Testing the Airbnb-Housing Ratio (AHR): A Case Study in New York City

Can the AHR explain housing price growth and its variability and be applied to future datasets?

Ruiyang Wang

University of Toronto, St. George Campus

February 2025

Introduction

Airbnb is a peer-to-peer platform allowing individuals to rent out properties or rooms for short-term stays. Consumers favor short-term rental (STR) platforms like Airbnb for their flexibility and cost-effectiveness compared to hotels. While STRs reduce market friction and improve housing resource utilization, they also cause local market fluctuations and negative externalities (Filippas & Horton, 2023).

Economic theories suggest STRs affect residential housing markets. Data from FRED shows the median U.S. house price increased from 165,300 dollars in Q1 2000 to 419,200 dollars in Q4 2024. Barron, Kung, and Proserpio (2019) found a 10% rise in Airbnb listings leads to a 0.76% increase in house prices. In Virginia, STRs like Airbnb removed 7,000 to 13,500 units from NYC's long-term housing market, impacting high-priced housing (Wachsmuth et al., 2018). Filippas and Horton (2023) note that high Airbnb density causes noise pollution, reducing housing prices (Trojanek, 2023). In London, a 10% increase in Airbnb listings is linked to a 3% rise in burglaries and five additional robberies per 100 properties (Lanfear & Kirk, 2024). These externalities harm local housing markets (Filippas & Horton, 2023).

As a result, this study use Airbnb-Housing Ratio (AHR) as a metric to analyze STR to housing price growth, and variability. NYC serves as a case study due to its strong tourism industry and high STR demand (Wachsmuth et al., 2018). The analysis integrates the USA Real Estate Dataset, detailing housing prices, and the New York City Airbnb Open Data, containing 2019 Airbnb listings. The datasets were cleaned and merged using Python (Python Software Foundation, 2023) and Jupyter Notebook (Kluyver et al., 2016).

Key variables are summarized through tables and visualizations. Violin plots compare pre and post-2019 New York housing counts, while AHR examines housing price trends. Figure 6 can't reveals a clear relationship between AHR and housing price growth. Addtionally, Figures 7.1 and 7.2 suggest that higher AHR is associated with greater long-term housing price variability in certain boroughs.

Data Cleaning/Loading

In [107…

```
#### Cleaning for US real estate data (1)
import pandas as pd
import os

# Check if the file exists before loading
file_path = r"C:\Users\User\Desktop\ECO225\Datasets\realtor-data.zip.csv"
if os.path.exists(file_path):
    # Load the dataset
    df = pd.read_csv(file_path)

    # Remove all rows with missing values
    df_cleaned = df.dropna()

    # Save the cleaned dataset
    cleaned_file_path = r"C:\Users\User\Desktop\ECO225\Datasets\realtor-data-cle
    df_cleaned.to_csv(cleaned_file_path, index=False)

else:
    print("Error: File not found. Please check the file path.")

# Define the cleaned file path
cleaned_file_path = r"C:\Users\User\Desktop\ECO225\Datasets\realtor-data-cleaned


#Cleaning for US rel estate data (2)
# Load the cleaned dataset
try:
    df = pd.read_csv(cleaned_file_path, parse_dates=['prev_sold_date'], dayfirst
except FileNotFoundError:
    print("Error: Cleaned file not found. Please check the file path.")
    exit()

# Step 1: Remove rows where price is less than a threshold (e.g., remove prices
df_cleaned = df[df['price'] > 2]

# Step 2: Filter for state = "New York"
ny_housing_data = df_cleaned[df_cleaned['state'] == 'New York']

# Step 3: Filter for the time range in 'prev_sold_date'
# Ensure the 'prev_sold_date' is already in datetime format
ny_housing_filtered = ny_housing_data[
    (ny_housing_data['prev_sold_date'] >= '2014-01-01') &
    (ny_housing_data['prev_sold_date'] <= '2024-12-31')
]

# Step 4: Save the filtered dataset to a new file
nyhousing_file_path = r"C:\Users\User\Desktop\ECO225\Datasets\NYhousing.csv"
ny_housing_filtered.to_csv(nyhousing_file_path, index=False)
```

In [207…

```
### Data cleaning/filtering for airbnb dataset
import pandas as pd
import warnings
warnings.filterwarnings("ignore")  # Suppress all warnings
# File path
file_path = r"C:\Users\User\Desktop\ECO225\Datasets\AB_NYC_2019.csv"

# Load the dataset with appropriate encoding
airbnb_data = pd.read_csv(file_path, encoding='latin1')

# Select only the specified columns
selected_columns = ['id', 'name', 'host_id', 'neighbourhood_group', 'neighbourho
filtered_data = airbnb_data[selected_columns]

# Remove rows with NA values in the selected columns
cleaned_data = filtered_data.dropna()

# Save the cleaned dataset to a new CSV file
output_path = r"C:\Users\User\Desktop\ECO225\Datasets\AB_NYC_2019_Cleaned.csv"
cleaned_data.to_csv(output_path, index=False)
```

In [109…

```
### Merging datasets
import pandas as pd

# Load datasets
housing_data = pd.read_csv(r"C:\Users\User\Desktop\ECO225\Datasets\new_york_data
rental_data = pd.read_csv(r"C:\Users\User\Desktop\ECO225\Datasets\AB_NYC_2019_Cl

# Rename 'neighbourhood_group' to 'city' in the rental dataset
rental_data.rename(columns={"neighbourhood_group": "city"}, inplace=True)

# Merge the datasets using an outer join on 'city'
merged_data = pd.merge(housing_data, rental_data, how="outer", on="city")

# Save the merged dataset to a new CSV file
merged_data.to_csv(r"C:\Users\User\Desktop\ECO225\Datasets\merged_data.csv", ind
```

In [211…

```
### calculating the  Airbnb-to-Housing Ratio
import pandas as pd

# Load the Airbnb dataset
airbnb_path = r"C:\Users\User\Desktop\ECO225\Datasets\AB_NYC_2019_Cleaned.csv"
airbnb_data = pd.read_csv(airbnb_path)

# Load the NY housing dataset
housing_path = r"C:\Users\User\Desktop\ECO225\Datasets\new_york_data.csv"
housing_data = pd.read_csv(housing_path)

# Step 1: Count the number of Airbnb listings per city
airbnb_counts = airbnb_data.groupby('neighbourhood_group').size().reset_index(na

# Step 2: Count the number of housing units per city
housing_counts = housing_data.groupby('city').size().reset_index(name='housing_c

# Step 3: Merge the counts based on the city
city_data = pd.merge(airbnb_counts, housing_counts, left_on='neighbourhood_group

# Step 4: Calculate the Airbnb-to-Housing Ratio
city_data['airbnb_to_housing_ratio'] = city_data['airbnb_count'] / city_data['hc
```

```
# Step 5: Save the result to a CSV file
output_path = r"C:\Users\User\Desktop\ECO225\Datasets\airbnb_to_housing_ratio.cs
city_data.to_csv(output_path, index=False)

### Summary table for the ratio
import pandas as pd

# Load the merged dataset (with the ratio)
merged_data_path = r"C:\Users\User\Desktop\ECO225\Datasets\airbnb_to_housing_rat
merged_data = pd.read_csv(merged_data_path)

# Calculate summary statistics
summary_stats = merged_data[['airbnb_to_housing_ratio']].describe().T

# Add additional metrics like standard deviation (if not included in describe())
summary_stats['std'] = merged_data['airbnb_to_housing_ratio'].std()

# Rename columns for better readability
summary_stats = summary_stats.rename(columns={
    'mean': 'Mean',
    'std': 'Standard Deviation',
    'min': 'Minimum',
    '25%': '25th Percentile',
    '50%': 'Median',
    '75%': '75th Percentile',
    'max': 'Maximum'
})
```

