

An aerial photograph of Rio de Janeiro, Brazil, showing the bay, mountains, and city skyline. The bay is filled with many sailboats. The city is built on a hillside, and the mountains are visible in the background.

The Price of Living in Brazil

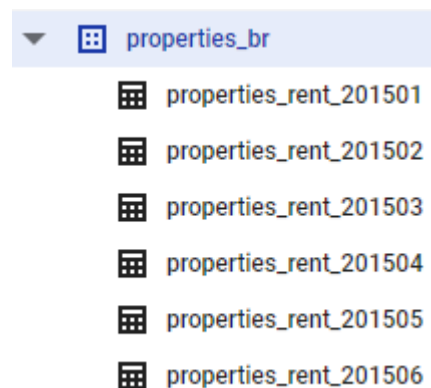
An Exploratory Data Analysis of the Brazil Real Estate Listings from Properati

@author: Reinelle Jan Bugnot

@contact: reinbugnot@gmail.com

Reference: Exploratory Data Analysis - Brazil Real Estate Listings.ipynb

Properati's Real Estate Listings 2015-2018



▼	properties_br
📁	properties_rent_201501
📁	properties_rent_201502
📁	properties_rent_201503
📁	properties_rent_201504
📁	properties_rent_201505
📁	properties_rent_201506

What's the properati about?

A collection of **76 datasets** containing property rent and sale data from Jan 2015 to Feb 2018

Field name	Type
id	STRING
created_on	DATE
operation	STRING
property_type	STRING
place_name	STRING
place_with_parent_names	STRING

What's inside?

Each dataset contains multiple entries with **27 features** including basic descriptions, price-related features, location features, property characteristics, and time-based features.

In total, there are about 15 million entries (rent + sell) across 2015 to 2018.

The Research Objective

Research Question

What factors determine the sale price of a Real Estate Unit (REU) in Brazil?



This house is worth
R\$20,000,000

Oh, too expensive ...

The Research Objective

Theory

Sale prices are determined by **both internal** (characteristics of the unit) **and external** (characteristics of the environment the unit is sold/rented) **factors**

Hypothesis

1. **SIZE**: Bigger REUs are more expensive than smaller REUs
2. **LOCATION**: REUs in Brazil's wealthier districts are more expensive than the REUs in less developed districts
3. **TIME**: Seasonality affects sale/rent prices; i.e., REUs are more expensive on some months in a year than others.
4. **TIME**: There are years when REUs are more expensive than other years.

Part I.

Initial Data Observations

Moving one data from source platform to Python



Code based from Google BigQuery and GCS Documentations

```
In [2]: 1 os.environ['GOOGLE_APPLICATION_CREDENTIALS'] = 'ServiceKey_GoogleCloud.json'
```

```
In [58]: 1 # Import data from BigQuery to Google Cloud Storage (RUN ONCE)
2
3 from google.cloud import bigquery
4 client = bigquery.Client()
5 bucket_name = 'tm-application'
6 project = "properati-data-public"
7 dataset_id = "properties_br"
8 tables = client.list_tables('properati-data-public.properties_br')
9
10 for table in tables:
11     destination_uri = "gs://{}/{}".format(bucket_name, table.table_id)
12     dataset_ref = bigquery.DatasetReference(project, dataset_id)
13     table_ref = dataset_ref.table(table.table_id)
14
15     extract_job = client.extract_table(
16         table_ref,
17         destination_uri,
18         location="US",
19     )
20     extract_job.result()
21
22     print(
23         "Exported {}:{}.{} to {}".format(project, dataset_id, table.table_id, destination_uri)
24     )
```

```
1 ## Google Cloud Storage to Python (Pandas Dataframe)
2
3 filename = 'properties_rent_201501'
4
5 fs = gcsfs.GCSFileSystem(project='tm-application-323908')
6 with fs.open('tm-application/' + filename) as f:
7     data = pd.read_csv(f)
```

Checking the Features

```
0          id
1      created_on
2      operation
3      property_type
4      place_name
5      place_with_parent_names
6      country_name
7      state_name
8      geonames_id
9      lat_lon
10     lat
11     lon
12     price
13     currency
14 price_aprox_local_currency
15     price_aprox_usd
16     surface_total_in_m2
17     surface_covered_in_m2
18     price_usd_per_m2
19     price_per_m2
20     floor
21     rooms
22     expenses
23     properati_url
24     description
25     title
26     image_thumbnail
dtype: object
```

Note: the dataset provided **NO DESCRIPTIONS** for the features so it's up to us to figure out what information they represent.

created_on

Is this when the property was built or listed on the platform?

price and price_aprox_local_currency

What's the different?

date in the filename (ex. properties_rent_201501.csv)

Is this the listing date or the sale/rent date?

id, place_with_parent_names, properati_url, title, image_thumbnail

Does not contain numerical / categorical / ordinal data (only descriptive) so I can drop this to save space in memory.

Assumptions

created_on

... specifies the date when the unit was listed on the platform

price and price_aprox_local_currency

... price more accurately determines the selling/rent price of the unit

date in the filename (ex. properties_rent_201501.csv)

... is this the date the unit was sold/rented on the platform (i.e.transaction date)

Feature Management

Drop unwanted features

Dropping id, place_with_parent_names, properati_url, description, title, and image_thumbnail

```
1 # Remove id, place_with_parent_names, properati_url, description, title, image_thumbnail
2 data.drop(['id', 'place_with_parent_names', 'properati_url', 'description', 'title', 'image_thumbnail'], axis=1, inplace=True)
3 data.head(3)
```

	created_on	operation	property_type	place_name	country_name	state_name	geonames_id	lat_lon	lat	lon	price
0	2014-08-07	rent	store	Catu	Brasil	Bahia	NaN	NaN	NaN	NaN	
1	2015-01-20	rent	apartment	Pernambuco	Brasil	Pernambuco	NaN	-8.126271,-34.903793	-8.126271	-34.903793	12
2	2014-10-29	rent	store	São Paulo	Brasil	São Paulo	NaN	-23.1546253,-45.7906242	-23.154625	-45.790624	

Add required features

Adding the date in the filename as transaction_year and transaction_month

```
1 ## Initial Feature Generation
2
3 # Create a new column 'List_date' based on the filename
4 data['transaction_year'] = filename[-6:-2]
5 data['transaction_month'] = filename[-2:]
6 data.head(3)
```

Remaining Columns

Location Features

place_name
state_name
geonames_id
lat_lon
lat
lon

Time Features

created_on
transaction_year
transaction_month

Unit Features

property_type
surface_total_in_m2
surface_covered_in_m2
floor
rooms

Money Features

currency
price_aprox_local_currency
price_aprox_usd
price_usd_per_m2
price_per_m2
expenses

Misc Features

operation

Target Variable

price

Part II.

Data Profiling and Cleaning

Automate Import Function

the genData() function

Since I anticipate that I'll have to do this repeatedly across the analysis, I wrote a short script to automate Data extraction from Google Cloud Storage (GCS) into a pandas DataFrame.

