Screencast :

https://drive.google.com/file/d/1zg_uEHCkrVQMac92aMm5jAEuZMD31zSw/view?usp=drive_link

In [ ]:
```
import numpy as np
import arviz as az
import pymc as pm
from scipy import stats
import seaborn as sns
import matplotlib.pyplot as plt
import pandas as pd
import pytensor.tensor as pt
import warnings
warnings.filterwarnings('ignore')
```

```
WARNING (pytensor.tensor.blas): Using NumPy C-API based implementation for BLAS funct
ions.
```

In [ ]:
```
def standardize(series):
    return (series - series.mean()) / series.std()
```

Introduction

- The dataset used for this project, **USA Real Estate Dataset**, consists of **2,226,382** entries with 10 columns, including information on housing prices, property characteristics, and location details. Key variables include:

1. **Price**: Listing or recently sold price
2. **House Size**: Living space in square feet
3. **City**: Geographic location

- For privacy, broker and address information are categorically encoded. While the dataset originally contains extensive features such as land size, number of bedrooms, and postal codes, we filtered the data to focus on price, house size, and city to estimate the causal relationship between house sizes and prices, accounting for City as a confounder.

- Dataset Link: https://www.kaggle.com/datasets/ahmedshahriarsakib/usa-real-estate-dataset *(Dataset is too huge)*

Estimand/Question:

- The goal of this project is to utilize statistical methods to infer the relationship between **House Prices** and **House Size**, with **City** acting as a confounder, using the US Real Estate Prices dataset. By leveraging the concepts taught in class, we aim to rigorously estimate the causal effect of house size on housing prices while accounting for geographic and market-specific variations.

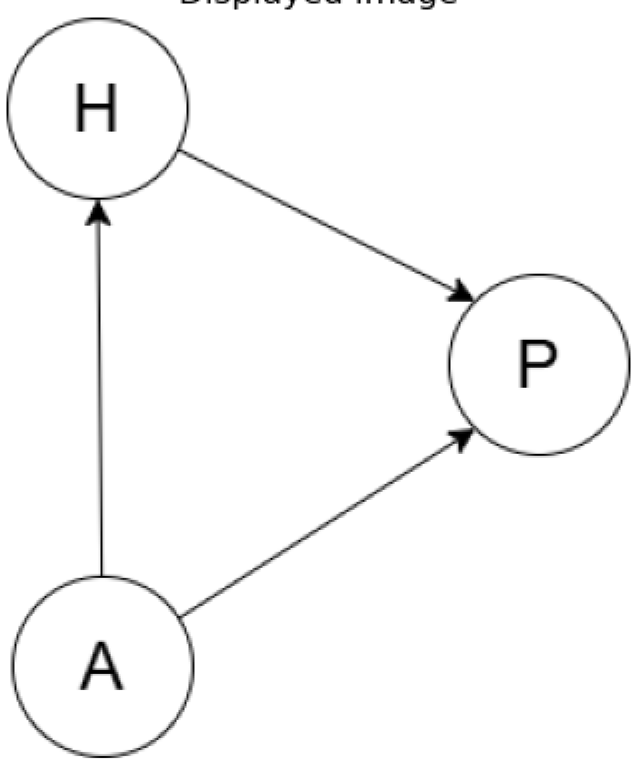
- This project demonstrates the practical application of learned methodologies to address real-world challenges in housing market analysis

Causal Model

- For the above-mentioned dataset, the DAG which can model our Estimand, and the Treatment-Outcome-Confounder variables is illustrated below:
  - House size (H) is assumed to have a direct causal effect on Price (P).
  - City (A) is assumed to affect both House size (H) and Price (P) directly.

In [ ]:
```
from PIL import Image
import matplotlib.pyplot as plt

# Specify the path to your image
image_path = "./PP.png"

# Open the image using PIL
image = Image.open(image_path)

# Display the image using Matplotlib
plt.imshow(image)
plt.axis('off')  # Hide axes for better display
plt.title("Displayed Image")
plt.show()
```

Displayed Image



Data Import and Pre-Processing

In [ ]:
```
df = pd.read_csv("realtor-data.csv")
df
```

Out[ ]:

	brokered_by	status	price	bed	bath	acre_lot	street	city	state	zip_(
0	103378.0	for_sale	105000.0	3.0	2.0	0.12	1962661.0	Adjuntas	Puerto Rico	6
1	52707.0	for_sale	80000.0	4.0	2.0	0.08	1902874.0	Adjuntas	Puerto Rico	6
2	103379.0	for_sale	67000.0	2.0	1.0	0.15	1404990.0	Juana Diaz	Puerto Rico	7
3	31239.0	for_sale	145000.0	4.0	2.0	0.10	1947675.0	Ponce	Puerto Rico	7
4	34632.0	for_sale	65000.0	6.0	2.0	0.05	331151.0	Mayaguez	Puerto Rico	6
...	...	...	...	...	...	...	...	...	...	
2226377	23009.0	sold	359900.0	4.0	2.0	0.33	353094.0	Richland	Washington	993
2226378	18208.0	sold	350000.0	3.0	2.0	0.10	1062149.0	Richland	Washington	993
2226379	76856.0	sold	440000.0	6.0	3.0	0.50	405677.0	Richland	Washington	993
2226380	53618.0	sold	179900.0	2.0	1.0	0.09	761379.0	Richland	Washington	993
2226381	108243.0	sold	580000.0	5.0	3.0	0.31	307704.0	Richland	Washington	993