In [124…

```
### Cleaning/filtering merged dataset
import warnings
warnings.filterwarnings("ignore")  # Suppress all warnings
import pandas as pd

# Load the merged dataset
merged_data_path = r"C:\Users\User\Desktop\ECO225\Datasets\merged_data.csv"
merged_data = pd.read_csv(merged_data_path)

# Check for duplicates
duplicates_count = merged_data.duplicated().sum()

# Drop duplicate rows
merged_data_cleaned = merged_data.drop_duplicates()

# Filter specific variables, including 'prev_sold_date'
###(Note that here I changed the airbnb price to be rental_price,
### and housing price to be housing_price by my hand in Excel,
### it was originally called price_x and price_y)
filtered_data = merged_data_cleaned[['street', 'id', 'city', 'bed', 'bath', 'acr
                                     'rental_price', 'room_type', 'prev_sold_dat

# Convert 'prev_sold_date' to datetime format
filtered_data['prev_sold_date'] = pd.to_datetime(filtered_data['prev_sold_date']

# Filter for dates between 2014 and 2024
filtered_data_time = filtered_data[
    (filtered_data['prev_sold_date'] >= '2014-01-01') &
    (filtered_data['prev_sold_date'] <= '2024-12-31')
]
```

```
# Save the cleaned and filtered dataset
filtered_data_path = r"C:\Users\User\Desktop\ECO225\Datasets\merged_data_final1.
filtered_data_time.to_csv(filtered_data_path, index=False)
```

In [126…

```
import pandas as pd

### This script processes a merged dataset
### and uses an external dataset (`NYhousing.csv`)
### to calculate and add three new variables:

import pandas as pd
# File paths
merged_data_path = r'C:\Users\User\Desktop\ECO225\Datasets\merged_data_final1.cs
ny_housing_path = r'C:\Users\User\Desktop\ECO225\Datasets\NYhousing.csv'
updated_data_path = r'C:\Users\User\Desktop\ECO225\Datasets\merged_with_counts_a

# Step 1: Load the datasets
merged_data = pd.read_csv(merged_data_path)
ny_housing_data = pd.read_csv(ny_housing_path)

# Step 2: Calculate housing_count for each city from `NYhousing.csv`
# Each row in `NYhousing.csv` represents a new housing unit
housing_count = ny_housing_data.groupby('city').size().to_dict()

# Step 3: Calculate Airbnb count for each city from the merged dataset
# Each unique 'id' represents a unique Airbnb listing
airbnb_count = merged_data.groupby('city')['id'].nunique().to_dict()

# Step 4: Calculate Airbnb-to-housing ratio
# Compute ratio only for cities that have both housing and Airbnb counts
airbnb_housing_ratio = {
    city: airbnb_count[city] / housing_count[city]
    for city in airbnb_count if city in housing_count and housing_count[city] >
}

# Step 5: Add the aggregated values directly into the dataset
# Map the values to each row based on the city
merged_data['housing_count'] = merged_data['city'].map(housing_count)
merged_data['airbnb_count'] = merged_data['city'].map(airbnb_count)
merged_data['airbnb_housing_ratio'] = merged_data['city'].map(airbnb_housing_rat

# Step 6: Save the updated dataset
merged_data.to_csv(updated_data_path, index=False)
```

Summary Statistic Tables

We selected eight independent variables and one response variable including controlled variables like 'bed', 'bath','room_type' and 'acre_lot' (house area) affect prices and help group data for analysis. Table 1 and 2 shows those controlled variables along with response. Table 3 displays a new variable that direcly help further analysis on price growth rate.

In [111…

```
### Generate SUmmary table 1
import pandas as pd

# File path for the dataset
```

```
dataset_path = r"C:\Users\User\Desktop\ECO225\Datasets\merged_with_counts_and_r

# Step 1: Load the dataset
data = pd.read_csv(dataset_path)

# Step 2: Select numerical variables excluding certain columns
excluded_columns = ['airbnb_housing_ratio', 'housing_count', 'airbnb_count', 's
numerical_columns = [
    col for col in data.columns
    if data[col].dtype in ['int64', 'float64'] and col not in excluded_columns
]

# Step 3: Generate summary statistics
summary_statistics = data[numerical_columns].describe()

# Step 4: Remove IQR rows (25%, 50%, 75%)
summary_statistics = summary_statistics.drop(index=['25%', '50%', '75%'])

# Step 5: Center-align text in the table and display it
try:
    from IPython.display import display
    styled_table = (
        summary_statistics.style
        .format("{:.2f}")
        .set_caption("<b>Table 1: Summary Statistics Table</b>")
        .set_table_styles([
            {'selector': 'td', 'props': [('text-align', 'center')]},  # Center
            {'selector': 'th', 'props': [('text-align', 'center')]}   # Center
        ])
    )
    display(styled_table)
except ImportError:
    pass
```

Table 1: Summary Statistics Table

	bed	bath	acre_lot	rental_price	housing_price
count	468405.00	468405.00	468405.00	465697.00	468405.00
mean	3.28	2.03	2.36	107.02	560537.21
std	2.29	1.31	296.29	156.39	375773.04
min	1.00	1.00	0.01	0.00	13000.00
max	23.00	16.00	100000.00	10000.00	27500000.00

Table 1 summarizes key variables, including 'bed', 'bath', 'acre_lot', 'rental_price', and 'housing_price,' with 468,405 observations for New York housing and 465,697 for Airbnb listings. On average, houses on sale have 3.28 bedrooms, 2.03 bathrooms, a 2.36-acre lot, and a price of 560,537.21, while short-term rentals (STRs) average 107.02. However, the dataset shows high variability due to large standard deviations. For example, housing prices range up to 27,500,000.

In [113…

```
### Summary table 2 for room_type
import pandas as pd
```

```
# File path for the dataset
dataset_path = r"C:\Users\User\Desktop\ECO225\Datasets\AB_NYC_2019_Cleaned.csv'

# Step 1: Load the Airbnb dataset
data = pd.read_csv(dataset_path)

# Step 2: Filter valid room types
valid_room_types = ['Private room', 'Entire home/apt', 'Shared room']
data = data[data['room_type'].isin(valid_room_types)]

# Step 3: Group by 'neighbourhood_group' and 'room_type', count occurrences
room_type_summary = (
    data.groupby(['neighbourhood_group', 'room_type'])
    .size()
    .reset_index(name='count')  # Rename size count to 'count'
    .pivot(index='neighbourhood_group', columns='room_type', values='count')  #
    .fillna(0)  # Fill missing values with 0
)

# Step 4: Add Total Listings
room_type_summary['Total Listings'] = room_type_summary.sum(axis=1)

# Step 5: Reset index for display
room_type_summary = room_type_summary.reset_index()

# Step 6: Remove unnecessary 'room_type' column
room_type_summary = room_type_summary.loc[:, ~room_type_summary.columns.str.con

# Step 7: Format only the numeric columns
try:
    from IPython.display import display
    numeric_columns = ['Entire home/apt', 'Private room', 'Shared room', 'Total
    display(
        room_type_summary.style.format(
            {col: "{:.0f}" for col in numeric_columns}  # Apply formatting only
        ).set_caption("<b>Table 2: Room Type Summary by Boroughs (Top 5)<b>")
        .set_table_styles([
            {'selector': 'td', 'props': [('text-align', 'center')]},  # Center
            {'selector': 'th', 'props': [('text-align', 'center')]}   # Center
            ])
    )
except ImportError:
    print("IPython is not available. Here is the raw table:")
    print(room_type_summary)
```

Table 2: Room Type Summary by Boroughs (Top 5)

room_type	neighbourhood_group	Entire home/apt	Private room	Shared room	Total Listings
0	Bronx	374	649	59	1082
1	Brooklyn	9462	10034	410	19906
2	Manhattan	13052	7866	477	21395
3	Queens	2066	3289	195	5550
4	Staten Island	175	184	9	368

Table 2 summarizes the 'room_type' variable in the Airbnb dataset for the top 5 boroughs. The three categories are 'Entire home/apt,' 'Private room,' and 'Shared room.' 'Entire home/apt' includes listings renting out whole homes, with the Bronx having 374 listings. 'Private room,' the most common and affordable option, has the highest number in Brooklyn in 2019 (AirDNA, 2023). 'Shared room,' where guests share a space with others, is the least popular category.