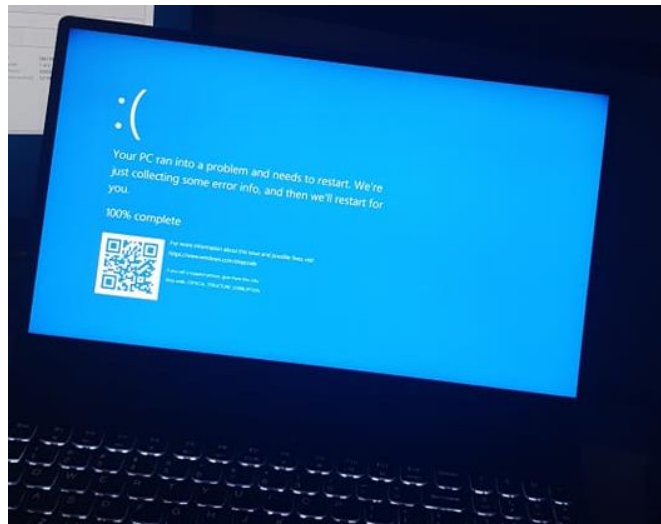
This also includes basic methods for parsing target year, target columns and specifying desired file count.

```
1  ## Automate the Transfer of data from GCS to Python (Pandas Dataframe)
2
3  def genData(year, column = None, file_count = None):
4
5      DATA = []
6
7      for _, _, bucket_data in fs.walk('tm-application/'):
8
9          for filename in tq.tqdm(bucket_data[:file_count]):
10
11              if int(filename[-6:-2]) == year:
12
13                  with fs.open('tm-application/' + filename) as f:
14                      data = pd.read_csv(f)
15
16                      if column:
17                          try:
18                              data = data.loc[:, column]
19                          except:
20                              print('Invalid Column!') # Drop unwanted columns
21
22                      else:
23                          data.drop(['id', 'place_with_parent_names', 'properati_url', 'description', 'title', 'image_thumbnail'], axis=1)
24
25                          # Add list_date column
26                          data['transaction_year'] = filename[-6:-2]
27                          data['transaction_month'] = filename[-2:]
28
29                      DATA.append(data.values)
30
31      if column:
32          DATA = pd.DataFrame(np.concatenate(DATA), columns=[column])
33      else:
34          DATA = pd.DataFrame(np.concatenate(DATA), columns=columns)
35
36      print("Load Complete!")
37
38      return DATA
```


Import Dataset

⚠ Total dataset too large for my Laptop's memory to handle

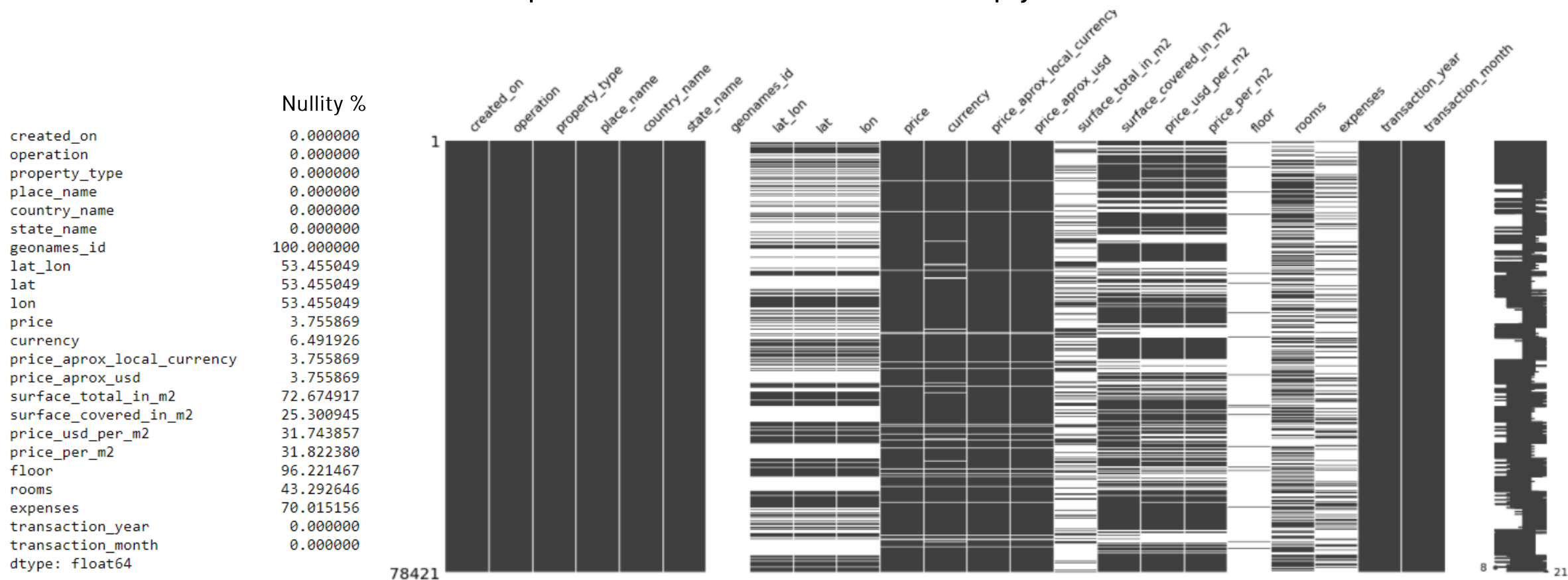
1. Focus comprehensive feature analysis on one target year >> 2016
2. Check if price varies significantly across the years 2016-2018
 - a. If price shifts significantly by year, perform comprehensive analysis of this years (year of sale has correlation)
 - b. If not, analysis on target year should be sufficient to conclude insights (year of sale no correlation)