2226382 rows × 12 columns

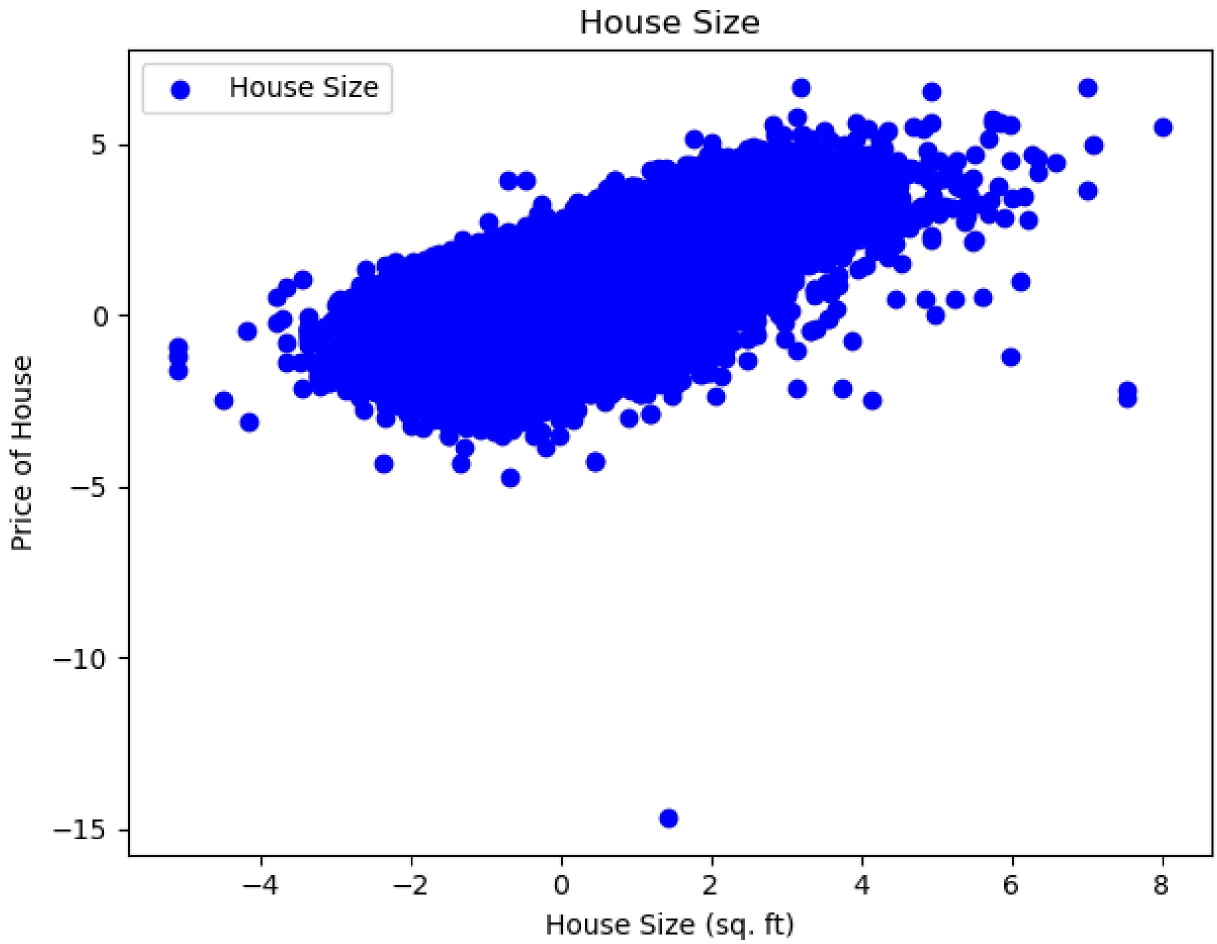
In [ ]:

```
filtered_df = df[['price', 'city', 'house_size']]
filtered_df = filtered_df.dropna()

# Nnumber of Cities to Confound On
n_cities = 10
top_cities = filtered_df['city'].value_counts().head(n_cities).reset_index()
print(top_cities)
top_cities = top_cities['city']

filtered_df = filtered_df[filtered_df['city'].isin(top_cities)]
filtered_df = filtered_df[filtered_df['house_size'] != 1560780.0]

print(filtered_df) # -> Final Data
```

```
          city  count
0        Houston  21700
1        Chicago  12081
2   Jacksonville  10856
3   Philadelphia   9587
4          Miami   9097
5    Los Angeles   8372
6         Tucson   8350
7  New York City   8179
8        Phoenix   7873
9         Dallas   7832
              price           city  house_size
45719     3600000.0  New York City      2338.0
45720    22000000.0  New York City      7020.0
45736     1250000.0  New York City       568.0
45738      395000.0  New York City       650.0
45740     8625000.0  New York City      2617.0
...             ...            ...         ...
2188864     92500.0         Dallas      1082.0
2188884    349900.0         Dallas      1472.0
2188887    420000.0         Dallas      1413.0
2188888    445504.0         Dallas      1440.0
2188907    380000.0         Dallas      1382.0

[103926 rows x 3 columns]
```