In [115…

```
#Summary table 3 for Housing and Airbnb in different city.
import pandas as pd

# File paths for the datasets
ny_housing_path = r"C:\Users\User\Desktop\ECO225\Datasets\NYhousing.csv"  # Rep
airbnb_data_path = r"C:\Users\User\Desktop\ECO225\Datasets\AB_NYC_2019_Cleaned

# Step 1: Load the datasets
ny_housing = pd.read_csv(ny_housing_path)
airbnb_data = pd.read_csv(airbnb_data_path)

# Step 2: Count the number of observations per city for NYhousing
housing_counts = ny_housing.groupby('city').size().reset_index(name='housing_co

# Step 3: Count the number of observations per neighbourhood_group for AB_NYC_2
airbnb_counts = airbnb_data.groupby('neighbourhood_group').size().reset_index(r

# Step 4: Merge the two datasets on the city/neighbourhood_group variables
summary_table = housing_counts.merge(
    airbnb_counts,
    left_on='city',
    right_on='neighbourhood_group',
    how='inner'
)

# Step 5: Eliminate rows where housing_count <= 1 or airbnb_count <= 1
summary_table = summary_table[(summary_table['housing_count'] > 1) & (summary_t

# Step 6: Calculate Airbnb-to-Housing Ratio
summary_table['airbnb_housing_ratio'] = summary_table['airbnb_count'] / summary

# Step 7: Keep only relevant columns and rename for clarity
summary_table = summary_table[['city', 'housing_count', 'airbnb_count', 'airbnb

# Step 8: Sort the table by ratio (high to low)
summary_table = summary_table.sort_values(by='airbnb_housing_ratio', ascending=

# Step 9: Display the summary table in an academic style
try:
    from IPython.display import display
    display(
        summary_table.style.format(
            {
                'housing_count': "{:.0f}",
                'airbnb_count': "{:.0f}",
                'airbnb_housing_ratio': "{:.2f}"
            }
        ).set_caption("<b>Table 3: Housing and Airbnb Summary (Filtered and Sor
        .set_table_styles([
```

```
            {'selector': 'td', 'props': [('text-align', 'center')]},  # Center
            {'selector': 'th', 'props': [('text-align', 'center')]}   # Center
            ])
    )
except ImportError:
    print("IPython is not available. Here is the raw table:")
    print(summary_table)

# Step 10: Save the summary table to a CSV file
output_path = r"C:\Users\User\Desktop\ECO225\Datasets\ratio_summary.csv"  # Rep
summary_table.to_csv(output_path, index=False)
```

Table 3: Housing and Airbnb Summary (Filtered and Sorted by Ratio)

	city	housing_count	airbnb_count	airbnb_housing_ratio
14	Queens	19	5550	292.11
3	Brooklyn	688	19906	28.93
2	Bronx	200	1082	5.41
4	Chelsea	2	9	4.50
13	Long Island City	2	9	4.50
8	Elmhurst	4	5	1.25
9	Flushing	53	24	0.45
19	Sunnyside	5	2	0.40
20	Woodhaven	10	3	0.30
18	Staten Island	1244	368	0.30
5	College Point	12	3	0.25
10	Forest Hills	16	4	0.25
11	Fresh Meadows	15	3	0.20

Table 3 publishes a key variable in this descriptive data analysis, 'airbnb_housing_ratio', which is calculated as:

$$\text{Airbnb Housing Ratio(AHR)} = \frac{\text{Number of Airbnb listings in one city}}{\text{Number of on-sale housing in the same city}}$$

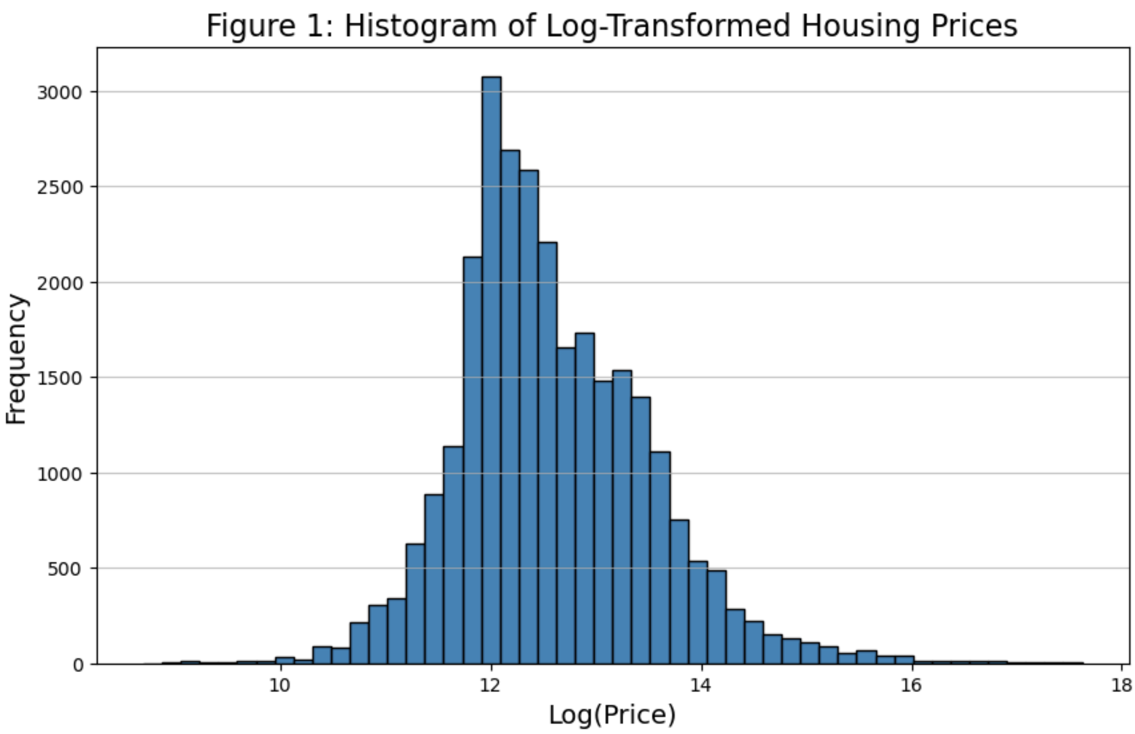
The Airbnb-Housing Ratio (AHR) shows the number of STRs per on-sale housing unit. Table 3 excludes rows with 'housing_count' or 'airbnb_count' values of 1 or 0 to focus on useful data. Queens has a very high AHR of 292.11, likely due to a data issue. Brooklyn also has a high AHR, suggesting many housing units or longterm rentals shift to STRs. In comparison, the Bronx shows a moderate impact, while Flushing and Staten Island have weaker STR effects.