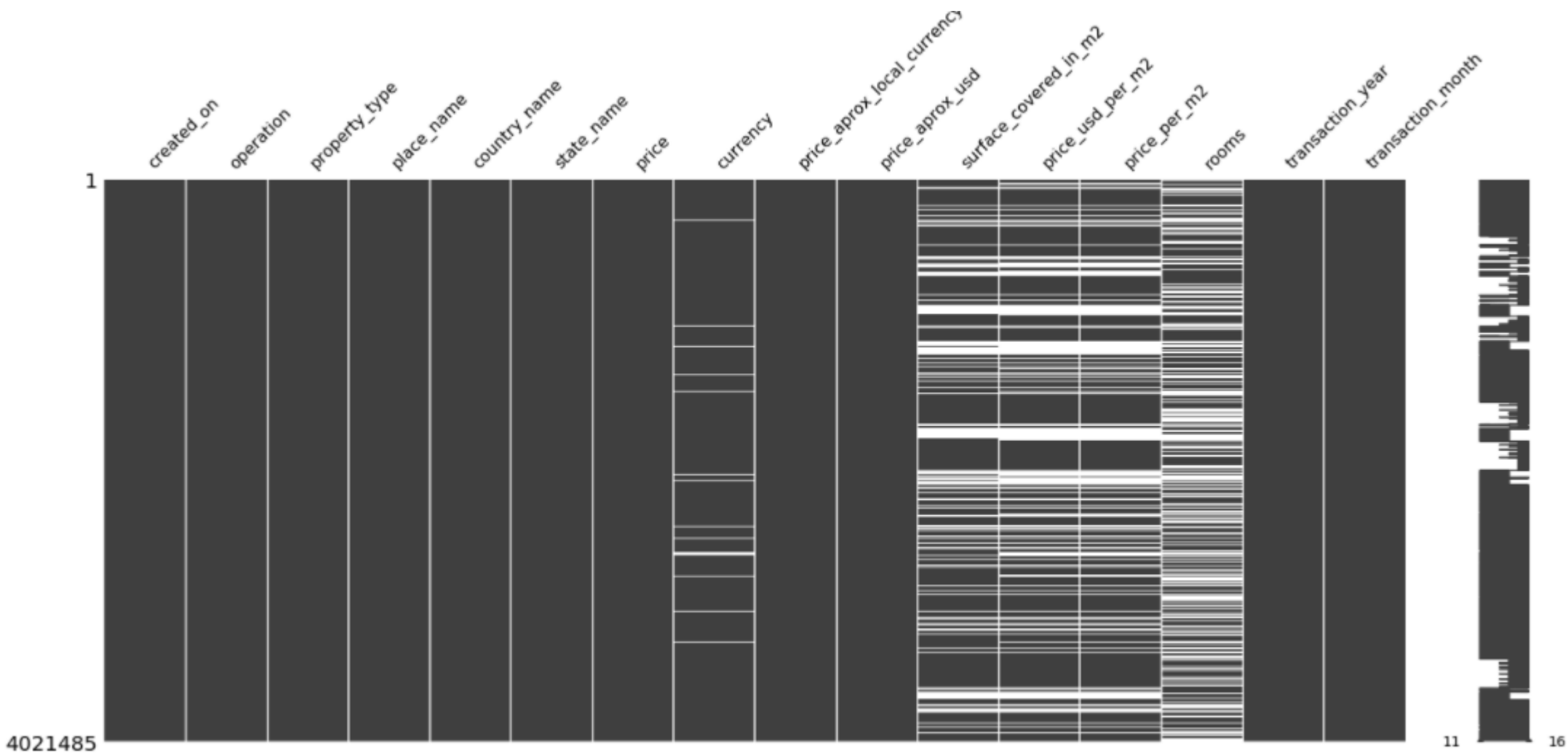
Sad Laptop can't handle processing 15M x 22 datapoints
(will be replaced with an actual rig by Sept)

Handling Missing Numbers

Drop columns that are more than 50% empty



Handling Missing Numbers



Numerical and Categorical Data

Convert numerical data to numerical format

Data points were in string format (maybe due to GCS to pandas transition)

```
In [207]: 1 for i in range(DATA.shape[1]):  
2     try:  
3         DATA.iloc[:,i] = pd.to_numeric(DATA.iloc[:,i])  
4     except:  
5         pass
```

Label-encode Categorical Data

operation, property_type, state_names, etc. are in categorical string format, which cannot be processed by models.

```
In [210]: 1 # Label encode all categorical data  
2  
3 CAT_LABELS = ['operation', 'property_type', 'place_name', 'country_name', 'state_name']  
4 CAT_KEYS = []  
5  
6 for i in CAT_LABELS:  
7     values, keys = pd.factorize(DATA[i])  
8     DATA[i] = values  
9     CAT_KEYS.append(keys)
```


Handling Outliers

View Statistical Summary

using pandas .describe()

	country_name	transaction_year
count	4021485.0	4021485.0
mean	0.0	2016.0
std	0.0	0.0
min	0.0	2016.0
5%	0.0	2016.0
50%	0.0	2016.0
95%	0.0	2016.0
max	0.0	2016.0

constant

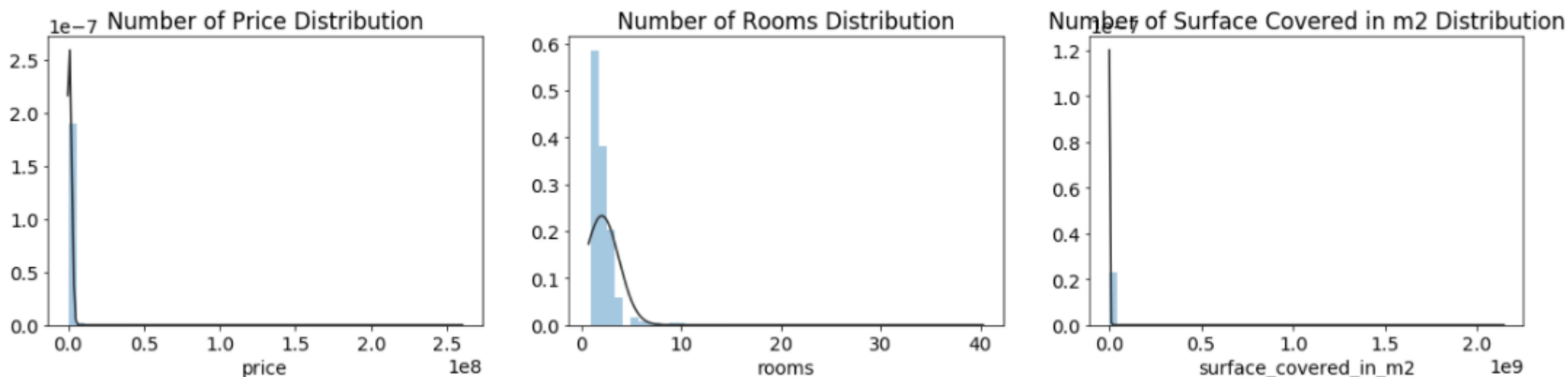
price	surface_covered_in_m2	rooms
4.021485e+06	2.995989e+06	2.332054e+06
7.493559e+05	5.217824e+03	2.048236e+00
1.499651e+06	3.282575e+06	1.713934e+00
0.000000e+00	0.000000e+00	1.000000e+00
8.000000e+02	4.000000e+01	1.000000e+00
4.300000e+05	9.500000e+01	2.000000e+00
2.400000e+06	3.800000e+02	4.000000e+00
2.600000e+08	2.147484e+09	4.000000e+01

Big jump after the 95th percentile

Handling Outliers

Checking the distribution of price, rooms, and surface covered

Data is skewed so hard by the outliers!