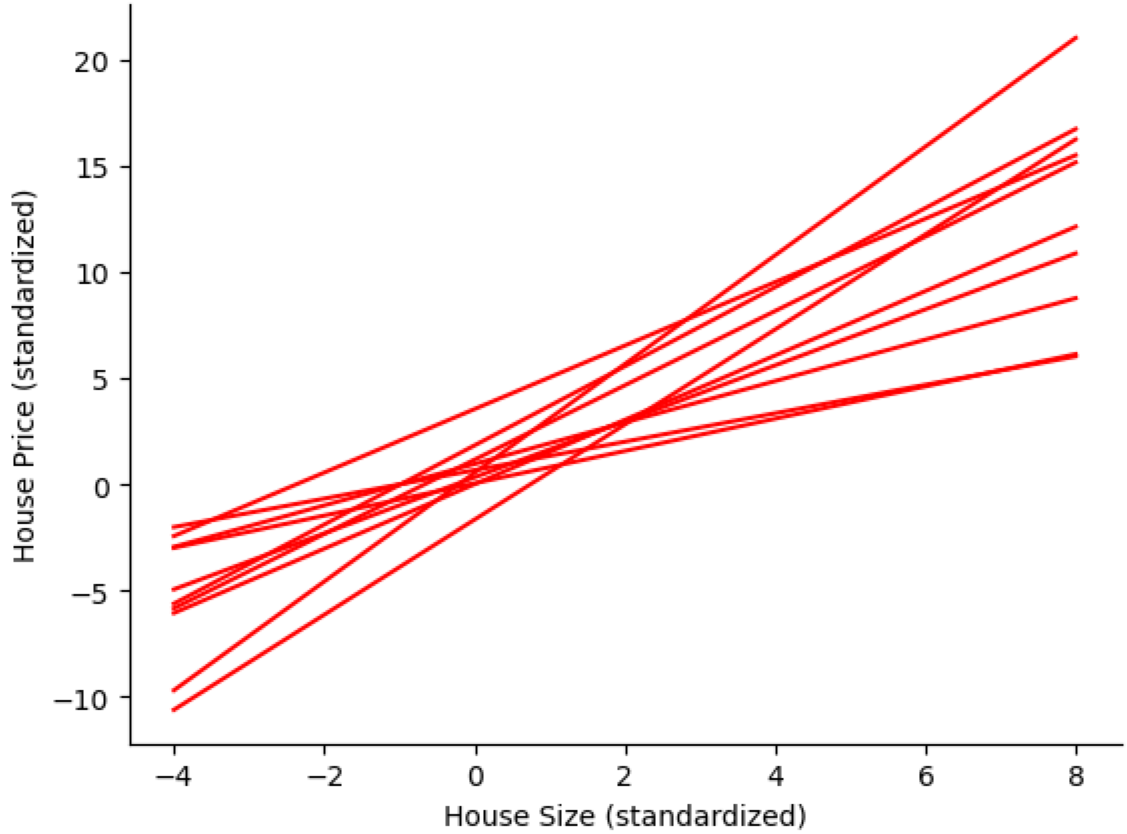
In [ ]:

```
house_size_standardized = standardize(np.log(filtered_df['house_size']))
price_standardized = standardize(np.log(filtered_df['price']))
city = pd.Categorical(filtered_df['city']).codes
city_categories = pd.Categorical(filtered_df['city']).categories
plt.figure(figsize=(12, 5))

# Scatter plot for house_size
plt.subplot(1, 2, 1)
plt.scatter(house_size_standardized, price_standardized, color='blue', label='House Si
plt.title('House Size')
plt.xlabel('House Size (sq. ft)')
plt.ylabel('Price of House')
plt.legend()
# Display the plots
plt.tight_layout()
plt.show()
```



Prior Predective Simulation

In [ ]:
```
# prior predictive simulation
n = 10
a = stats.norm.rvs(1, 1.5, size=n)
b_a = stats.norm.rvs(1, 1, size=n)
house_price_seq = np.linspace(-4, 8, 40)

for i in range(n):
    mu = a[i] + b_a[i] * house_price_seq
    plt.plot(house_price_seq, mu, color='r')


plt.xlabel("House Size (standardized)")
plt.ylabel("House Price (standardized)")
sns.despine()
```



Statistical Model

- **Intercept (alpha)**:
  - A Normal distribution with a mean of 1 and a standard deviation of 1.5 was used. This choice allows flexibility in capturing varying intercepts for different cities.
  - alpha $\sim$ Normal(1, 1.5)
- **Slope (beta)**:
  - A LogNormal distribution ensures positive slopes, as it is expected that house prices increase with house size. This prior also accommodates variability.
  - beta $\sim$ LogNormal(1, 1)
- **Standard Deviation (sigma)**:
  - An Exponential prior constrains the standard deviation to positive values, reflecting the non-negative nature of variability in housing prices.
  - sigma $\sim$ Exponential(1)
- These priors are validated through prior predictive simulation above.

Outcome Variable

- The outcome variable, price, is modeled using a Normal distribution:
- This is appropriate for the following reasons:

  - Housing prices are continuous, and their standardized form typically follows a roughly normal distribution.
  - The standard deviation parameter (sigma) accounts for variability in prices across different cities and house sizes.
  - The statistical model aligns well with the causal model (illustrated in the DAG) by accounting for the relationships between variables:
  - This is explicitly incorporated into the model:
    - mu = alpha[city] + beta[city] * house_size_standardized
- This ensures the causal relationship between house size and price is estimated correctly, while adjusting for the effects of the city.

Mathematical Notation

$mu = \alpha_{C[i]} + \beta_{C[i]} \cdot H$
$\sigma \sim Exponential(1)$
$P \sim \mathcal{N}(mu, \sigma)$

Handling City as a Confounder

Categorical Encoding:

- The city variable is encoded to represent city-level effects in the model.

Separate parameters are defined for each city's intercept (alpha) and slope (beta):

- alpha = pm.Normal('alpha', mu=1, sigma=1.5, shape=n_cities)
- beta = pm.LogNormal('beta', mu=1, sigma=1, shape=n_cities)

