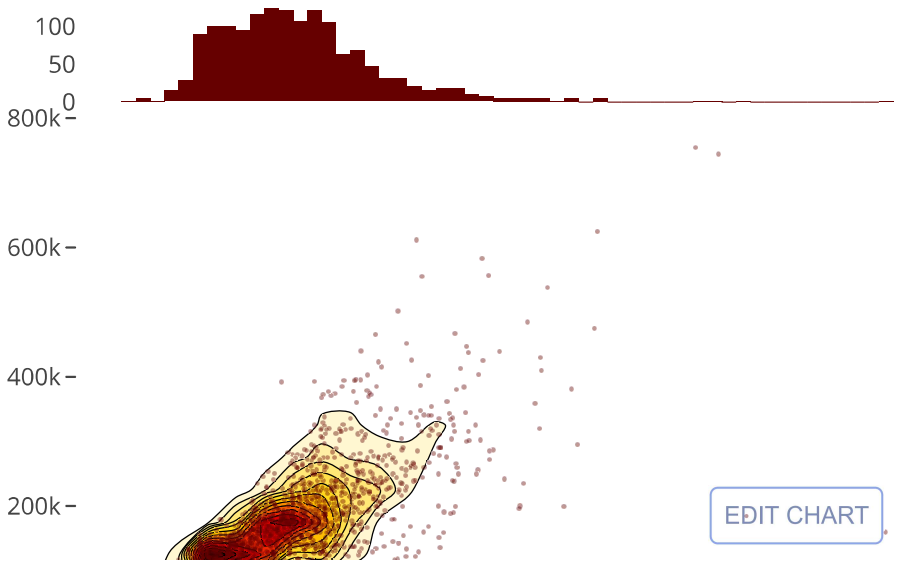


Figure 15

Heterogeneity and stratification

We continue exploring the relationship between SalePrice and GrLivArea, stratifying by BldgType.

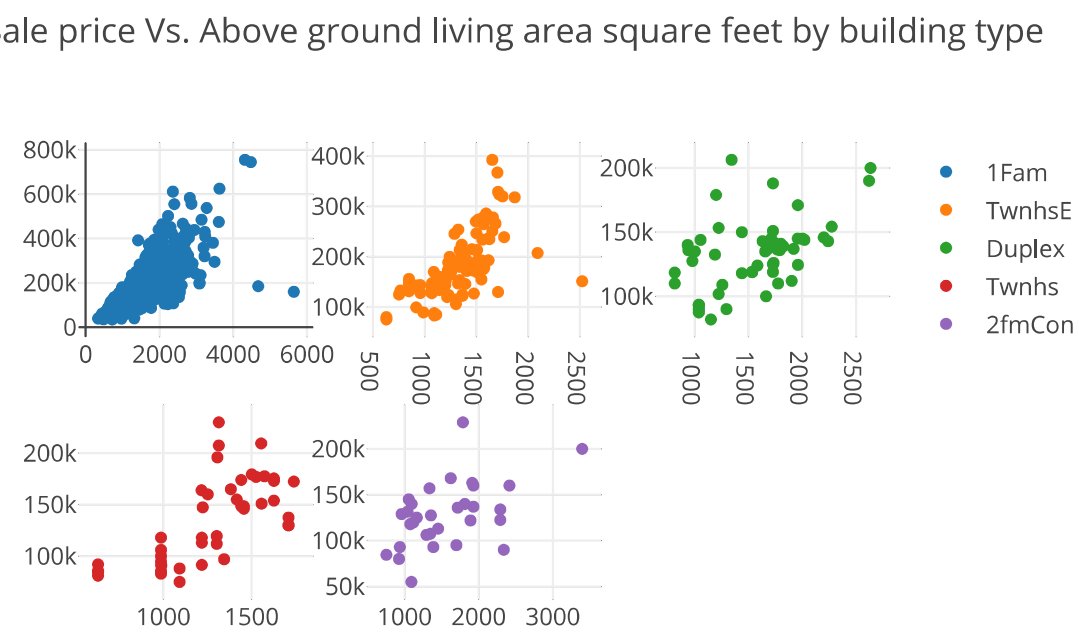
```
trace0 = go.Scatter(x=df.loc[df['BldgType'] == '1Fam']['GrLivArea'], y=df.loc[df['BldgType'] ==
trace1 = go.Scatter(x=df.loc[df['BldgType'] == 'TwnhsE']['GrLivArea'], y=df.loc[df['BldgType'] =
trace2 = go.Scatter(x=df.loc[df['BldgType'] == 'Duplex']['GrLivArea'], y=df.loc[df['BldgType'] =
```

```
trace3 = go.Scatter(x=df.loc[df['BldgType'] == 'Twnhs']['GrLivArea'], y=df.loc[df['BldgType'] ==
trace4 = go.Scatter(x=df.loc[df['BldgType'] == '2fmCon']['GrLivArea'], y=df.loc[df['BldgType'] =

fig = tools.make_subplots(rows=2, cols=3)

fig.append_trace(trace0, 1, 1)
fig.append_trace(trace1, 1, 2)
fig.append_trace(trace2, 1, 3)
fig.append_trace(trace3, 2, 1)
fig.append_trace(trace4, 2, 2)

fig['layout'].update(height=400, width=800, title='Sale price Vs. Above ground living area squar
                                                  ' by building type')
py.iplot(fig)
```

stratify.py hosted with ♡ by GitHub view raw

stratify.py

```
This is the format of your plot grid:
[ (1,1) x1,y1 ]  [ (1,2) x2,y2 ]  [ (1,3) x3,y3 ]
[ (2,1) x4,y4 ]  [ (2,2) x5,y5 ]  [ (2,3) x6,y6 ]
```



EDIT CHART

Figure 16

In almost all the building types, SalePrice and GrLivArea shows a positive linear relationship. In the results below, we see that the correlation between SalepPrice and GrLivArea in 1Fam building type is the highest at 0.74, while in Duplex building type the correlation is the lowest at 0.49.

```
print(df.loc[df.BldgType=="1Fam", ["GrLivArea", "SalePrice"]].corr())
print(df.loc[df.BldgType=="TwnhsE", ["GrLivArea",
"SalePrice"]].corr())
print(df.loc[df.BldgType=='Duplex', ["GrLivArea",
"SalePrice"]].corr())
print(df.loc[df.BldgType=="Twnhs", ["GrLivArea",
"SalePrice"]].corr())
print(df.loc[df.BldgType=="2fmCon", ["GrLivArea",
"SalePrice"]].corr())
```

```
             GrLivArea  SalePrice
GrLivArea     1.000000   0.738956
SalePrice     0.738956   1.000000
             GrLivArea  SalePrice
GrLivArea     1.000000   0.641622
SalePrice     0.641622   1.000000
             GrLivArea  SalePrice
GrLivArea     1.000000   0.490441
SalePrice     0.490441   1.000000
             GrLivArea  SalePrice
GrLivArea      1.00000    0.66924
SalePrice      0.66924    1.00000
             GrLivArea  SalePrice
GrLivArea     1.000000   0.498502
SalePrice     0.498502   1.000000
```

Table 10

Categorical bivariate analysis

We create a contingency table, counting the number of houses in each cell defined by a combination of building type and the general zoning classification.

```
x = pd.crosstab(df.MSZoning, df.BldgType)
x
```

BldgType MSZoning	1Fam	2fmCon	Duplex	Twnhs	TwnhsE
C (all)	9	1	0	0	0
FV	38	0	0	9	18
RH	9	2	3	0	2
RL	1025	16	43	10	57
RM	139	12	6	24	37

Table 11

Below we normalize within rows. This gives us the proportion of houses in each zoning classification that fall into each building type variable.

```
x.apply(lambda z: z/z.sum(), axis=1)
```

Table 12

We can also normalize within the columns. This gives us the proportion of houses within each building type that fall into each zoning classification.

```
x.apply(lambda z: z/z.sum(), axis=0)
```


Table 13

One step further, we will look at the proportion of houses in each zoning class, for each combination of the air conditioning and building type variables.

```
df.groupby(["CentralAir", "BldgType",
"MSZoning"]).size().unstack().fillna(0).apply(lambda x: x/x.sum(),
axis=1)
```

Table 14

The highest proportion of houses in the data are the ones with zoning RL, with air conditioning and 1Fam building type. With no air conditioning, the highest proportion of houses are the ones in zoning RL and Duplex building type.

Mixed categorical and quantitative data

To get fancier, we are going to plot a violin plot to show the distribution of SalePrice for houses that are in each building type category.

```
data = []
for i in range(0,len(pd.unique(df['BldgType']))):
    trace = {
            "type": 'violin',
            "x": df['BldgType'][df['BldgType'] == pd.unique(df['BldgType'])[i]],
            "y": df['SalePrice'][df['BldgType'] == pd.unique(df['BldgType'])[i]],
            "name": pd.unique(df['BldgType'])[i],
            "box": {
                "visible": True
            },
            "meanline": {
                "visible": True
            }
        }
    data.append(trace)
```

```
fig = {
    "data": data,
    "layout" : {
        "title": "",
        "yaxis": {
            "zeroline": False,
        }
    }
}


py.iplot(fig)
```

price_violin_plot.py hosted with ♡ by GitHub view raw

price_violin_plot.py

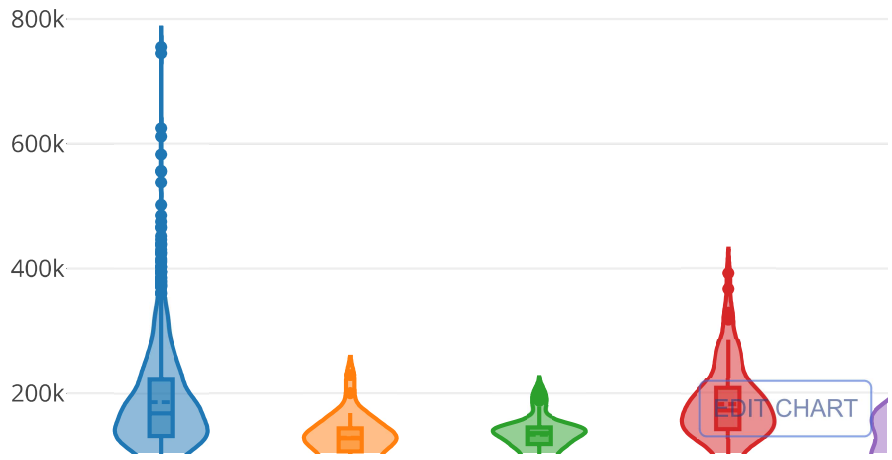


Figure 17

We can see that the SalesPrice distribution of 1Fam building type are slightly right-skewed, and for the other building types, the SalePrice distributions are nearly normal.

Jupyter notebook for this post can be found on Github, and there is an nbviewer version as well.

Reference:

Statistics with Python | Coursera

This specialization is designed to teach learners beginning and intermediate concepts of statistical analysis using the...

www.coursera.or

Data Science Statistics Tutorial Python Machine Learning

About Help Legal