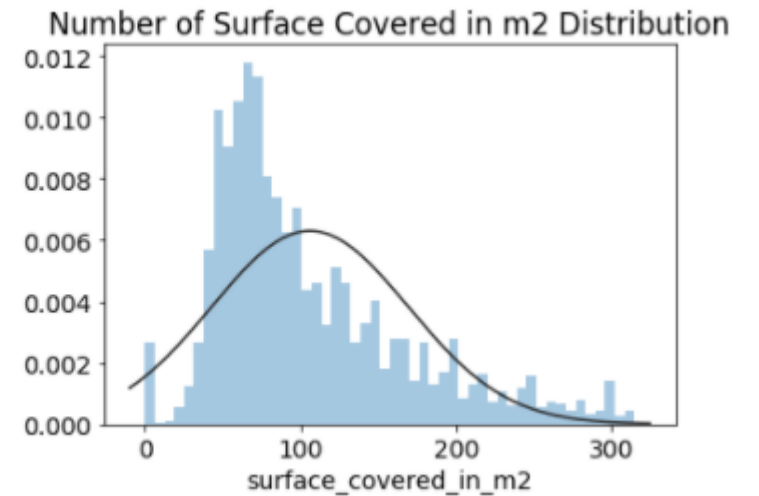
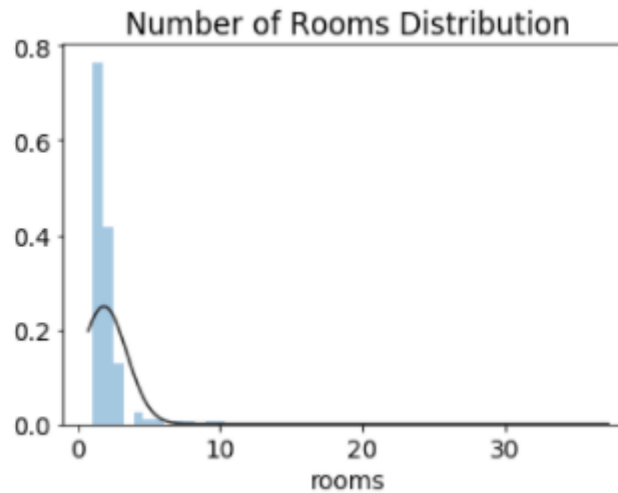
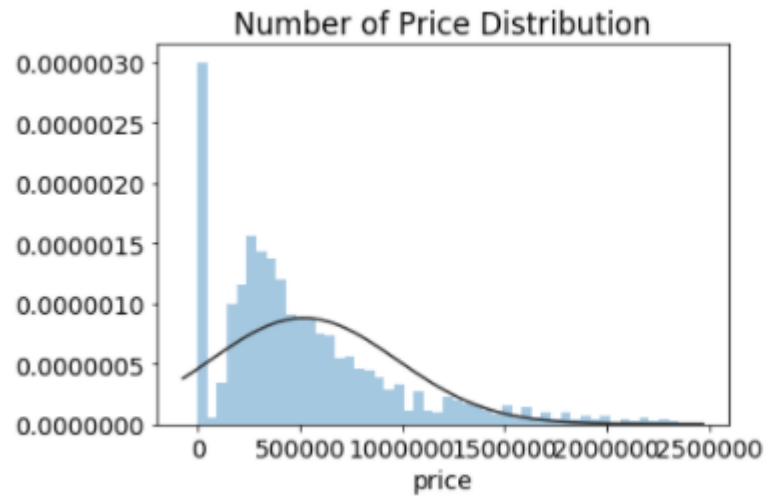


```
In [216]: 1 # Remove outliers outside the 95th percentile for PRICE
          2 upper_quant = numeric_features.price.quantile([.95])
          3 numeric_features = numeric_features.loc[(numeric_features.price < upper_quant.values[0])]
          4
          5 # Remove outliers outside the 5th to 95th percentile for surface_covered_in_m2
          6 upper_quant = numeric_features.surface_covered_in_m2.quantile([.95])
          7 numeric_features = numeric_features.loc[(numeric_features.surface_covered_in_m2 < upper_quant.values[0])]
```

Handling Outliers

Adjusting the percentile for Rooms

As suspected, removing some outliers from surface covered also removed some from rooms

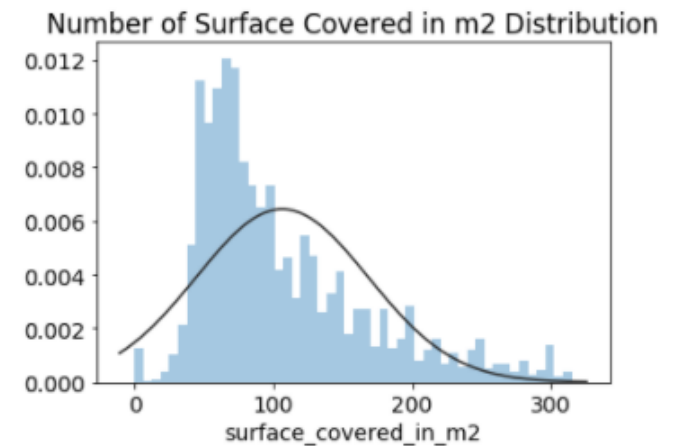
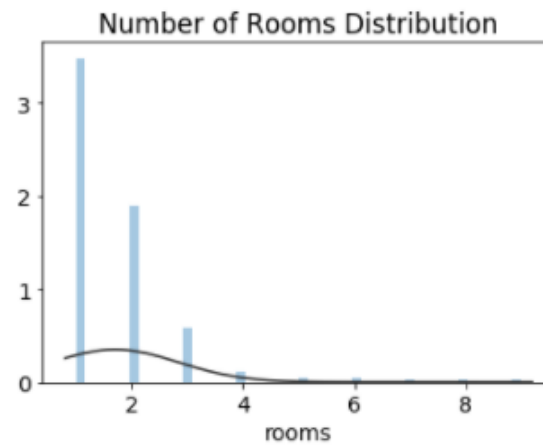
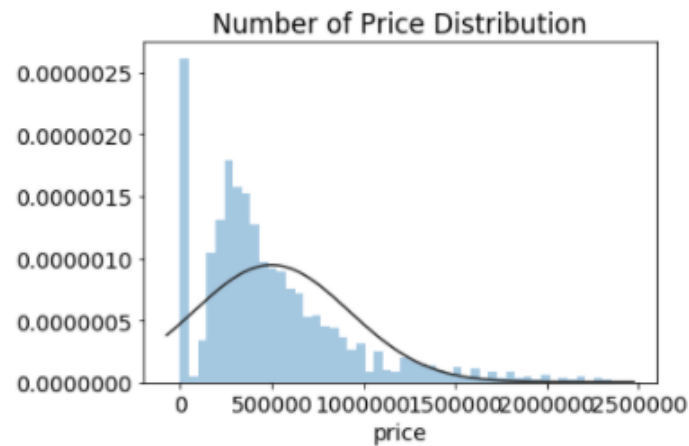


Handling Outliers

Adjusting the percentile for Rooms

As suspected, removing some outliers from surface covered also removed some from rooms

	rooms
count	1.635783e+06
mean	1.817903e+00
std	1.602754e+00
min	1.000000e+00
5%	1.000000e+00
50%	1.000000e+00
99%	1.000000e+01
max	3.700000e+01