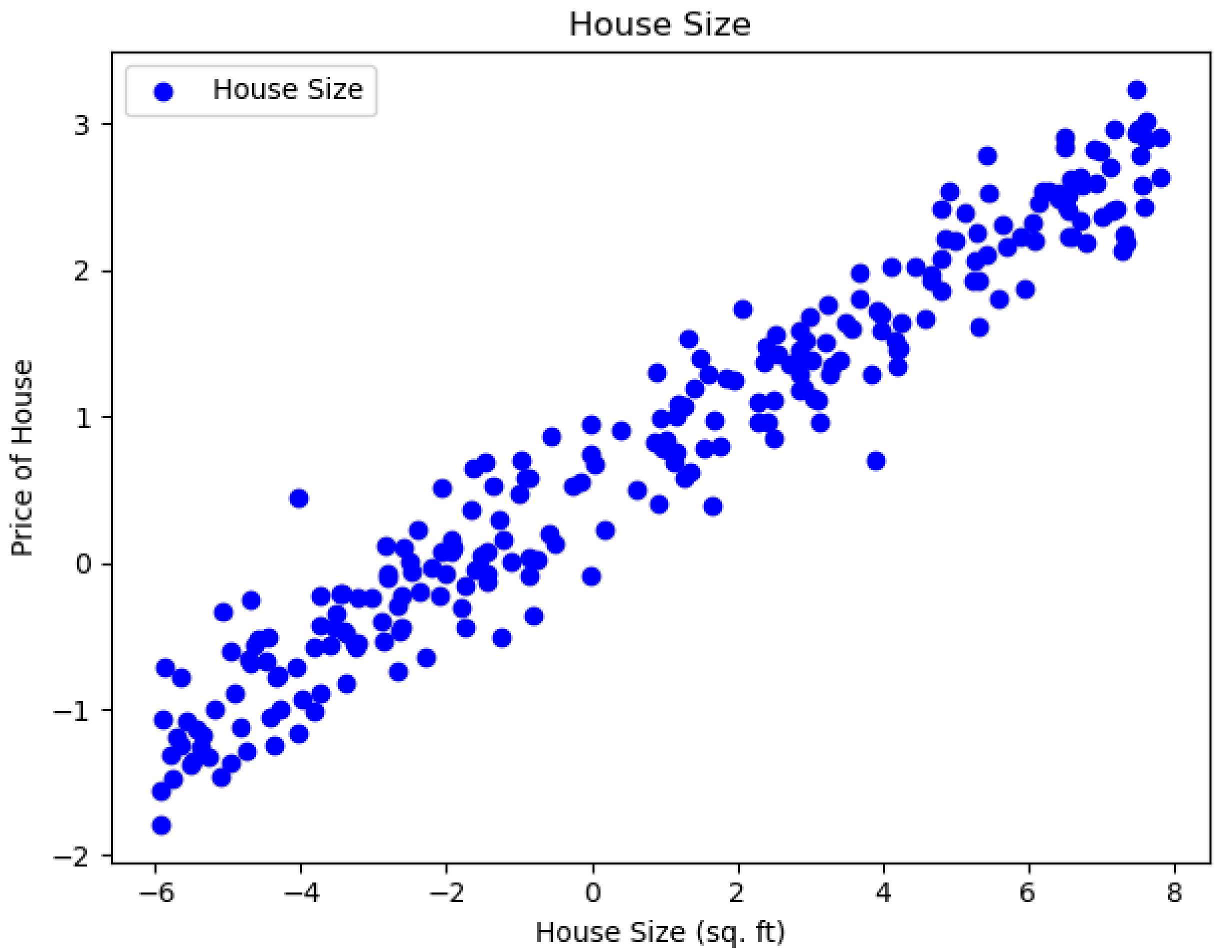
This accounts for the variability in price and house size due to market conditions in different cities, reducing confounding bias.

Model Validation

- The model is validated using synthetic data, where the parameter values (alpha, beta, sigma) are predefined:
  - alpha = 0.5
  - beta = 0.3
  - sigma = 0.3
- 250 House size are extracted from a unniform distribution.
- The House Prices are calcualted using the statistical model defined above.
- The posterior estimates of these parameters are compared with their true values.

Model Testing on Simulated Data

In [ ]:
```
np.random.seed(42)
n_samples = 250  # Number of data points per city

# Pre-defined parameters
true_alpha = 0.5  # True intercepts for each city
true_beta = 0.3     # True slopes for each city
true_sigma = 0.3

# Generate synthetic data
simulated_house_size = stats.uniform.rvs(-6, 14, n_samples)
mu = true_alpha + true_beta * simulated_house_size
simulated_price_standardized = stats.norm.rvs(mu, true_sigma, n_samples)


# Create a DataFrame for easier manipulation
simulated_data = pd.DataFrame({
    'simulated_house_size': simulated_house_size,
    'simulated_price_standardized': simulated_price_standardized
})

plt.figure(figsize=(12, 5))

# Scatter plot for house_size
plt.subplot(1, 2, 1)
plt.scatter(simulated_data['simulated_house_size'], simulated_data['simulated_price_st
plt.title('House Size')
plt.xlabel('House Size (sq. ft)')
plt.ylabel('Price of House')
plt.legend()

# Display the plots
plt.tight_layout()
plt.show()
```



In [ ]:
```
# PyMC model for simulated data
with pm.Model() as simulated_model:

    alpha = pm.Normal('alpha', mu=1, sigma=1.5)
    beta = pm.Normal('beta', mu=1, sigma=1)
    sigma = pm.Exponential('sigma', 1)

    # Linear model
    mu = alpha + beta * simulated_data['simulated_house_size']

    # Likelihood
    pred = pm.Normal('p', mu=mu, sigma=sigma, observed=simulated_data['simulated_price

    # Inference
    simulated_idata= pm.sample(idata_kwargs={"log_likelihood": True})
```

```
Auto-assigning NUTS sampler...
Initializing NUTS using jitter+adapt_diag...
Multiprocess sampling (4 chains in 4 jobs)
NUTS: [alpha, beta, sigma]
Output()
```

```
Sampling 4 chains for 1_000 tune and 1_000 draw iterations (4_000 + 4_000 draws tota
l) took 13 seconds.
```

In [ ]:
```
az.summary(simulated_idata)
# true_alpha = 0.5
```

```
# true_beta = 0.3
# true_sigma = 0.3
```