Plots, Histograms, Figures

In this part, the study shows different plots and graphs, aiming to visualize variables and relationships between different variables, especially response variables and showing relationships with several other explanatory variables. We filter out boroughs that is statistical meaningless.

In [148…

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

# Load the dataset
file_path = r"C:\Users\User\Desktop\ECO225\Datasets\NYhousing.csv"
ny_housing = pd.read_csv(file_path)

# Check if the 'price' column exists
if 'price' in ny_housing.columns:
    # Apply log transformation to the 'price' column
    ny_housing['log_price'] = np.log1p(ny_housing['price'])  # log1p handles lo

    # Plot the histogram of the log-transformed prices
    plt.figure(figsize=(10, 6))
    plt.hist(ny_housing['log_price'].dropna(), bins=50, edgecolor='black', colo
    plt.title('Figure 1: Histogram of Log-Transformed Housing Prices', fontsize
    plt.xlabel('Log(Price)', fontsize=14)
    plt.ylabel('Frequency', fontsize=14)
    plt.grid(axis='y', alpha=0.75)
    plt.show()
else:
    print("The column 'price' does not exist in the dataset.")
```



The histogram of log-transformed housing prices shows a more balanced distribution, and a log transformation addresses the skewness caused by extremely high-priced properties. The majority of housing prices fall within the log range of 11 to 13, approximately 60,000 to 500,000 in actual prices.

In [257…

```python
# Boxplot of Log-Transformed Airbnb-to-Housing Ratio
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np

# Load the dataset
ratio_summary_path = r"C:\Users\User\Desktop\ECO225\Datasets\ratio_summary.csv'
ratio_summary = pd.read_csv(ratio_summary_path)

# Apply log transformation to the Airbnb-to-Housing Ratio
ratio_summary['log_airbnb_housing_ratio'] = np.log(ratio_summary['airbnb_housi

# Create the horizontal boxplot
plt.figure(figsize=(10, 6))
sns.boxplot(
    x=ratio_summary['log_airbnb_housing_ratio'],
    color='skyblue',  # Use an academic-friendly color
    linewidth=1.5
)

# Add labels and title
plt.title("Figure 2: Boxplot of Log-Transformed Airbnb-Housing-Ratio(AHR)", for
plt.xlabel("Log(AHR + 1)", fontsize=12)
plt.grid(axis='x', linestyle='--', alpha=0.7)
plt.tight_layout()

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

Figure 2: Boxplot of Log-Transformed Airbnb-Housing-Ratio(AHR)

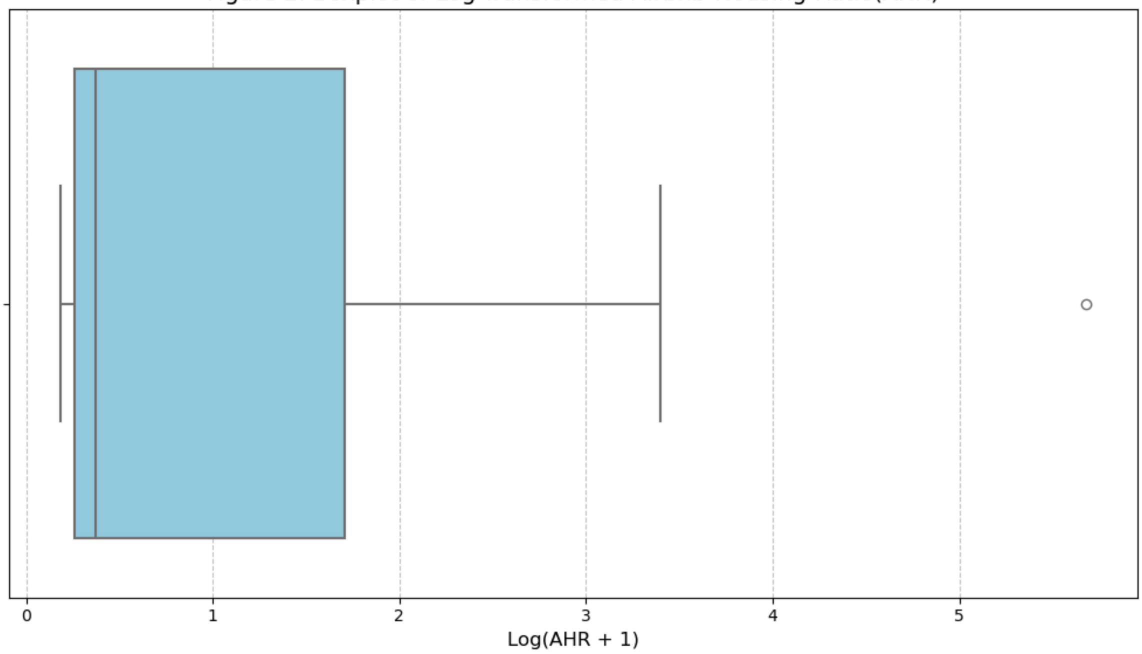


Figure 2 illustrates the distribution of Airbnb-Housing-Ratio (AHR), a key variable in this analysis. Due to an extreme AHR observed for Queens, a log transformation was applied to better represent the data. The x-axis, Log(AHR + 1), is the transformation of the original AHR values.

$$0 = \ln(AHR_1 + 1)$$

$$1 = \ln(AHR_2 + 1)$$

$$AHR_1 = e^0 - 1 = 1 - 1 = 0$$

$$AHR_2 = e^1 - 1 \approx 2.718 - 1 = 1.718$$

The boxplot is right-skewed. The median falls between 0 and 1 on the transformed scale, translating to an original AHR range of 0 to 1.718. Since the median is slightly closer to 0, this indicates that over 50% of boroughs in New York City have fewer than 0.859 STR units per on-sale housing unit.

In [259…

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np

# Step 1: Introduce the dataset
airbnb_path = r"C:\Users\User\Desktop\ECO225\Datasets\AB_NYC_2019_Cleaned.csv"
airbnb_data = pd.read_csv(airbnb_path)

# Step 2: Ensure 'price' is numeric and handle non-numeric or missing values
airbnb_data['price'] = pd.to_numeric(airbnb_data['price'], errors='coerce')  #
airbnb_data = airbnb_data.dropna(subset=['price'])  # Drop rows with NaN prices

# Step 3: Filter the Airbnb dataset for prices greater than or equal to $10
airbnb_data = airbnb_data[airbnb_data['price'] >= 10]

# Step 4: Apply log transformation to price
airbnb_data['log_price'] = np.log1p(airbnb_data['price'])  # Log(price + 1)

# Step 5: Plot the 2019 Airbnb log-transformed price distribution using a histo
plt.figure(figsize=(10, 6))
sns.histplot(
    airbnb_data['log_price'],
    kde=True,
    bins=50,
    color='skyblue',  # Use light green color for the bars
    edgecolor='black'  # Add black edges for better clarity
)

# Step 6: Add labels and title
plt.title("Figure 3: Histogram of Log-Transformed Airbnb Prices (2019)", fontsi
plt.xlabel("Log(Price + 1)", fontsize=12)
plt.ylabel("Frequency", fontsize=12)
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.tight_layout()