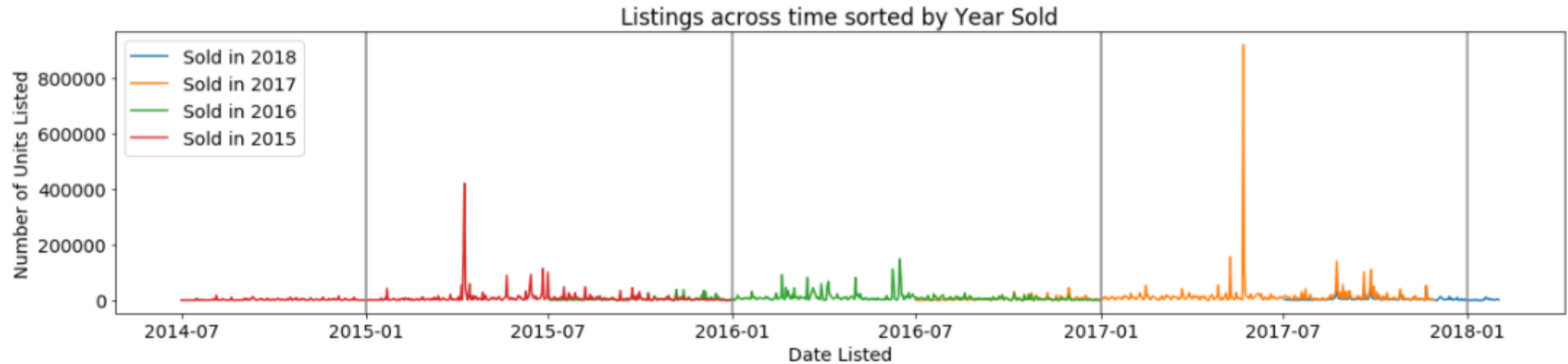


Part III.
Exploratory Data Analysis
(for 2016)

Checking Assumptions

created_on vs date on filename

Confirms (with a high probability) that **created_on** is the date the property was **listed** on the platform; while the **date on the filename** is the date when the property was **sold** on the platform



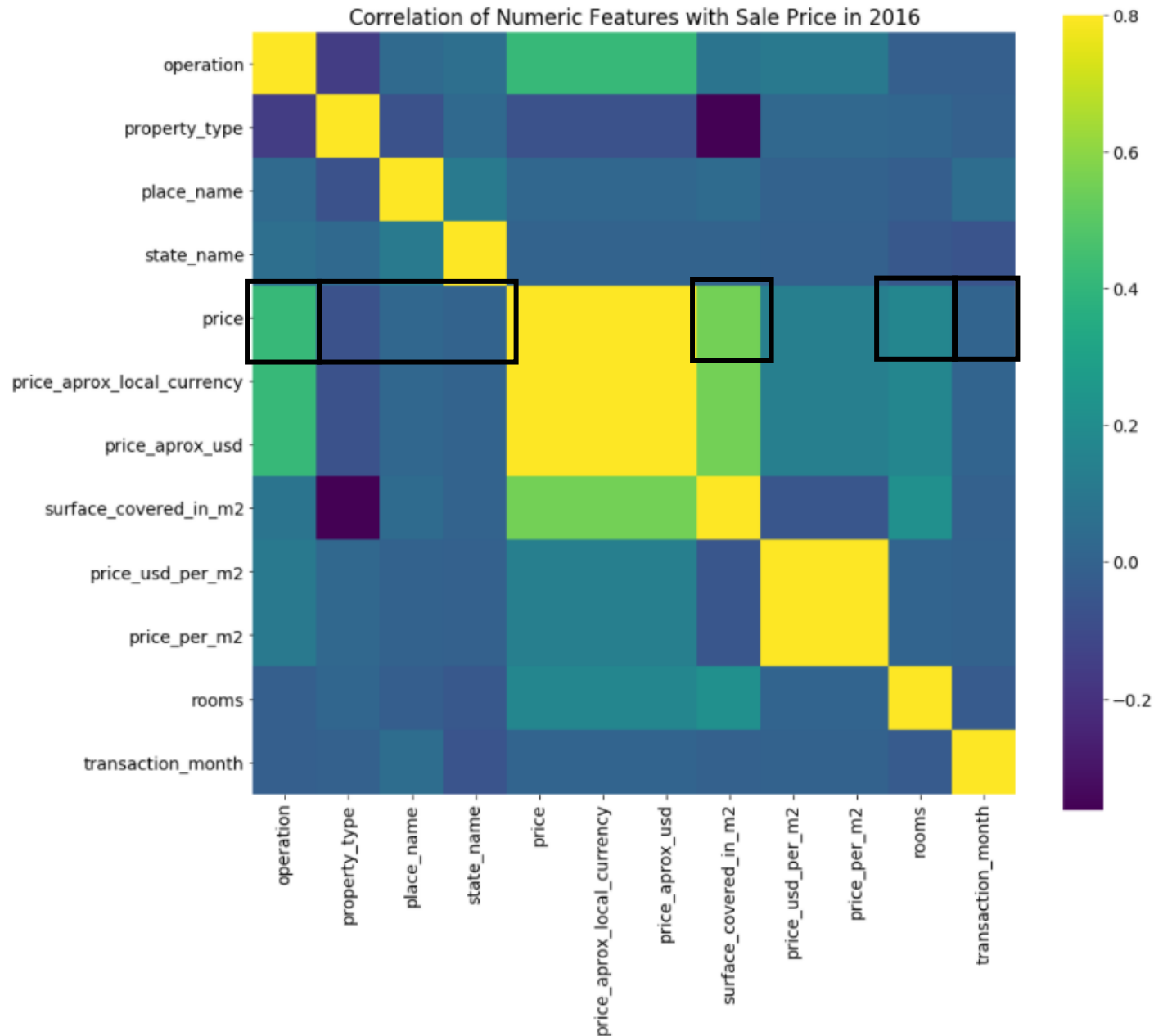
Feature Correlation Plot

Observations

Moderate correlation between
price and operation, and
price and surface_covered_in_m2

Weak Correlation between price and rooms

No Correlation between
price and property_type,
price and location variables, and
price and time variables



Feature Correlation Plot

Observations

Moderate correlation between price and operation, and price and surface_covered_in_m2

Weak Correlation between price and rooms

No Correlation between price and property_type, price and location variables, and price and time variables

Interpretation

Reasonable since selling price is reasonably higher than rent price, and bigger surface covered does affirm our initial hypothesis

More rooms can indicate a bigger unit which can reflect high prices, but not always (ex. 1-2 room penthouse condominium unit)

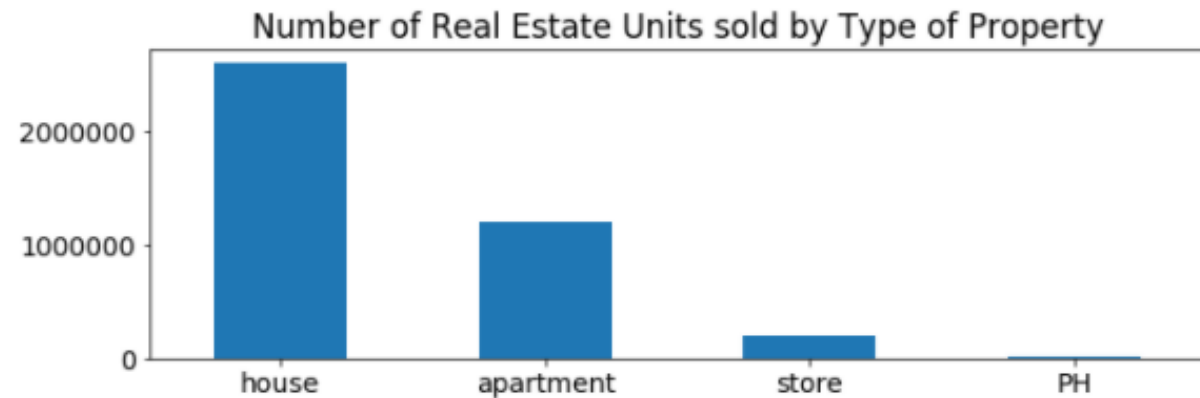
Expected that correlation chart would not reveal the relationship for multi-label categorical data.