Out[ ]:

	mean	sd	hdi_3%	hdi_97%	mcse_mean	mcse_sd	ess_bulk	ess_tail	r_hat
alpha	0.524	0.020	0.487	0.561	0.0	0.0	5333.0	3182.0	1.0
beta	0.295	0.005	0.286	0.304	0.0	0.0	5706.0	3026.0	1.0
sigma	0.300	0.014	0.272	0.324	0.0	0.0	6052.0	3517.0	1.0

Posterior Model

In [ ]:

```
with pm.Model() as model:
    a = pm.Normal('a', mu = 1, sigma = 1.5, shape = n_cities)
    b = pm.LogNormal('b', mu = 1, sigma = 1, shape = n_cities)
    sigma = pm.Exponential('sigma', 1)
    mu = a[city] + b[city] * house_size_standardized
    pred_price = pm.Normal('p', mu = mu, sigma = sigma, observed = price_standardized)

    idata = pm.sample(idata_kwargs={"log_likelihood": True})
```

```
Auto-assigning NUTS sampler...
Initializing NUTS using jitter+adapt_diag...
Multiprocess sampling (4 chains in 4 jobs)
NUTS: [a, b, sigma]
Output()
```

```
Sampling 4 chains for 1_000 tune and 1_000 draw iterations (4_000 + 4_000 draws tota
l) took 72 seconds.
```

In [ ]:

```
az.summary(idata)
```

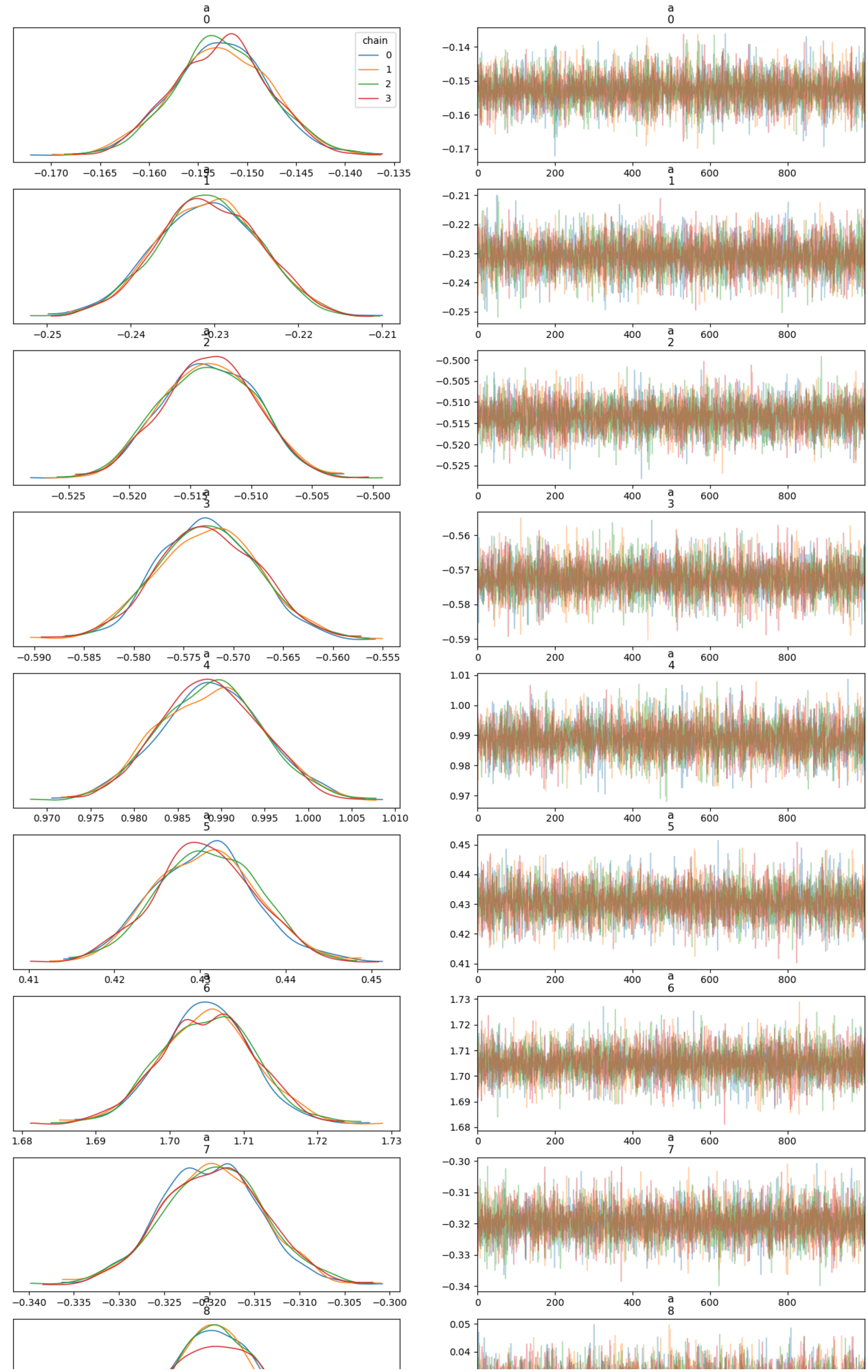
Out[ ]:

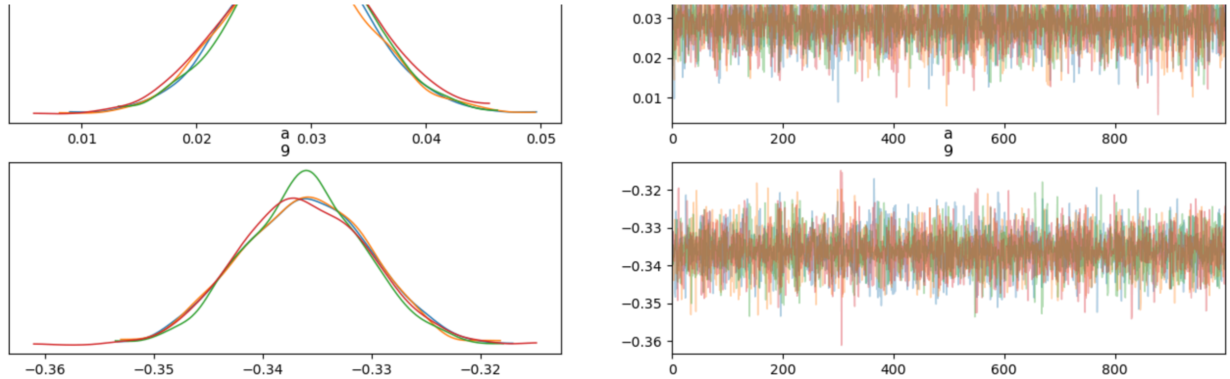
	mean	sd	hdi_3%	hdi_97%	mcse_mean	mcse_sd	ess_bulk	ess_tail	r_hat
a[0]	-0.153	0.005	-0.162	-0.144	0.0	0.0	10693.0	3023.0	1.0
a[1]	-0.231	0.006	-0.242	-0.219	0.0	0.0	9671.0	3014.0	1.0
a[2]	-0.513	0.004	-0.521	-0.507	0.0	0.0	8364.0	3202.0	1.0
a[3]	-0.573	0.005	-0.583	-0.563	0.0	0.0	8494.0	2535.0	1.0
a[4]	0.989	0.006	0.978	1.001	0.0	0.0	8588.0	3280.0	1.0
a[5]	0.431	0.006	0.420	0.442	0.0	0.0	9101.0	3357.0	1.0
a[6]	1.705	0.006	1.693	1.717	0.0	0.0	9568.0	3200.0	1.0
a[7]	-0.320	0.006	-0.330	-0.309	0.0	0.0	8329.0	3286.0	1.0
a[8]	0.029	0.006	0.018	0.040	0.0	0.0	10751.0	2714.0	1.0
a[9]	-0.336	0.006	-0.346	-0.325	0.0	0.0	9911.0	3307.0	1.0
b[0]	0.605	0.005	0.596	0.614	0.0	0.0	8872.0	2985.0	1.0
b[1]	0.700	0.006	0.689	0.712	0.0	0.0	8859.0	2924.0	1.0
b[2]	0.690	0.004	0.682	0.697	0.0	0.0	7729.0	3460.0	1.0
b[3]	0.665	0.007	0.652	0.677	0.0	0.0	10106.0	2862.0	1.0
b[4]	0.516	0.005	0.507	0.524	0.0	0.0	8512.0	3216.0	1.0
b[5]	0.605	0.006	0.594	0.616	0.0	0.0	8331.0	2805.0	1.0
b[6]	0.644	0.004	0.635	0.651	0.0	0.0	8722.0	3085.0	1.0
b[7]	0.573	0.006	0.560	0.585	0.0	0.0	8637.0	2981.0	1.0
b[8]	0.573	0.007	0.560	0.586	0.0	0.0	10446.0	3195.0	1.0
b[9]	0.599	0.008	0.585	0.614	0.0	0.0	9251.0	2817.0	1.0
sigma	0.530	0.001	0.528	0.532	0.0	0.0	8501.0	2453.0	1.0

In [ ]:

```
az.plot_trace(idata, var_names=["a"], legend=True, figsize=(16,30), compact=False)
```