# Step 7: Display the plot
plt.show()
```

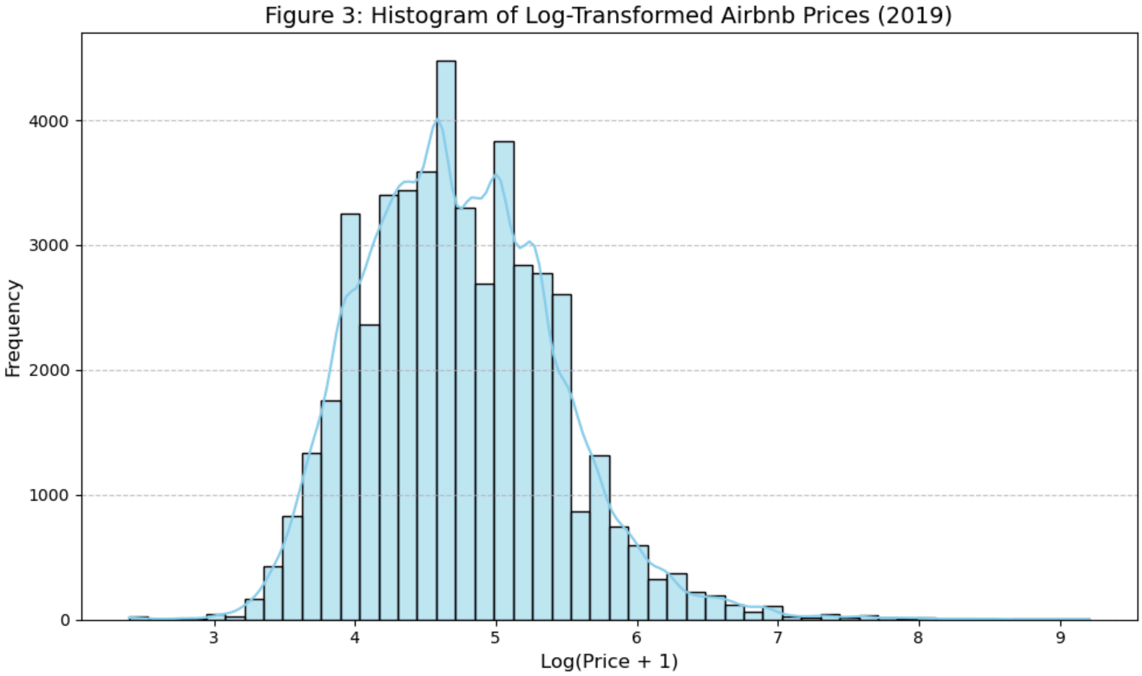


Figure 3 introduces the distribution on Airbnb prices. Since most STRs' prices are relatively low, and some STRs's price are very high, presenting the original histogram will be highly right skewed and effect our future analysis. Thus, a log transformation is applied to X-axis, which is similar to Figure 2. After the transformation, the distribution is slightly right skewed due to extreme high values at right. Most data lies on X=4 to X=5.5, which is between 53.60 and 245.97 in real price.

In [154…

```
# Violin Plot for Housing Price Distributions for selected Cities (Pre-2019 vs
import warnings
warnings.filterwarnings("ignore")  # Suppress all warnings

import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from matplotlib.ticker import FuncFormatter

# Load the dataset
housing_data_path = r"C:\Users\User\Desktop\ECO225\Datasets\NYhousing.csv"
housing_data = pd.read_csv(housing_data_path)

# Filter data for relevant columns and determine Pre-2019 and Post-2019 periods
housing_data['period'] = housing_data['prev_sold_date'].apply(
    lambda x: 'Pre-2019' if pd.to_datetime(x).year < 2019 else 'Post-2019'
)
housing_data = housing_data[['city', 'price', 'period']]

# Get the list of the 10 cities to plot
cities_to_plot = [
    'Brooklyn', 'Queens', 'Staten Island',
    'Bronx', 'Flushing', 'Forest Hills',
    'Fresh Meadows', 'Elmhurst'
]

# Create a 5x2 grid for subplots
```

```
fig, axes = plt.subplots(2, 4, figsize=(25, 18), sharey=True)

# Flatten the axes array for easy iteration
axes = axes.flatten()

# Define a formatter for the y-axis to show regular prices
def currency_format(x, _):
    return f'${int(x):,}'

formatter = FuncFormatter(currency_format)

# Loop through each city and plot its violin plot
for i, city in enumerate(cities_to_plot):
    city_data = housing_data[housing_data['city'] == city]
    sns.violinplot(
        data=city_data,
        x='period',
        y='price',
        order=['Pre-2019', 'Post-2019'],  # Ensure the correct order of x-axis
        scale='count',
        density_norm='count',  # Updated parameter
        ax=axes[i],
        palette={'Pre-2019': 'steelblue', 'Post-2019': 'lightgreen'}  # Updatec
    )
    axes[i].set_title(city, fontsize=12)
    axes[i].set_xlabel("")
    axes[i].set_ylabel("Housing Price (USD)")
    axes[i].grid(alpha=0.3)
    # Apply the formatter to each subplot's y-axis
    axes[i].yaxis.set_major_formatter(formatter)

# Hide any unused subplots if there are fewer than 10 cities
for j in range(len(cities_to_plot), len(axes)):
    fig.delaxes(axes[j])

# Adjust layout
fig.suptitle("Figure 4: Violin Plot for Housing Price Distributions for selecte
plt.tight_layout(rect=[0, 0, 1, 0.96])  # Adjust rect to fit title
plt.show()
```

Figure 4: Violin Plot for Housing Price Distributions for selected Cities (Pre-2019 vs Post-2019)

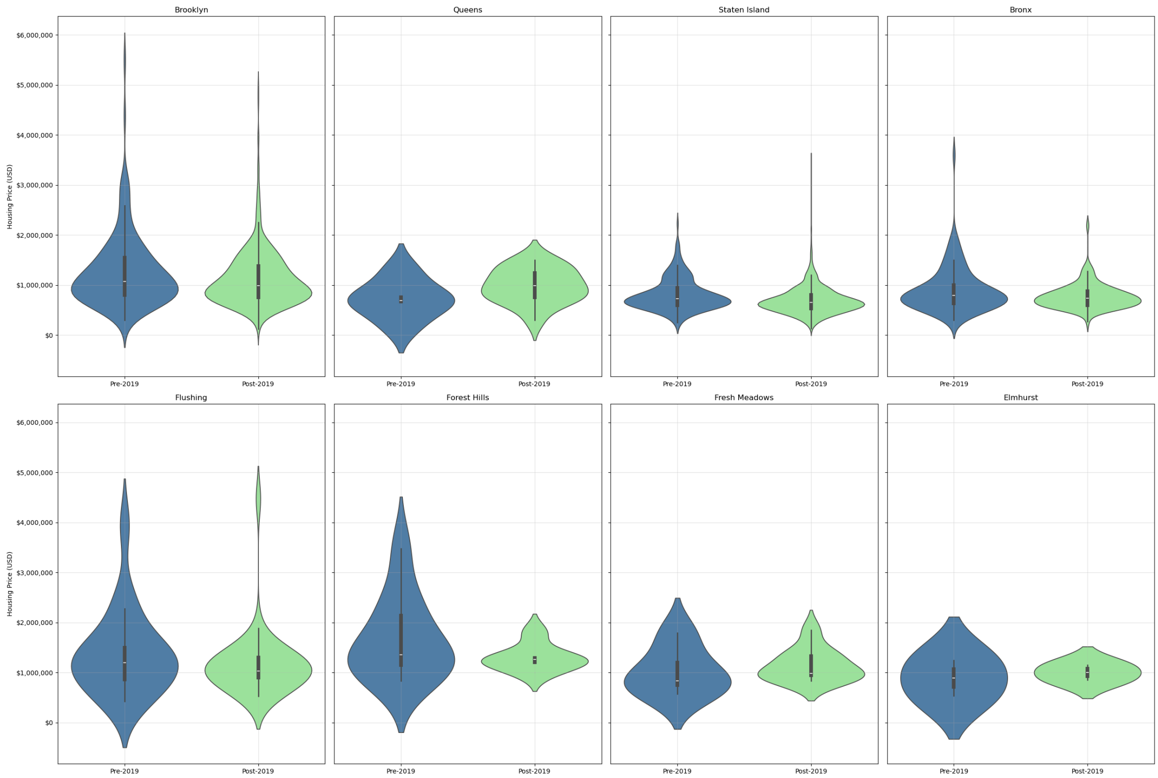


Figure 4 shows the housing price distribution for different boroughs using violin plots. A larger area indicates more housing availability. Most boroughs experienced a decrease in housing price range and total supply after 2019.

Bronx's price range shrink from around 4,000,000 dollars to 2,500,000 dollars, along with a decrease in total housing supply. Staten Island, however, showed an expanded range to 3,750,000 dollars, but these are limited to high-priced units, which have minimal impact on the overall market(Texas 2036, n.d).

Forest Hills was the most affected, with a significant drop in price range and housing supply. The number of homes sold decreased by 16.2%, from 99 in February 2023 to 83 in February 2024, according to Redfin.

In [157…