Property Type vs Sale Price

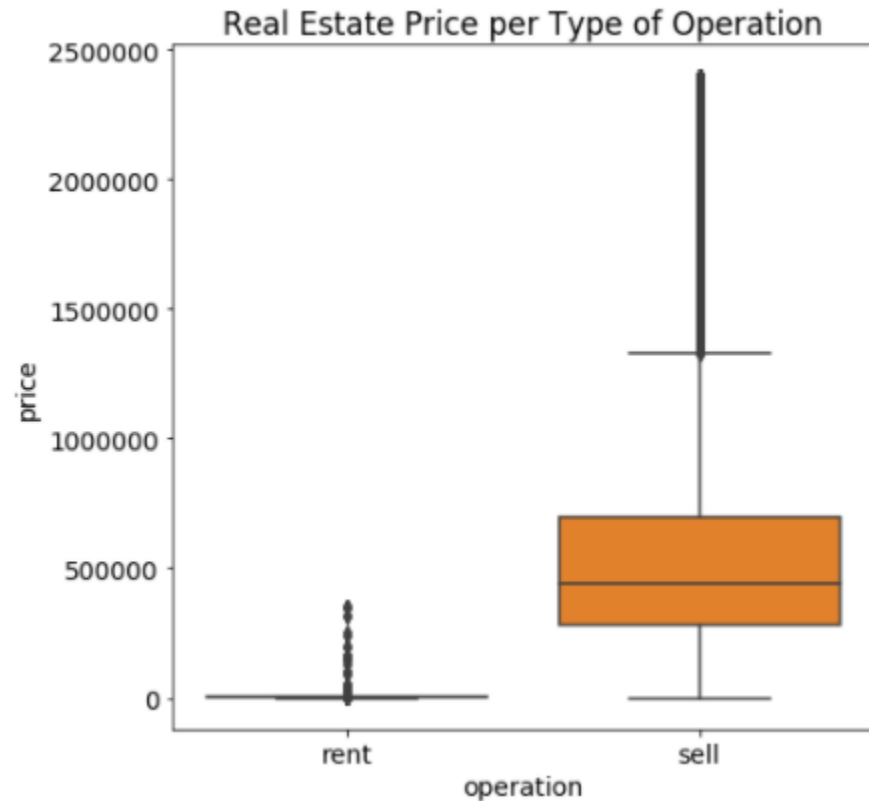


Type of Property is not a good indicator of Price

With the exception of 'store' properties, there seems to be no significant shifts in price ranges across different property types.



Operation vs Sale Price



Operation greatly affects Price

Of course, selling a unit is typically more expensive (in terms of listing price) than renting.

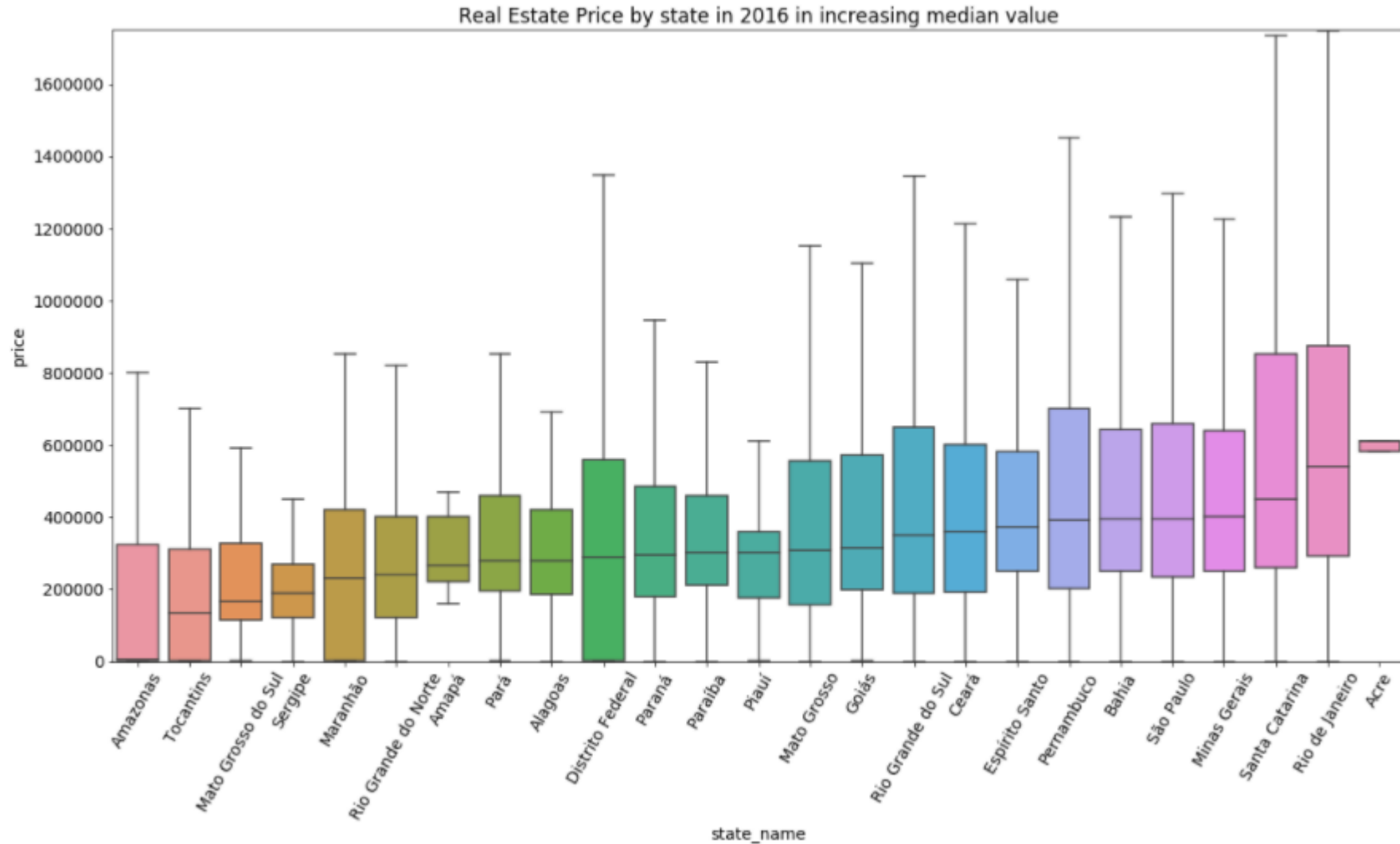
However, it is possible that the drastic difference in price ranges is overpowering some other feature behavior in our dataset. This is explored in the notebook for more details – but the short answer is, no, there is no drastic effect.