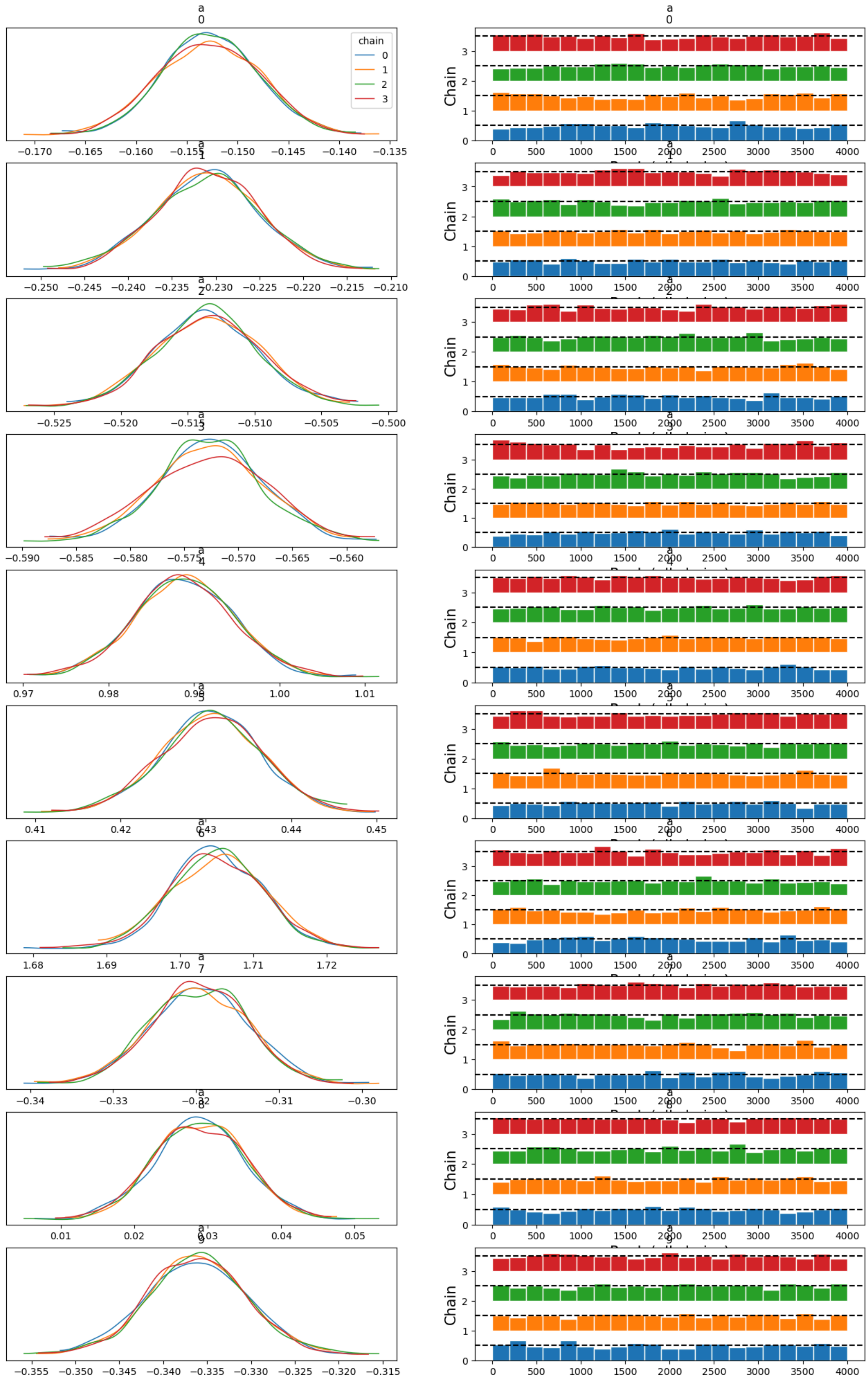
Out[ ]:
```
array([[<Axes: title={'center': 'a\n0'}>,
        <Axes: title={'center': 'a\n0'}>],
       [<Axes: title={'center': 'a\n1'}>,
        <Axes: title={'center': 'a\n1'}>],
       [<Axes: title={'center': 'a\n2'}>,
        <Axes: title={'center': 'a\n2'}>],
       [<Axes: title={'center': 'a\n3'}>,
        <Axes: title={'center': 'a\n3'}>],
       [<Axes: title={'center': 'a\n4'}>,
        <Axes: title={'center': 'a\n4'}>],
       [<Axes: title={'center': 'a\n5'}>,
        <Axes: title={'center': 'a\n5'}>],
       [<Axes: title={'center': 'a\n6'}>,
        <Axes: title={'center': 'a\n6'}>],
       [<Axes: title={'center': 'a\n7'}>,
        <Axes: title={'center': 'a\n7'}>],
       [<Axes: title={'center': 'a\n8'}>,
        <Axes: title={'center': 'a\n8'}>],
       [<Axes: title={'center': 'a\n9'}>,
        <Axes: title={'center': 'a\n9'}>]], dtype=object)
```


a
0
chain
0
1
2
3
a
1
a
2
a
3
a
4
a
5
a
6
a
7
a
8



In [ ]:
```
az.plot_trace(idata, var_names=["a"], legend=True, compact=False, kind="rank_bars", fi
```

Out[ ]:
```
array([[<Axes: title={'center': 'a\n0'}>,
        <Axes: title={'center': 'a\n0'}, xlabel='Rank (all chains)', ylabel='Chai
n'>],
       [<Axes: title={'center': 'a\n1'}>,
        <Axes: title={'center': 'a\n1'}, xlabel='Rank (all chains)', ylabel='Chai
n'>],
       [<Axes: title={'center': 'a\n2'}>,
        <Axes: title={'center': 'a\n2'}, xlabel='Rank (all chains)', ylabel='Chai
n'>],
       [<Axes: title={'center': 'a\n3'}>,
        <Axes: title={'center': 'a\n3'}, xlabel='Rank (all chains)', ylabel='Chai
n'>],
       [<Axes: title={'center': 'a\n4'}>,
        <Axes: title={'center': 'a\n4'}, xlabel='Rank (all chains)', ylabel='Chai
n'>],
       [<Axes: title={'center': 'a\n5'}>,
        <Axes: title={'center': 'a\n5'}, xlabel='Rank (all chains)', ylabel='Chai
n'>],
       [<Axes: title={'center': 'a\n6'}>,
        <Axes: title={'center': 'a\n6'}, xlabel='Rank (all chains)', ylabel='Chai
n'>],
       [<Axes: title={'center': 'a\n7'}>,
        <Axes: title={'center': 'a\n7'}, xlabel='Rank (all chains)', ylabel='Chai
n'>],
       [<Axes: title={'center': 'a\n8'}>,
        <Axes: title={'center': 'a\n8'}, xlabel='Rank (all chains)', ylabel='Chai
n'>],
       [<Axes: title={'center': 'a\n9'}>,
        <Axes: title={'center': 'a\n9'}, xlabel='Rank (all chains)', ylabel='Chai
n'>]],
      dtype=object)
```


a
0
chain
0
1
2
3
Chain
a
1
a
2
a
3
a
4
a
5
a
6
a
7
a
8
a
9

Rank (all chains)