```
# Stacked Bar Plot of Log-Transformed Housing and Airbnb Counts by City
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np

# File paths for the datasets
ny_housing_path = r"C:\Users\User\Desktop\ECO225\Datasets\NYhousing.csv"
ratio_summary_path = r"C:\Users\User\Desktop\ECO225\Datasets\ratio_summary.csv"

# Step 1: Load the datasets
ny_housing = pd.read_csv(ny_housing_path)
ratio_summary = pd.read_csv(ratio_summary_path)

# Step 2: Prepare data for the plot
stacked_data = ratio_summary[['city', 'housing_count', 'airbnb_count']]
```

```
# Step 3: Calculate log-transformed counts
stacked_data['log_housing_count'] = np.log1p(stacked_data['housing_count'])  #
stacked_data['log_airbnb_count'] = np.log1p(stacked_data['airbnb_count'])  # Lo

# Step 4: Add total count (original, not log-transformed) for sorting
stacked_data['total_count'] = stacked_data['housing_count'] + stacked_data['ai

# Step 5: Sort data by total count (high to low)
stacked_data = stacked_data.sort_values(by='total_count', ascending=False)

# Step 6: Create the Stacked Bar Plot
plt.figure(figsize=(12, 8))
plt.bar(
    stacked_data['city'],
    stacked_data['log_housing_count'],
    color='steelblue',
    label='Log Housing Count',
    edgecolor='black'
)
plt.bar(
    stacked_data['city'],
    stacked_data['log_airbnb_count'],
    bottom=stacked_data['log_housing_count'],
    color='lightgreen',
    label='Log Airbnb Count',
    edgecolor='black'
)

# Step 7: Add labels and title
plt.title('Figure 5: Stacked Bar Plot of Log-Transformed Housing and Airbnb Cou
plt.xlabel('City', fontsize=12)
plt.ylabel('Log-Transformed Count', fontsize=12)
plt.xticks(rotation=45, ha='right', fontsize=10)
plt.legend(title='Count Type', fontsize=10)

# Step 8: Adjust layout and show the plot
plt.tight_layout()
plt.grid(axis='y', linestyle='--', alpha=0.5)
plt.show()
```

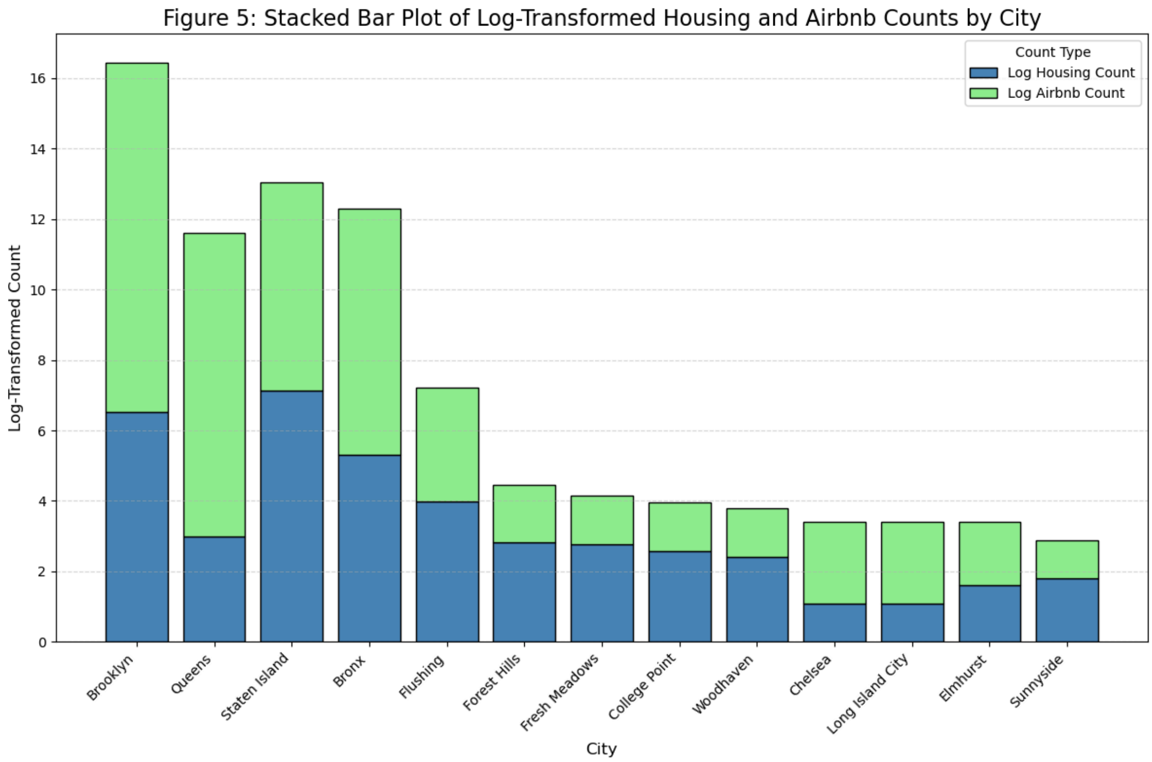
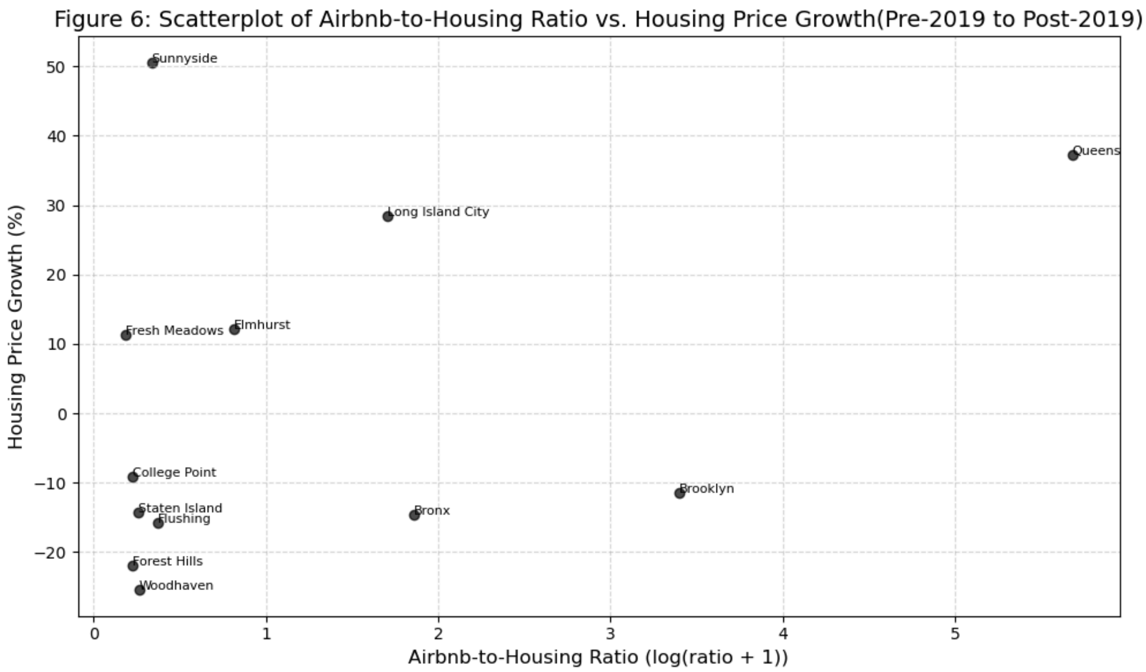


Figure 5 shows Airbnb and on-sale housing counts in a stacked bar plot. The y-axis uses log transformation to handle large differences in counts. Brooklyn, Queens, Staten Island, Bronx, and Flushing have the most housing resources. Brooklyn and Queens have more Airbnb listings, while Staten Island and Bronx rely less on Airbnb.

In [160…