Location vs Sale Price

Location does indicate Price

As opposed to what the correlation plot suggests, location does seem to be a good indicator of price.

As seen from the chart, some states generally offer more expensive REUs than others.

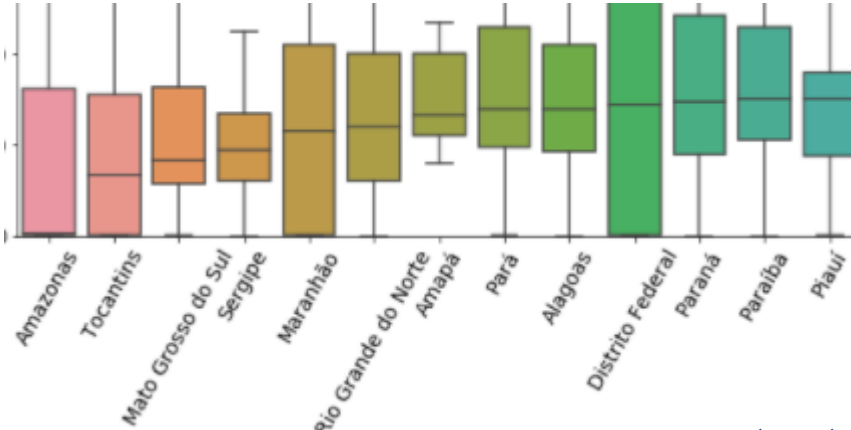
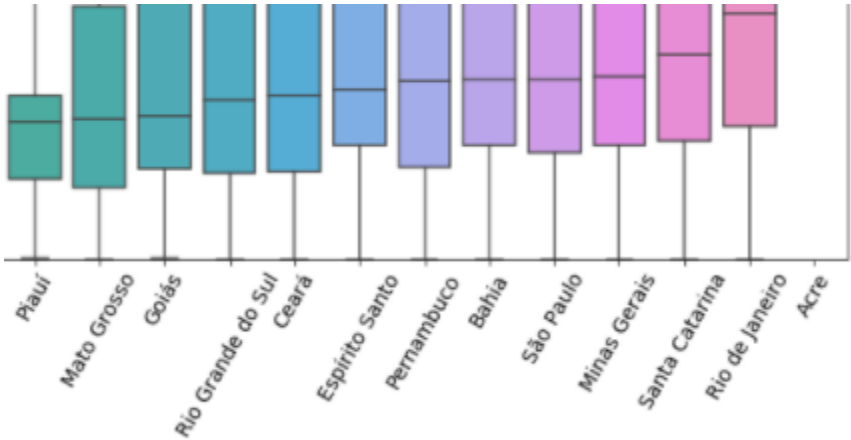


Location vs Sale Price – why?

Rank	State	GDP per capita (R\$)
1	Distrito Federal	64,653
2	São Paulo	33,624
3	Rio de Janeiro	31,064
4	Espírito Santo	29,996
5	Santa Catarina	27,771

23	Ceará	10,473
24	Paraíba	10,151
25	Alagoas	9,333
26	Maranhão	8,760
27	Piauí	8,137

worldatlas.com



Transaction Month vs Sale Price



There is no price seasonality

Prices across different months in the year does not show significant shifts in value, indicating that price is not affected by the month of transaction.

Part IV.
Exploratory Data Analysis
(for 2016–2018)

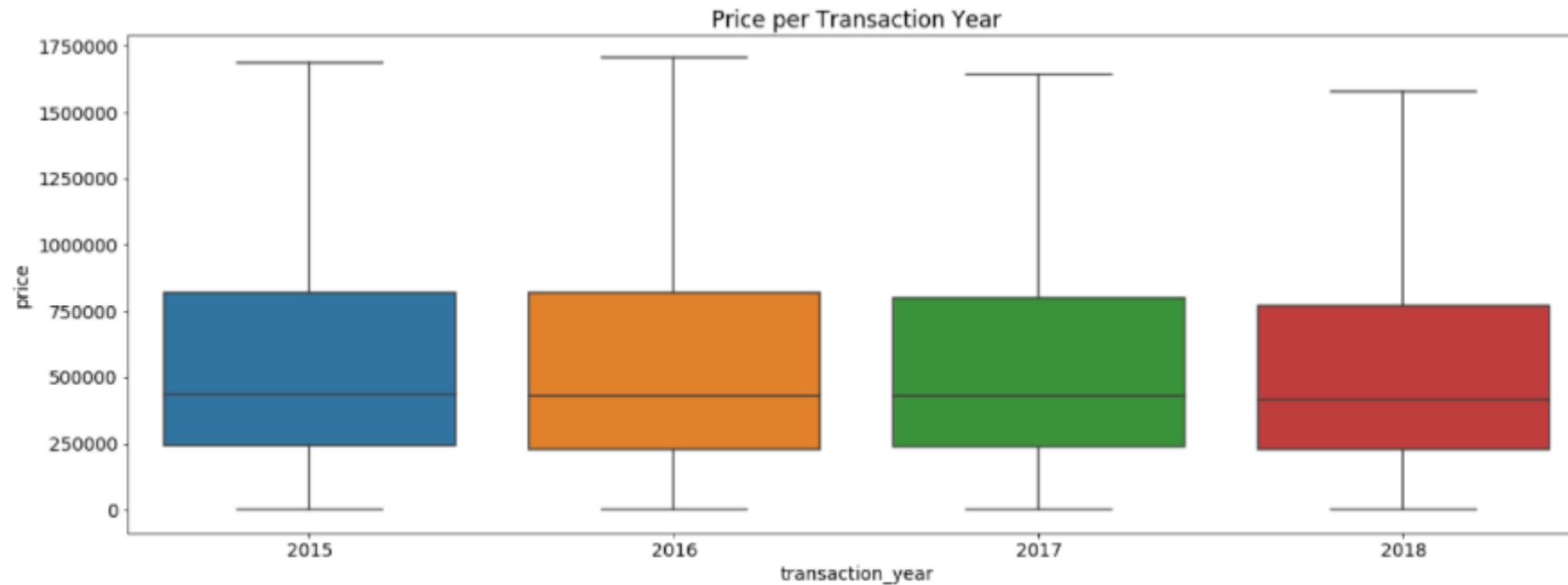
Automate Import Function (updated)

For year-by-year observations

Updated to include the Data Profiling and Cleaning steps indicated above, as well as to optimize data flow to not exceed memory restrictions.

```
In [97]: 1 ## Update the data generator function to automatically drop NaNs and outliers
2 ## Automate the Transfer of data from GCS to Python (Pandas Dataframe)
3
4 def genData2(year, column = None, file_count = None, null_threshold = 50):
5
6     DATA = []
7
8     for _, _, bucket_data in fs.walk('tm-application/'):
9
10         for filename in tq.tqdm(bucket_data[:file_count]):
11
12             if int(filename[-6:-2]) == year:
13
14                 with fs.open('tm-application/' + filename) as f:
15                     data = pd.read_csv(f)
16
17                     # Add List_date column
18                     data['transaction_year'] = filename[-6:-2]
19                     data['transaction_month'] = filename[-2:]
20
21                     if column:
22                         try:
23                             data = data.loc[:, column]
24                         except:
25                             print('Invalid Column!') # Drop unwanted columns
26
27                     else:
28                         data.drop(['id', 'place_with_parent_names', 'properati_url', 'description', 'title', 'image_thumbnail'], axis=1, inplace=True)
29
30                     DATA.append(data.values)
31
32 if column:
33     DATA = pd.DataFrame(np.concatenate(DATA), columns=[column]) #input from user
34 else:
35     DATA = pd.DataFrame(np.concatenate(DATA), columns=columns) #default full columns
36
37 data_cols = DATA.columns #gets the updated column based on column input
38
39 ## Drop missing numbers
40 DATA = DATA[DATA[data_cols[(DATA.isna().mean() * 100) <= null_threshold]]]
41
42 #print("Drop NaNs success!")
43
44 ## Convert to numeric features
45 for i in range(DATA.shape[1]):
46     try:
47         DATA.iloc[:,i] = pd.to_numeric(DATA.iloc[:,i])
48     except:
49         pass
50
51 #print("Convert to Numeric Features success!")
```


Transaction Year vs Sale Price



Real Estate prices are generally consistent across the years

Prices across different years (2015 to 2018) does not show any significant shifts in value, indicating that price is not affected by the year of transaction.

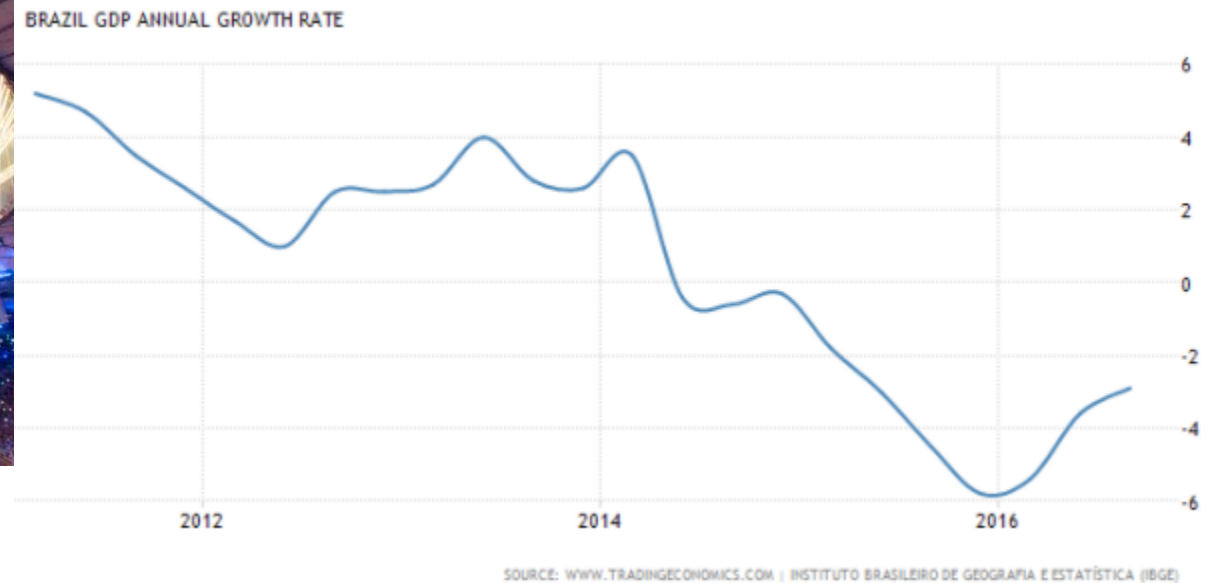
Transaction Year vs Sale Price – why?

The Price per Transaction Year graph is unexpected

Considering 'big events' that happened in Brazil within the 2015 to 2018 time period, I expected to observe some price variations across the years.



Rio 2016 Olympics



GDP leap in 2017 – 2018

Total transactions per State in 2015-2018



Data is heavily biased to Sao Paulo

We run the risk of creating a conclusion that is biased on Sao Paulo's real estate environment than any other states.

Why so many in Sao Paulo?

The Properati startup probably identified Sao Paulo as their beachhead market and focused on capturing Sao Paulo.

Part V.

Conclusion

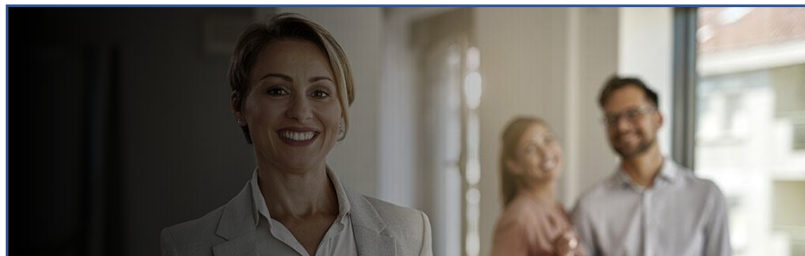
Conclusion

Sale prices are indeed determined by both internal and external factors

Insights:

1. Size of a Real Estate Unit is a good indicator of price. Generally, bigger units are more expensive than smaller ones. Confirms the hypothesis.
2. The location of a Real Estate Unit is a good indicator of price. From the presented plots, it is clear that wealthier states (i.e., more developed) generally have more expensive REUs than less wealthy states as people have a higher buying power. Confirms the hypothesis.
3. Time does not affect price. We saw that prices are generally consistent over months in a year, and across all the years in our 2015-2018 datasets. This rejects the hypothesis.

Who Cares??



Real Estate Agents and Companies

Data-driven people and companies that specifically focus on the real estate market should be aware of what particular factors affect the price of units; and whether these factors comes from reliable and sufficient data.

Home-Buyers

Ex. People who wants to buy the best home they could afford with their budget.

Real Estate Investors

The idea that time does not have immediate effect on real estate prices points investors towards real factors that affect the value of their investments.

Recommendations

More Data for other cities

The information and insight we generated may be heavily biased towards Sao Paulo. In order to get a better picture of the Brazilian Real Estate market, we have to gather more listing information across other states.

or maybe it's NOT biased?

Sao Paulo, being the most populous state in Brazil might be the reason it's also the most active in the Real Estate platform; and hence, perhaps the data we have is insight in and of itself, that Sao Paulo's Real Estate dynamic is a good enough central indicator of Brazil's overall Real Estate development.