Model Results Characteristics

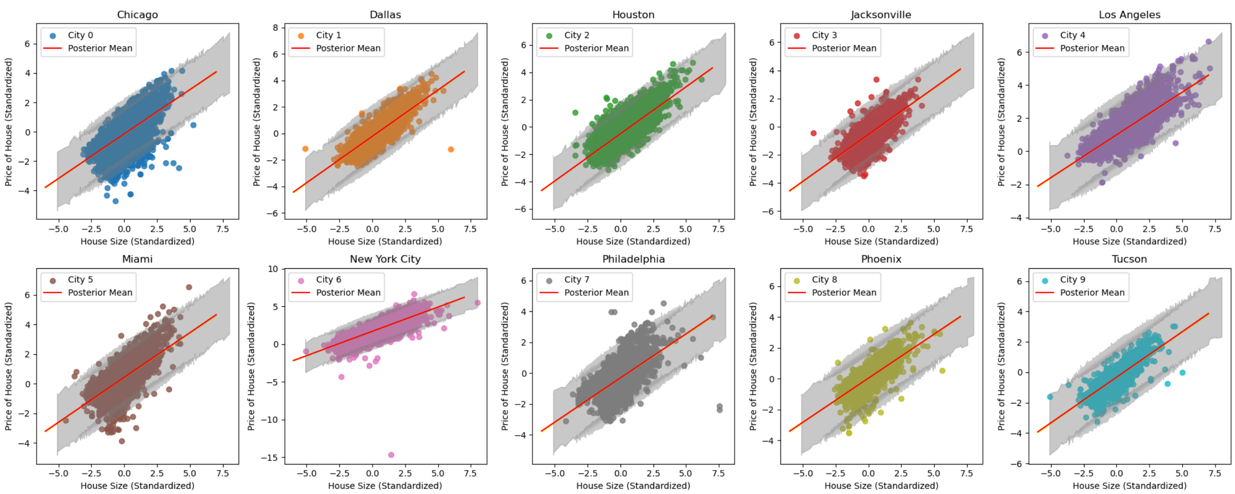
- The Chains are stationary and stay within high-probablility region of distribution.
- The Chain explores the full region of posterior from beginning of the trace.
- The Chains stay in the same high probability region.
- In the bar plots, The distribution of the bar plots is relatively uniform.
- Almost all parameters have a very high estimate of number of effective samples.

Posterior Predective Check

In [ ]:
```
size_sequence = np.arange(-6, 8)
idata_thinned = az.extract(idata, num_samples= 1000)

mu_pred = np.zeros((n_cities, len(size_sequence), idata_thinned.sizes['sample']))
mu_ob = np.zeros((n_cities, len(house_size_standardized), idata_thinned.sizes['sample'

a_values = idata_thinned.a.values
b_values = idata_thinned.b.values

for i, h in enumerate(size_sequence):
    for j in range(10):
        mu_pred[j][i] = a_values[j] + b_values[j] * h

for i, h in enumerate(house_size_standardized):
    for j in range(10):
        mu_ob[j][i] = stats.norm.rvs(a_values[j] + b_values[j] * h)

# mu_mean = mu_pred.mean(1)
```

In [ ]:
```
unique_cities = np.unique(city)
n_cities = len(unique_cities)

rows = 2
cols = 5

plt.figure(figsize=(20, 8))

for i, city_code in enumerate(unique_cities):

    city_mask = city == city_code
    x_city = house_size_standardized[city_mask]
    y_city = price_standardized[city_mask]

    city_mean = mu_pred[city_code].mean(axis=-1)

    plt.subplot(rows, cols, i + 1)
    plt.scatter(x_city, y_city, color=plt.cm.tab10(city_code), label=f'City {city_code

    plt.plot(size_sequence, city_mean, color="red", label="Posterior Mean")
    az.plot_hdi(
        size_sequence,
        mu_pred[city_code].T,
        hdi_prob=0.95,
```

```
        color='yellow',
        fill_kwargs={"alpha": 0.4},
        smooth=False)
    az.plot_hdi(
        house_size_standardized,
        mu_ob[city_code].T,  # Transpose to match (samples, sequence) structure
        hdi_prob=0.95,
        color="gray",
        fill_kwargs={"alpha": 0.4},
        smooth=False
    )

    plt.title(f'{city_categories[city_code]}')
    plt.xlabel('House Size (Standardized)')
    plt.ylabel('Price of House (Standardized)')
    plt.legend()

# Adjust layout
plt.tight_layout()
plt.show()
```



Inference

1. The above charts showcases scatter plots of standardized house size versus standardized house price across ten different cities, indicating distinct demographic and market dynamics for each city.

2. Each city's data includes a posterior mean regression line (red) and HDI intervals (gray bands), highlighting variability in model predictions.

3. The linear trends across all cities suggest a positive correlation between house size and price, but the slope and spread vary by location, reflecting city-specific housing market characteristics.

4. The varying density and scatter of points in each city imply differing levels of variability and predictability of housing prices based on size.

5. This visualization aids in understanding the spatial heterogeneity in the model's performance, which is crucial for targeted interventions or policy decisions in real estate

markets.

6. The gray uncertainty bands around the posterior mean provide a sense of the model's confidence in its predictions. For most cities, the majority of observed data points fall within these intervals, suggesting that the model's uncertainty estimates are appropriate.

7. The Posterior Predictive Checks reveal heterogeneity across cities. For instance, cities like Chicago and Los Angeles exhibit tighter clustering of data points and narrower uncertainty bands, indicating more consistent predictive performance. In contrast, cities like Philadelphia and Tucson have larger spreads in the data and wider uncertainty intervals, suggesting greater variability or potential model misspecification in these cases.

Discussion

The analysis revealed a clear trend: house prices generally increase with an increase in house size, as expected. This relationship was consistent across most cities, affirming the intuitive understanding that larger homes command higher market values. Additionally, the model highlighted the influence of city-specific factors, with house prices being significantly higher in more expensive cities. This finding aligns with real-world market dynamics, where geographic and economic factors drive property values in premium locations. By including City as a confounder, the model effectively disentangled the effect of house size on price from city-level variations, providing a more accurate estimate of the causal relationship.

Future Considerations

To further enrich the study, other potential confounders such as the number of bedrooms and bathrooms can be incorporated into the model. These features are critical components of housing valuation, as homes with more bedrooms and bathrooms often appeal to larger families or buyers seeking premium amenities, potentially driving higher prices. Including these additional confounders would allow for a more comprehensive understanding of how various property features contribute to pricing. It would also enable a comparative analysis between the effects of house size, bedrooms, and bathrooms on price, offering insights into which factors have the most significant impact.

Extended Analysis

Such an extended analysis could reveal interactions between these confounders as well—such as whether larger houses with fewer bedrooms or bathrooms exhibit different price trends compared to smaller houses with better amenities.