```
# Scatterplot of Airbnb-to-Housing Ratio (2019) vs. Housing Price Growth(Pre-20
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np

# File paths for the datasets
ny_housing_path = r"C:\Users\User\Desktop\ECO225\Datasets\NYhousing.csv"
ratio_summary_path = r"C:\Users\User\Desktop\ECO225\Datasets\ratio_summary.csv"

# Step 1: Load the datasets
ny_housing = pd.read_csv(ny_housing_path, parse_dates=['prev_sold_date'])  # Pa
ratio_summary = pd.read_csv(ratio_summary_path)

# Step 2: Filter data for pre-2019 and post-2019 periods
pre_2019 = ny_housing[(ny_housing['prev_sold_date'] >= '2014-01-01') & (ny_hous
post_2019 = ny_housing[(ny_housing['prev_sold_date'] >= '2019-01-01') & (ny_hou

# Step 3: Calculate average housing prices by city for each period
avg_pre_2019 = pre_2019.groupby('city')['price'].mean().reset_index(name='avg_p
avg_post_2019 = post_2019.groupby('city')['price'].mean().reset_index(name='avg

# Step 4: Merge pre- and post-2019 averages
housing_price_growth = avg_pre_2019.merge(avg_post_2019, on='city')
housing_price_growth['price_growth'] = (
    (housing_price_growth['avg_price_post_2019'] - housing_price_growth['avg_pr
    housing_price_growth['avg_price_pre_2019']
) * 100  # Calculate percentage growth
```

```
# Step 5: Merge with Airbnb-to-Housing Ratio
merged_data = housing_price_growth.merge(ratio_summary[['city', 'airbnb_housing

# Step 6: Apply log transformation to the Airbnb-to-Housing Ratio
merged_data['log_airbnb_housing_ratio'] = np.log1p(merged_data['airbnb_housing_

# Step 7: Create the scatterplot
plt.figure(figsize=(10, 6))
plt.scatter(
    merged_data['log_airbnb_housing_ratio'],
    merged_data['price_growth'],
    alpha=0.7,
    color='black',
    edgecolor='black'
)
plt.title('Figure 6: Scatterplot of Airbnb-to-Housing Ratio vs. Housing Price (
plt.xlabel('Airbnb-to-Housing Ratio (log(ratio + 1))', fontsize=12)
plt.ylabel('Housing Price Growth (%)', fontsize=12)
plt.grid(True, linestyle='--', alpha=0.5)

# Annotate city names
for i, row in merged_data.iterrows():
    plt.annotate(row['city'], (row['log_airbnb_housing_ratio'], row['price_grow

plt.tight_layout()
plt.show()
```



In Figure 6, it construct a scatterplot between AHR and housing price growth. The price growth for different city is calculated as:

$$\text{Housing Price Growth } (\backslash\%) = \frac{A - B}{B} \times 100\%$$

Where:

$$A = \text{Average housing price post-2019}$$

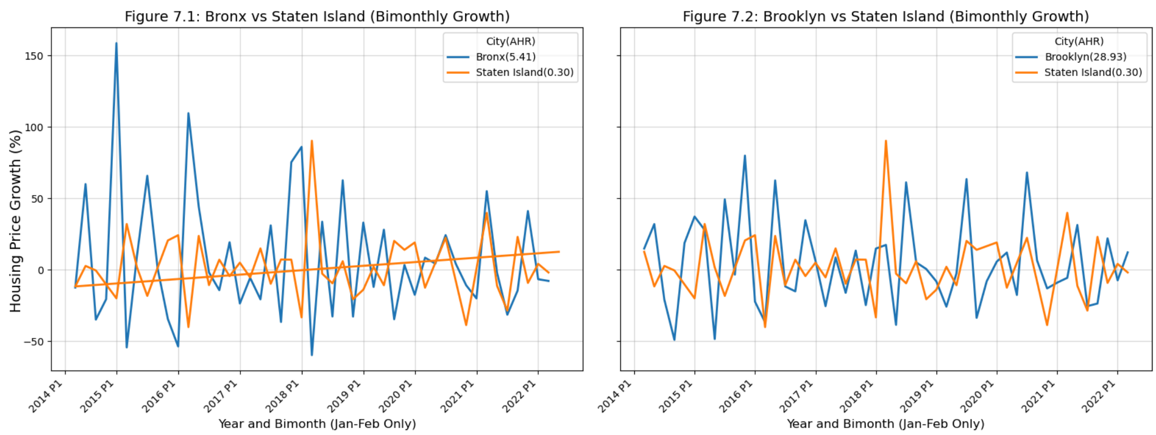
$$B = \text{Average housing price pre-2019}$$

According to Table 3 and Figure 6, Queens has the highest AHR but does not exhibit the highest housing price growth. Summyside, however, shows the highest housing price growth (50%) despite having a low AHR. Within the log-transformed range of 0 to 1, three cities show positive price growth, while five experience negative growth. Moreover, as seen with Bronx and Brooklyn, two high AHR boroughs does not have a high price growth. Therefore, no general pattern suggests a positive relationship between AHR and housing price growth rate.

In [55]:

```
import pandas as pd
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings("ignore")  # Suppress all warnings
# File paths for the datasets
ny_housing_path = r"C:\Users\User\Desktop\ECO225\Datasets\NYhousing.csv"

# Load the dataset
ny_housing = pd.read_csv(ny_housing_path)

# Filter data for Bronx, Brooklyn, and Staten Island
selected_cities = ['Bronx', 'Brooklyn', 'Staten Island']
filtered_data = ny_housing[ny_housing['city'].isin(selected_cities)]

# Extract year and month from `prev_sold_date`
filtered_data['prev_sold_date'] = pd.to_datetime(filtered_data['prev_sold_date
filtered_data['year'] = filtered_data['prev_sold_date'].dt.year
filtered_data['month'] = filtered_data['prev_sold_date'].dt.month

# Create a "bimonthly period" column (e.g., Jan-Feb is 1, Mar-Apr is 2, etc.)
filtered_data['bimonthly_period'] = (filtered_data['month'] - 1) // 2 + 1

# Group by year, bimonthly period, and city to calculate average housing price
price_trends_bimonthly = (
    filtered_data.groupby(['year', 'bimonthly_period', 'city'])
    .agg({'price': 'mean'})
    .reset_index()
)

# Calculate housing price growth (% change)
price_trends_bimonthly['price_growth'] = (
    price_trends_bimonthly.groupby('city')['price'].pct_change() * 100
)

# Combine year and bimonthly period for x-axis labels
price_trends_bimonthly['year_bimonth'] = (
    price_trends_bimonthly['year'].astype(str) + ' P' + price_trends_bimonthly
)

# Set up the figure and axes for 1x2 layout
fig, ax = plt.subplots(1, 2, figsize=(16, 6), sharey=True)

# Define cities to compare with Staten Island
comparison_cities = [('Bronx', '5.41'), ('Brooklyn', '28.93')]
titles = [
    'Figure 7.1: Bronx vs Staten Island (Bimonthly Growth)',
    'Figure 7.2: Brooklyn vs Staten Island (Bimonthly Growth)',
```

```
]

# Plot each city against Staten Island with solid lines
for i, (city, ahr) in enumerate(comparison_cities):
    city_data = price_trends_bimonthly[price_trends_bimonthly['city'] == city]
    staten_island_data = price_trends_bimonthly[price_trends_bimonthly['city']

    # Plot city data (solid line)
    ax[i].plot(
        city_data['year_bimonth'],
        city_data['price_growth'],
        label=f'{city}({ahr})',
        linestyle='-',
        linewidth=2
    )

    # Plot Staten Island data (solid line)
    ax[i].plot(
        staten_island_data['year_bimonth'],
        staten_island_data['price_growth'],
        label='Staten Island(0.30)',
        linestyle='-',
        linewidth=2
    )

    # Customize x-axis labels to only show January-February periods
    jan_feb_labels = city_data[city_data['bimonthly_period'] == 1]['year_bimont
    ax[i].set_xticks(jan_feb_labels)
    ax[i].set_xticklabels(jan_feb_labels, rotation=45, ha='right', fontsize=10

    ax[i].set_title(titles[i], fontsize=14)
    ax[i].set_xlabel('Year and Bimonth (Jan-Feb Only)', fontsize=12)
    ax[i].legend(title='City(AHR)')
    ax[i].grid(alpha=0.5)

# Add shared y-axis label
fig.supylabel('Housing Price Growth (%)', fontsize=14)
plt.tight_layout()

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



Figures 7.1 and 7.2 present bimonthly housing price trends for cities with large and small AHR values. Staten Island is used as a low AHR benchmark due to its

sufficient data. Queens could not be included in the long-term analysis due to dataset limitations, and an unintentional line in Figure 7.1 is not part of the plot.

In Figure 7.1, the Bronx shows high price variability, with peaks exceeding 150% growth and valleys below -50%. In Figure 7.2, Brooklyn displays similar volatility but with smaller peaks.

Compared to the Bronx and Brooklyn, Staten Island demonstrates a more stable growth trend, with only one peak below 100% and all other growth rates under 50%.

These graphs suggest a positive relationship between housing price variability and AHR in the selected cities, which may impact housing affordability and stability.

Conclusion

This study used the boroughs of New York City to analyze whether AHR (Airbnb-Housing Ratio) helps explain housing trends. AHR was found to be a useful metric for cities with sufficient Airbnb and on-sale housing data. The analysis explored the impact of STRs on housing affordability and stability. Figure 6 showed no clear relationship between AHR and housing price growth, likely due to limited data. However, Figures 7.1 and 7.2 revealed a potential relationship in the mentioned boroughs, where higher AHR is associated with greater price variability.

Limitations & Next Steps

Figure 6 did not show a clear relationship between AHR and housing price growth. This is because the scatterplot has too few data points.

Figures 7.1 and 7.2 also have limited value since they only compare three boroughs. According to Table 3, we need boroughs with enough Airbnb and on-sale housing data for long-term comparisons. However, only Bronx, Brooklyn, and Staten Island had sufficient data due to issues with the provided dataset.

The "USA Real Estate Dataset" uses 'state' and 'city' variables, while the "New York City Airbnb Open Data" uses 'neighborhood_group' for boroughs. Queens lacks sufficient on-sale housing data, leading to a high AHR. Additionally, the "city" variable mixes boroughs like Bronx and Brooklyn with cities like New York City, but excluding Manhattan etc. These issues caused confusion during the analysis.

After filtering, there were only a few observations left for the required boroughs.

This study shows AHR helps analyze STRs and on-sale housing in NYC. Future work should include more data and more cities to improve comparisons between cities and better understand AHR's effect on housing prices.

Citations

AirDNA. (2023). Why Airbnb’s 2023 summer release focuses on private rooms. AirDNA Blog. Retrieved from https://www.airdna.co/blog/why-airbnbs-2023-summer-release-focuses-on-private-rooms

Barron, K., Kung, E., & Proserpio, D. (2019). The effect of home-sharing on house prices and rents: Evidence from Airbnb. Marketing Science, 40(1), 23-47. https://doi.org/10.1287/mksc.2020.1232

Federal Reserve Bank of St. Louis. (n.d.). Median sales price of houses sold for the United States [Data set]. FRED. Retrieved February 6, 2025, from https://fred.stlouisfed.org/series/MSPUS

Filippas, A., & Horton, J. J. (2023). The tragedy of your upstairs neighbors: The negative externalities of home-sharing platforms. Retrieved from https://apostolos-filippas.com/papers/airbnb.pdf

Gomonov, D. (n.d.). New York City Airbnb Open Data [Data set]. Kaggle. Retrieved from https://www.kaggle.com/datasets/dgomonov/new-york-city-airbnb-open-data

Kluyver, T., Ragan-Kelley, B., Pérez, F., Granger, B., Bussonnier, M., Frederic, J., Kelley, K., Hamrick, J., Grout, J., Corlay, S., Ivanov, P., Avila, D., Abdalla, S., & Willing, C. (2016). Jupyter Notebooks – A publishing format for reproducible computational workflows. In F. Loizides & B. Schmidt (Eds.), Positioning and Power in Academic Publishing: Players, Agents and Agendas (pp. 87–90). IOS Press. Retrieved from https://jupyter.org

Koster, H. R. A., van Ommeren, J., & Volkhausen, N. (2024). The impact of short-term rental activity on house prices. Annals of Regional Science. https://doi.org/10.1007/s00168-024-01122-3

Lanfear, C., & Kirk, D. (2024). The promise and perils of the sharing economy: The impact of Airbnb lettings on crime. Criminology, 62(1), 123-145. Retrieved from https://onlinelibrary.wiley.com/doi/10.1111/1745-9125.12383

Python Software Foundation. (2023). Python programming language (Version 3.x) [Software]. Retrieved from https://www.python.org

Redfin. (2024). Forest Hills Housing Market Trends. Retrieved from https://www.redfin.com/neighborhood/30574/NY/New-York/Forest-Hills/housing-market

Sakib, A. S. (n.d.). USA Real Estate Dataset [Data set]. Kaggle. Retrieved from https://www.kaggle.com/datasets/ahmedshahriarsakib/usa-real-estate-dataset

Texas 2036. (n.d.). Are institutions to blame for high housing prices? Retrieved from https://texas2036.org/posts/are-institutions-to-blame-for-high-housing-prices/

Trojanek, R. (2023). How do different noise pollution sources affect apartment prices? International Journal of Strategic Property Management, 27(6), 351–361. https://doi.org/10.3846/ijspm.2023.20563

Wachsmuth, D., Kerrigan, D., Chaney, D., & Shillolo, A. (2018). The high cost of short-term rentals in New York City. McGill University, Urban Politics and Governance Research Group. Retrieved from https://www.mcgill.ca/newsroom/files/newsroom/channels/attach/airbnb-report.pdf

In [ ]: